ISSN 2464–4617 (print) ISSN 2464–4625 (CD-ROM)

CSRN 2901

(Eds.)

- **Vaclav Skala**
  University of West Bohemia, Czech Republic

Computer Science Research Notes

27. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision
WSCG 2019
Plzen, Czech Republic
May 27 – 30, 2019

Proceedings

WSCG 2019

Proceedings – Part I

ISSN 2464–4617 (print) ISSN 2464–4625 (CD-ROM)

ISSN 2464–4617 (print) ISSN 2464–4625 (CD-ROM)

CSRN 2901

(Eds.)

- **Vaclav Skala**
  University of West Bohemia, Czech Republic

Computer Science Research Notes

27. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision
WSCG 2019
Plzen, Czech Republic
May 27 – 30, 2019

Proceedings

WSCG 2019

Proceedings – Part I

Vaclav Skala - UNION Agency

ISSN 2464–4617 (print) ISSN 2464–4625 (CD-ROM)

Computer Science Research Notes
CSRN 2901

Editor-in-Chief: Vaclav Skala
c/o University of West Bohemia
Univerzitni 8
CZ 306 14 Plzen
Czech Republic
skala@kiv.zcu.cz http://www.VaclavSkala.eu

Managing Editor: Vaclav Skala

Publisher & Author Service Department & Distribution:
Vaclav Skala - UNION Agency
Na Mazinach 9
CZ 322 00 Plzen
Czech Republic
Reg.No. (ICO) 416 82 459

Published in cooperation with the University of West Bohemia
Univerzitní 8, 306 14 Pilsen, Czech Republic

ISSN 2464-4617 (Print) ***ISSN 2464-4625*** (CD/DVD)
ISBN 978-80-86943-37-4 (CD/-ROM)

WSCG 2019

International Program Committee

Anderson,M. (Brazil)
Baranoski,G. (Canada)
Benes,B. (USA)
Benger,W. (Austria)
Bilbao,J. (Spain)
Bouatouch,K. (France)
Bourke,P. (Australia)
Coquillart,S. (France)
Dachsbacher,C. (USA)
Durikovic,R. (Slovakia)
Feito,F. (Spain)
Ferguson,S. (United Kingdom)
Galo,M. (Brazil)
Gavrilova,M. (Canada)
Giannini,F. (Italy)
Gudukbay,U. (Turkey)
Guthe,M. (Germany)
Chaudhuri,D. (India)
Chover,M. (Spain)
Jeschke,S. (Austria)
Juan,M. (Spain)
Klosowski,J. (USA)
Lee,J. (USA)
Liu,S. (China)
Lobachev,O. (Germany)
Manoharan,P. (India)
Max,N. (USA)
Molla,R. (Spain)
Montrucchio,B. (Italy)
Muller,H. (Germany)
Pan,R. (China)
Paquette,E. (Canada)
Pedrini,H. (Brazil)
Richardson,J. (USA)
Rojas-Sola,J. (Spain)
Sarfraz,M. (Kuwait)
Savchenko,V. (Japan)
Segura,R. (Spain)
Skala,V. (Czech Republic)
Sousa,A. (Portugal)
Stuerzlinger,W. (Canada)
Szecsi,L. (Hungary)
Thalmann,D. (Switzerland)
Tokuta,A. (USA)
Torrens,F. (Spain)
Trapp,M. (Germany)
Wuensche,B. (New Zealand)
Wuethrich,C. (Germany)
Yin,Y. (USA)

WSCG 2019

Board of Reviewers

Anderson,M. (Brazil)
Assarsson,U. (Sweden)
Ayala,D. (Spain)
Baranoski,G. (Canada)
Benes,B. (USA)
Benger,W. (Austria)
Bilbao,J. (Spain)
Birra,F. (Portugal)
Bouatouch,K. (France)
Boukaz,S. (France)
Bourke,P. (Australia)
Cakmak,H. (Germany)
Carmo,M. (Portugal)
Carvalho,M. (Brazil)
Coquillart,S. (France)
Dachsbacher,C. (USA)
De Martino,J. (Brazil)
Durikovic,R. (Slovakia)
Eisemann,M. (Germany)
Feito,F. (Spain)
Ferguson,S. (United Kingdom)
Galo,M. (Brazil)
Gavrilova,M. (Canada)
Gdawiec,K. (Poland)
Gerhards,J. (France)
Giannini,F. (Italy)
Goncalves,A. (Portugal)
Gudukbay,U. (Turkey)
Guthe,M. (Germany)
Hansford,D. (USA)
Hermosilla Casajus,P. (Germany)
Chaudhuri,D. (India)
Chover,M. (Spain)
Ivrissimtzis,I. (United Kingdom)
Jeschke,S. (Austria)
Joan-Arinyo,R. (Andorra)
Jones,M. (United Kingdom)
Juan,M. (Spain)
Kanai,T. (Japan)
Klosowski,J. (USA)
Komorowski,J. (Poland)
Krueger,J. (Germany)
Kurt,M. (Turkey)
Kyratzi,S. (Greece)
Larboulette,C. (France)
Lee,J. (USA)
Lisowska,A. (Poland)
Liu,S. (China)
Lobachev,O. (Germany)
Manoharan,P. (India)
Manzke,M. (Ireland)
Mastermayer,A. (Germany)
Max,N. (USA)
Meyer,A. (France)
Mohd Suaib,N. (Malaysia)
Molla,R. (Spain)
Montrucchio,B. (Italy)

Muller,H. (Germany)
Nishio,K. (Japan)
Oliveira,J. (Portugal)
Oyarzun Laura,C. (Germany)
Pan,R. (China)
Papaioannou,G. (Greece)
Paquette,E. (Canada)
Pasko,A. (United Kingdom)
Pedrini,H. (Brazil)
Ramires Fernandes,A. (Portugal)
Richardson,J. (USA)
Rodrigues,N. (Portugal)
Rojas-Sola,J. (Spain)
Sarfraz,M. (Kuwait)
Savchenko,V. (Japan)
Segura,R. (Spain)
Shum,H. (United Kingdom)
Sik-Lanyi,C. (Hungary)
Skala,V. (Czech Republic)
Sousa,A. (Portugal)
Stuerzlinger,W. (Canada)
Szecsi,L. (Hungary)
Thalmann,D. (Switzerland)
Todt,E. (Brazil)
Tokuta,A. (USA)
Toler-Franklin,C. (USA)
Torrens,F. (Spain)
Trapp,M. (Germany)
Tytkowski,K. (Poland)
Vanderhaeghe,D. (France)
Weber,A. (Germany)
Wuensche,B. (New Zealand)
Wuethrich,C. (Germany)
Cai,Y. (Singapore)
Yin,Y. (USA)
Yoshizawa,S. (Japan)
Zwettler,G. (Austria)

CSRN 2901

WSCG 2019 Proceedings
Part I

Contents

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

Cubic Spline Interpolation in Real-Time Applications using Three Control Points

Jens Ogniewski
Linköping University
Information Coding Group
Department of Electrical Engineering
581 83 Linköping, Sweden
jens.ogniewski@liu.se

ABSTRACT

Spline interpolation is widely used in many different applications like computer graphics, animations and robotics. Many of these applications are run in real-time with constraints on computational complexity, thus fueling the need for computational inexpensive, real-time, continuous and loop-free data interpolation techniques. Often Catmull-Rom splines are used, which use four control-points: the two points between which to interpolate as well as the point directly before and the one directly after. If interpolating over time, this last point will ly in the future. However, in real-time applications future values may not be known in advance, meaning that Catmull-Rom splines are not applicable. In this paper we introduce another family of interpolation splines (dubbed Three-Point-Splines) which show the same characteristics as Catmull-Rom, but which use only three control-points, omitting the one "in the future". Therefore they can generate smooth interpolation curves even in applications which do not have knowledge of future points, without the need for more computational complex methods. The generated curves are more rigid than Catmull-Rom, and because of that the Three-Point-Splines will not generate self-intersections within an interpolated curve segment, a property that has to be introduced to Catmull-Rom by careful parameterization. Thus, the Three-Point-Splines allow for greater freedom in parameterization, and can therefore be adapted to the application at hand, e.g. to a requested curvature or limitations on acceleration/deceleration. We will also show a method that allows to change the control-points during an ongoing interpolation, both with Thee-Point-Splines as well as with Catmull-Rom splines.

Keywords

Spline Interpolation, Parameterization, Real-Time, Animation, Catmull-Rom Splines

1 INTRODUCTION

Spline interpolation is widely used for the creation of smooth paths e.g. in computer graphics or for the motion of robots. These splines have typically to fulfill four mathematical properties:

1. they have to be (at least) C^1 continuous,
2. they have to be interpolating splines, i.e. pass through all of their control points,
3. they have to be affine invariant, i.e. the sum of all blending polynomials is always 1, for all values of the interpolation variable, and
4. they must allow for local control, i.e. each control-point influences only a small, compact part of the overall generated curve.

Probably the most used interpolating splines are Catmull-Rom splines (CR), which fulfill all of the above properties, are fast to calculate and simple to use. CR use 4 control points - the two defining the current interpolation segment as well as the one directly before and the one directly after; if interpolating over time the latter point will ly in the future. However, in real-time applications limited information is available about future points, especially there might be no future information available apart from the next control point, rendering CR and all other conventional interpolation splines unsuitable. The spline in this paper, called Three-Point-Splines (TPS), exhibits the same properties as CR, but does not rely on the knowledge of any future points, and is thus applicable even in these scenarios.

In some cases the path to the next control point might be blocked, e.g. by a moving object, normally neces-

sitating motion replanning and probably stopping the current motion. As alternative, a method is introduced enabling to change the next control-point during an ongoing interpolation without loosing any of their properties, for both TPS and CR. This can be used e.g. in animations for computer games or for instant reactions of robots to a changed environment, thus enabling the robots to perform safer and more efficiently, a critical factor for their success in real-world applications [ET14].
Furthermore, TPS exhibit a very rigid behavior, and less variation as comparable splines (like e.g. CR). Because of this the curves generated by TPS are naturally self-intersection-free, a property that has to be introduced to CR by careful parameterization. In contrast, TPS allow for a range of different parameterizations that all are guaranteed to be self-intersection-free, and can therefore be adapted to meet the requirements of the application like e.g. limits on speed or acceleration/deceleration. Typically, reparameterizations are needed to ensure that these limits are kept, which introduces computational overhead. Also, not all reparameterizations have an analytical solution (like e.g. arc-length parameterization).
Finally, it is shown that the TPS are of lower computational complexity than CR which is important for real-time systems with limited computational capabilities.
TPS have only been described once earlier, in a conference poster by the author [Ogn13]. This paper extends the earlier work by describing the exact properties of TPS and by introducing a recursive evaluation form and parameterization scheme, needed for artifact-reduction. The change of the next control-point during an ongoing interpolation has not been described earlier.

1.1 Related Work

TPS are asymmetric splines that use three control-points. Asymmetric interpolation, in the sense of using a different number of points before (i.e. "in the past" of) and after (i.e. "in the furture" of) the current interpolation segment, was first proposed in [Yua96], albeit in a more theoretical-stochastic fashion, without a discussion of specific splines or their mathematical properties. Interpolation using three points was first explored in [Dod97]. However, in their work only functions of second degree were considered, and thus it was not possible to construct an interpolating spline and reach C^1 continuity at the same time. Furthermore, their interpolation scheme is symmetric, and uses a point in the past for the first half of the interpolation segment and a point in the future for the second half. An interpolating spline which does not use future but reaches a higher continuity than C^0 has to be asymmetric by definition.
Edwin Catmull and Raphael Rom introduced their splines already in [CR74], and later publications built on this initial approach, for example [DB88] or [KB84] which explored the basic properties of the splines further. The latter one even introduced an additional relaxation term, with which C^1 continuity can no longer be guaranteed though. Both did not explore more advanced parameterization techniques, which were made possible by the introduction of recursive evaluation forms in [JAW67]. Correct parameterization has been proven to be a vital part of spline interpolation, since uniform parameterizations can lead to unwanted behavior like loops. Thus, a number of different publications have examined this issue, often based on the findings of [IDS99], where for the first time a scheme was proposed to include non-uniform parameterization in CR and similar splines. Two of the most popular schemes are the centripetal [Lee89] and the chordal [ML88] scheme, which have been applied to CR in [NDH09]. The work described in [CYK11] built on this further by applying methods introduced in [DM96] and [Flo08] to prove that the centripetal parameterization is the only scheme that guarantees no self-intersections in CR.

1.2 Definitions

In this paper piecewise polynonmial interpolation is considered over a number of control-points $\mathbf{P}_i$, where the current interpolation segment $\mathbf{r}$ is interpolating between the control-points $\mathbf{P}_r$ and $\mathbf{P}_{r+1}$. Although only the properties of this segment are considered, its properties apply to all other segments as well, and especially the continuation between segments is guaranteed if not stated otherwise.
The following definitions are used:

$$\mathbf{D}_{a,b} = \mathbf{P}_a - \mathbf{P}_b$$

for the difference vector (between two points), and

$$\mathbf{l}_{(t)} = t\mathbf{D}_{r+1,r} + \mathbf{P}_r, t \in \mathbb{R}$$

for the line spawned by the two control-points of the interpolation segment.
The rest of this paper is organized as follows: section 2 introduces the uniform form of TPS interpolation, section 3 its recursive version, section 4 shows how control-points can be changed during an ongoing interpolation (for both TPS and for CR), and section 5 concludes the paper. Proofs and Derivations can be found in the appendix.

ISBN 978-80-86943-37-4

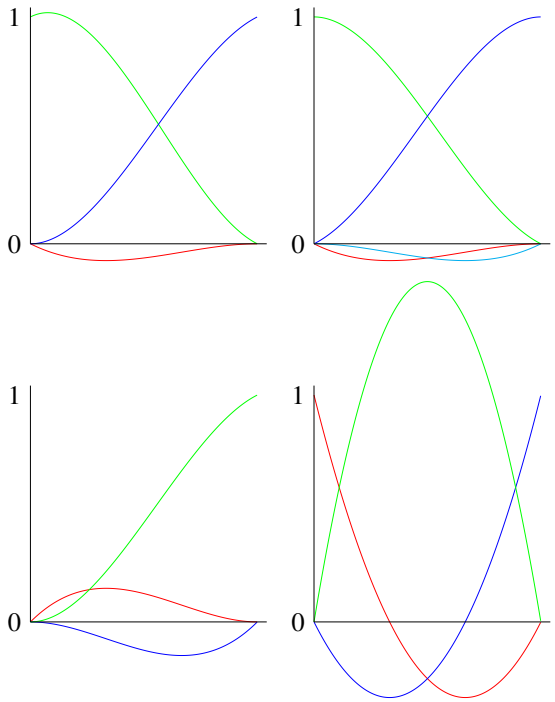


Figure 1: Blending polynomials:
Upper row: Three-Point-Spline (left) and Catmull-Rom (right): Red: polynomial for $\mathbf{P}_{r-1}$, green: polynomial for $\mathbf{P}_r$ and blue: polynomial for $\mathbf{P}_{r+1}$; Catmull-Rom includes an additional polynomial (cyan) for $\mathbf{P}_{r+2}$; all polynomials use an α of 0.5
Lower row: the polynomials for the difference vector form (left) and their derivatives (right):
$u(u-1)^2$ (red), $u^2(u-1)$ (blue) and $u^2(3-2u)$ (green)

2 DERIVATION OF THE UNIFORM FORM

Taking the aforementioned mathematical properties as constraints, the interpolation equations are set up accordingly and lead to the following equation, with u as interpolation variable, $0 \leq u \leq 1$:

$$\begin{aligned}\mathbf{F}_{TPS(u)} = \quad & (-\alpha u^3 + 2\alpha u^2 - \alpha u)\mathbf{P}_{r-1} \\ + \quad & (2u^3 - (3+\alpha)u^2 + \alpha u + 1)\mathbf{P}_r \\ + \quad & ((\alpha-2)u^3 + (3-\alpha)u^2)\mathbf{P}_{r+1}\end{aligned} \tag{1}$$

with α a parameterization value that can be chosen freely. Figure 1 shows the blending polynomials, with the ones from CR for comparison.
These equation can be rewritten to be based on the differences of points rather than based on points:

$$\begin{aligned}\mathbf{F}_{TPS(u)} = \quad & \alpha u(u-1)^2\mathbf{D}_{r,r-1} \\ + \quad & \alpha u^2(u-1)\mathbf{D}_{r+1,r} \\ + \quad & u^2(3-2u)\mathbf{D}_{r+1,r} \\ + \quad & \mathbf{P}_r\end{aligned} \tag{2}$$

where the first two terms blend the derivatives at points $\mathbf{P}_r$ and $\mathbf{P}_{r+1}$ respectively, while being 0 at both endpoints. The rest of the equation interpolates between the two points of the segment, while having a zero derivative at the endpoints (see also figure 1). Thus, this form gives a good overview of how the spline will behave (a similar form was already used in [KB84] albeit not for TPS). For comparison, uniform CR can be written in a similar way:

$$\begin{aligned}\mathbf{F}_{CR(u)} = \quad & \alpha u(u-1)^2\mathbf{D}_{r+1,r-1} \\ + \quad & \alpha u^2(u-1)\mathbf{D}_{r+2,r} \\ + \quad & u^2(3-2u)\mathbf{D}_{r+1,r} \\ + \quad & \mathbf{P}_r\end{aligned} \tag{3}$$

Note that CR includes an additional "future" point $\mathbf{P}_{r+2}$. The main difference between CR and TPS is which difference vectors are used for the derivatives, which has a high impact on how the splines behave. Especially, using the same difference vector in the second and the third term is exactly what gives the TPS their rigid behavior.
Incidentally, the α values for the different segments do not need to be the same, as long as they are the same where the same segment is concerned, i.e. the α value from the first term (in equation 2) needs to be the same α value used for the second term in the previous segment, or otherwise the interpolation will only be G^1 continuous, which is true for both CR and TPS.
Of special interest is of course the maximal distance $d_{\max}$ of the curve segment generated by TPS to the line $\mathbf{l}_{(t)}$, as already discussed in [Flo08] and [CYK11]. Also, this distance is needed to prove that TPS will never generate loops (for parameterizations inside a sensible bound), but might lead to overshots.
We define $\alpha a_{(u)} = \alpha u(u-1)^2$, blending $\mathbf{D}_{r,r-1}$, and $b_{(u)} = u^2((\alpha-2)u + (3-\alpha))$ (combining the second and the third blending polynomials of equation 2), blending $\mathbf{D}_{r+1,r}$. Inserting these functions into the point-line-distance equation yields (see also the appendix 7.1.2):

$$d_{(u)} = ||\mathbf{D}_{r,r-1}||\alpha a_{(u)}(1-\cos^2\theta)^{\frac{1}{2}} \tag{4}$$

with θ the angle between $\mathbf{D}_{r,r-1}$ and $\mathbf{D}_{r+1,r}$.
From the derivative $d'_{(u)}$ of equation 4 it is clear that only one extreme point exists for $0 < u < 1$, a maximum which lies at $u = \frac{1}{3}$, i.e. where the derivative of $a_{(u)}$ becomes zero. Thus, the position of the maximal distance of the curve segment from the line $\mathbf{l}_{(t)}$ lies at the same u-value in each segment and the maximum distance can therefore be easily calculated.
The three points used for the current interpolation segment ($\mathbf{P_{r-1}}$, $\mathbf{P_r}$, $\mathbf{P_{r+1}}$) span a plane on which all points of the generated curve for the current segment lie (the special case where all lie on the line $\mathbf{l}_{(t)}$ will be discussed later). The plane also contains the line $\mathbf{l}_{(t)}$. As was shown earlier, the distance to this line depends

ISBN 978-80-86943-37-4

only on the a-part of the interpolation function $\mathbf{F}_{CR(u)}$ (i.e. the b-part of the function lies completely on the $\mathbf{l}_{(t)}$), which is non-zero for $0 < u < 1$. Since a has no zero-point in the interpolation interval all points of the generated curve-segment lie on the same side of the line.

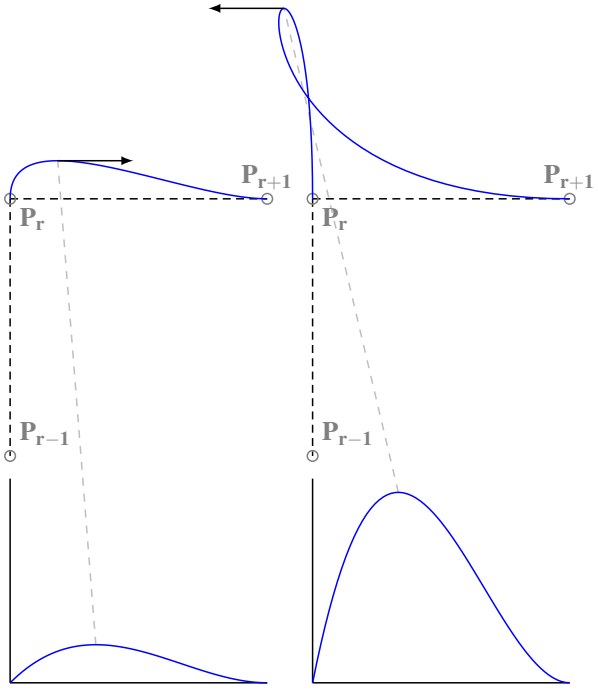


Figure 2: the different behavior of the three-point-spline with different αs: The interpolating curves (upper row) and the distance to the line $\mathbf{l}_{(t)}$ (lower row)
a) (left) with $\alpha = 1$, representing the case of $0 < \alpha < 3$
c) (right) $\alpha = 5$, representing the case of $\alpha >= 4$
The direction of the tangents in the respective extreme-points (i.e. the maximum distances to $\mathbf{l}_{(t)}$) are visualized by arrows; note that they are not in the same scale as the curves for better visualization

The interpolation function can be divided into two parts: a part e whose tangents are parallel to the line (i.e. which blend the vector $\mathbf{D}_{r+1,r}$) and a part f whose tangents are perpendicular to the line. While the f function-part only depends on the a-part of the interpolation function (as was shown earlier), the e function-part might depend on both a and b since $\mathbf{D}_{r,r-1}$ is not necessarily perpendicular to $\mathbf{D}_{r+1,r}$.
If the generated curve has a loop, both function-parts e and f change the signs of their tangents at some point in the loop (i.e. move into one direction first, than into the other, see also figure 2), which means that the loop has to contain an extreme point for both e and f, i.e. both function-parts need to have an extreme point in the interval $0 < u < 1$. It was already shown that the a-function-part has a maximum, but since it is only depending on one vector (i.e. all points generated by a lie on a segment of the line through $\mathbf{P_{r-1}}$ and $\mathbf{P_r}$) this is not enough to cause a loop, and whether a loop exist or not depends on the extreme-points of b. Looking at the derivative of b it becomes obvious that it contains one at $u = 0$ (which it does by definition of the blending functions), and another one that depends on the chosen value for α:

$\alpha = 0$: at $u = 1$
$0 < \alpha < 2$: at $u > 1$
$\alpha = 2$: at $u = 0$
$2 < \alpha < 3$: at $u < 0$
$\alpha = 3$: at $u = 0$
$\alpha > 3$: at $0 < u < 1$

This means that for $0 <= \alpha <= 3$ b does not have an extreme point in the interpolation interval and thus the generated curve will not contain a loop.
For $\alpha > 3$, the first derivative of the b-function-part will be negative in the beginning of the interpolation segment, and become positive after the extreme-point. If this extreme point lies after or at the extreme-point of a (which is located at $u = 1/3$), the generated curve will contain a loop. Inserting $u = 1/3$ int b' yields:

$$b'_{(\frac{1}{3})} = (\tfrac{4}{3} - \tfrac{1}{3}\alpha), \tag{6}$$

Which means that the generated curve will contain a loop for $\alpha >= 4$. The interval $3 < \alpha < 4$ would need further analysis, but this is omitted here since the interval $0 <= \alpha <= 3$ is deemed to be large enough, since larger αs lead to curves with a large maximal distance to $\mathbf{l}_{(t)}$, and are therefore of limited use. Also, note that in CR freedom of loops is much harder to introduce, in fact only centripetal CR fulfills it [CYK11]; the reason for the comparable large range of possible parameterizations of TPS which are guaranteed to be free from loops lies in the rigid behavior of TPS.
Unfortunately, the case where all points of the current segment lie on the line $\mathbf{l}_{(t)}$ is harder to examine. For $\mathbf{D}_{r+1,r}$ and $\mathbf{D}_{r,r-1}$ pointing in exact opposite directions the generated line will contain a segment where the interpolation function moves into one direction first, than in the other. If $\mathbf{D}_{r+1,r}$ and $\mathbf{D}_{r,r-1}$ are pointing in the exact same direction this depends on the length of these two vectors, and on the value chosen for α. On a side-note, setting $\alpha = 1$ means that the function will behave like a linear blending function for $\mathbf{D}_{r,r-1} = \mathbf{D}_{r+1,r}$ (as with Catmull-Rom for $\alpha = 0.5$ and $\mathbf{D}_{r,r-1} = \mathbf{D}_{r+1,r} = \mathbf{D}_{r+2,r+1}$), which means it will interpolate with a constant first derivative i.e. it will move with a constant speed if interpolating over time.
Since the distance to $\mathbf{l}_{(t)}$ depends on α, setting α to a low value will lead to a curve that deviates only slightly from $\mathbf{l}_{(t)}$. However, this will also result in large changes of the length of the tangent vectors (i.e. changes in the

ISBN 978-80-86943-37-4

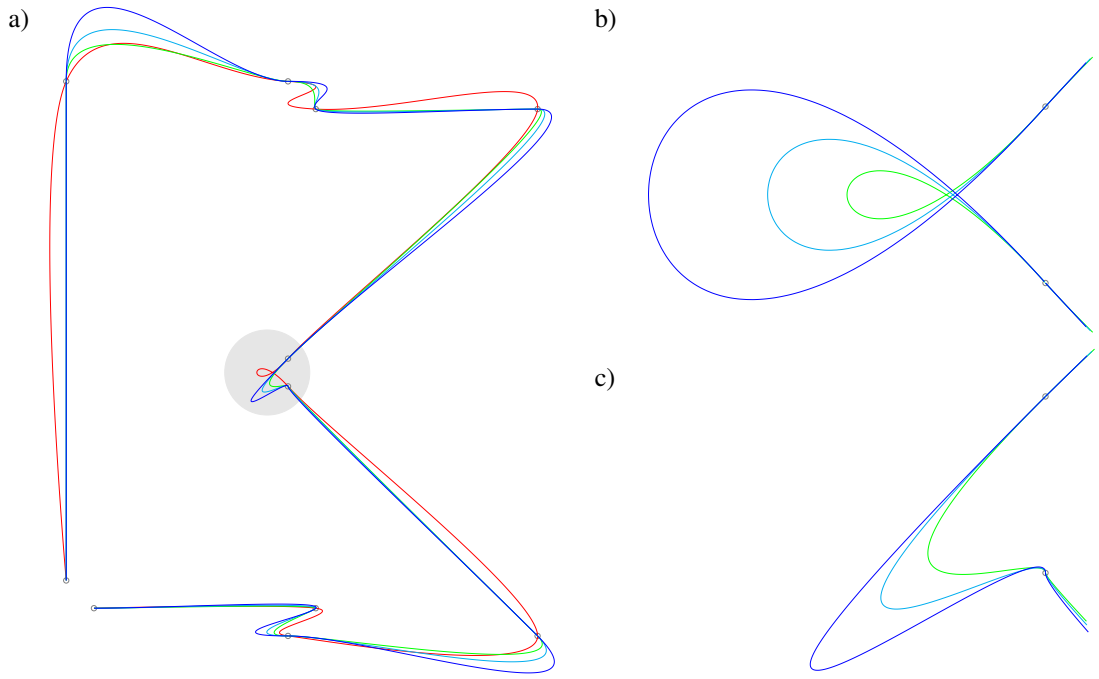


Figure 3: Comparison of CR and TPS with different parameterizations:
a) Left: CR with $\alpha = 0.5$ (red), TPS with $\alpha = 0.5$ (green), $\alpha = 0.7$ (cyan) and $\alpha = 1$ (blue)
Right: zoomed in on the light-gray circle:
b) Top right: CR and c) Bottom right: TPS, both with parameterizations of $\alpha = 0.5$ (green), $\alpha = 0.7$ (cyan) and $\alpha = 1$ (blue)

speed if used for animations or robot movement), since a small α results in short tangents in the control-points, but not all terms are bound by α (see also equation 2 and figure 1, to the right). Thus the tangents in the middle of the curve segments would be comparably large. Depending on the applications, limits can be determined on how far each curve segment is allowed to stray from $\mathbf{l}_{(t)}$, and how large changes of the lengths of the tangent-vectors are allowed to be, and thus a good value for α can be determined.
It is interesting to look at the derivative of TPS for $\alpha = 1.0$:

$$\begin{aligned} \mathbf{F}'_{TPS(u)} &= (3u^2 - 4u + 1)\mathbf{D}_{r,r-1} + (4u - 3u^2)\mathbf{D}_{r+1,r} \\ &= (4u - 3u^2)(\mathbf{D}_{r+1,r} - \mathbf{D}_{r,r-1}) + \mathbf{D}_{r,r-1} \end{aligned}$$

Thus, the derivative becomes an interpolation itself, between the derivatives at the point $\mathbf{P}_r$ and $\mathbf{P}_{r+1}$.
A positive α value means also that $\mathbf{P}_{r-1}$ and the point with the maximum distance to $\mathbf{l}_{(t)}$ will ly on opposite sides of $\mathbf{l}_{(t)}$. This means that if $\mathbf{P}_{r+2}$ lies between $\mathbf{l}_{(t)}$ and the generated curve segment, the next curve segment will cross the current one, thus generating a self-intersection in consecutive segments. For more than two dimensions, the points $\mathbf{P}_{r-1}$, $\mathbf{P}_r$, $\mathbf{P}_{r+1}$ and $\mathbf{P}_{r+2}$ need to lie in the same plane for this self-intersection to happen, since each curve segment of course lies completely in the plane spanned by the three points it uses for the interpolation. The reason for these possible self-intersections lies in the fact that it does not include a term containing the next interpolation segment (e.g. $\mathbf{D}_{r+2,r}$). Without this information, it is of course impossible to construct a curve that is guaranteed to be free of self-intersections in consecutive segments. Note that this applies even to the recursive form shown in the next section.
A comparison between different parameterizations of TPS and CR is given in figure 3. Although TPS do not lead to loops like CR, it introduces other artifacts in the form of overshots. Taking a look at the distance of the generated curve to the line $\mathbf{l}_{(t)}$ (equation 4), it becomes clear that it depends on the length of the last segment $\mathbf{D}_{r,r-1}$ rather than on the length of the current segment $\mathbf{D}_{r+1,r}$. Thus, if a long line segment is followed by a much shorter line segment the interpolation will unfortunately always overshoot in the shorter segment. A better parameterization to conquer this issue will be developed in the next section.

3 RECURSIVE EVALUATION FORM AND PARAMETERIZATION

Using TPS, one might be tempted to simply vary the different α values depending on the length of the

ISBN 978-80-86943-37-4

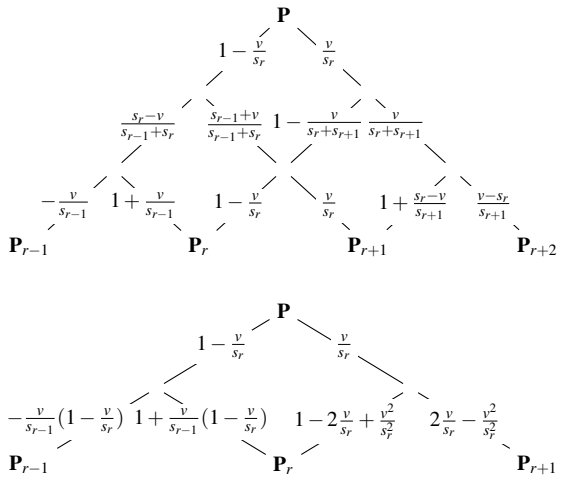


Figure 4: Different recursive formulations, with $s_r = ||D_{r+1,r}||^\beta$: a) (top) CR, b) (bottom) TPS

segment, but this can lead to unwanted side-effects, especially it would no longer be guaranteed that the α values would be inside the bounds established earlier. Instead, here one of the main ideas behind chordal, centripetal and other related parameterizations is used, namely varying the interpolation interval with the distance between the different points, using the interpolation variable v:

$$0 \leq v \leq s_r = ||\mathbf{D}_{r+1,r}||^\beta, \tag{7}$$

Conventionally, recursive CR-formulations are defined regarding to the sum of the lengths of the interpolation segments:

$$s'_0 = 0$$
$$s'_r = \sum_{i=0}^{r-1} s'_i + ||\mathbf{D}_{r,r-1}||^\beta \tag{8}$$

However, this more common formulation for recursive CR can be easily received from the one given here, and vice versa.

Using the formulation of (equation 7) reveals that the derivative of recursive CR in point $\mathbf{P}_r$ is no longer a simple scalar product of $\mathbf{D}_{r+1,r-1}$ multiplied by the interpolation parameter α but becomes:

$$\mathbf{F}'_{CR,rec(v=0)} = \frac{\frac{s_r}{s_{r-1}}\mathbf{D}_{r,r-1}+\frac{s_{r-1}}{s_r}\mathbf{D}_{r+1,r}}{s_{r-1}+s_r} \tag{9}$$

(similar for $v = s_r$ since it is still C^1 continuous.)

The centripetal and similar parameterizations of CR are facilitated by the fact that they can be expressed by a recursive evaluation form. Thus, to be able to use similar parameterizations of TPS they are first expressed in a recursive form as well, which is given in figure 4b), with the corresponding form of CR in figure 4a), and example interpolations in figure 5.

The function of the distance to the line $\mathbf{l}_{(t)}$ (equation 4) becomes for recursive TPS:

$$d_{(v)} = ||\mathbf{D}_{r+1,r}||^\beta ||\mathbf{D}_{r,r-1}||^{1-\beta} a'_{(v)} \left(1-\cos^2\theta\right)^{\frac{1}{2}}$$

with $a'_{(v)} = \frac{v}{s_r}(\frac{v}{s_r}-1)^2$ (10). (See also the appendix 7.2.1 for a more detailed derivation.)

Thus, β can be seen as a blending factor of how much the length of the current segment influences the distance to the infinite line $\mathbf{l}_{(t)}$, and how much the segment before it influences this distance. For $\beta = 1$ the distance to $\mathbf{l}_{(t)}$ is only depending on the length of the current segment, and not at all on the length of the segment before. For $\beta = 0$ it behaves exactly like equation 2 for $\alpha = 1$, i.e. it becomes a uniform parameterization. In the same way, the recursive formulation of CR becomes the uniform parameterization with $\alpha = 0.5$ by setting β to 0.

Since the distance to $\mathbf{l}_{(t)}$ is now bounded by the length of the current segment the generated curves do not overshoot anymore while interpolating comparably small segments, as can be seen in figure 5, which shows CR and TPS with different parameterizations. Also, an additional curve was added (in magenta) which uses different values for β, depending on the length of the segment. The idea behind this is to minimize the $||\mathbf{D}_{r+1,r}||^\beta$ and $||\mathbf{D}_{r,r-1}||^{1-\beta}$ factors (and thus the distance to $\mathbf{l}_{(t)}$)as much as possible.

Again it is possible to use different β values for different segments, just as it is to use different α valuess in the uniform versions, but again the same β value has to be used for the same segment, i.e. the β value used to blend $\mathbf{D}_{r,r-1}$ has to be equal to the β value used to blend $\mathbf{D}_{r+1,r}$ during the interpolation of the last segment, to maintain C^1 continuity. This applies to both TPS and CR, however for CR only the centripetal ($\beta = 0.5$) parameterization is guaranteed to be free from loops. For TPS β can be chosen freely, without risk for self-intersections (in the same interpolation segment), as will be shown next.

Similar as in section 2 the places of the extreme points can be found of the distance function to $\mathbf{l}_{(t)}$ for recursive TPS, and again only one extreme point exists, a maximum at $v = \frac{1}{3s_r}$, i.e. again after $\frac{1}{3}$ of the interpolation interval, which is not surprising taking the similar derivatives of the blending weights for $\mathbf{D}_{r,r-1}$ into account. From the derivative it becomes obvious that the blending function b'_v for $\mathbf{D}_{r+1,r}$ is always positive in this point, i.e. the interpolated curve will never contain any loops, independent of which value β has (See also the appendix 7.2.1 for a more detailed derivation.). It should be pointed out however that whereas centripetal CR is also free from self-intersections in neighboring segments, this property cannot be introduced to the TPS, due to the fact that it is missing a term for the next segment in its

ISBN 978-80-86943-37-4

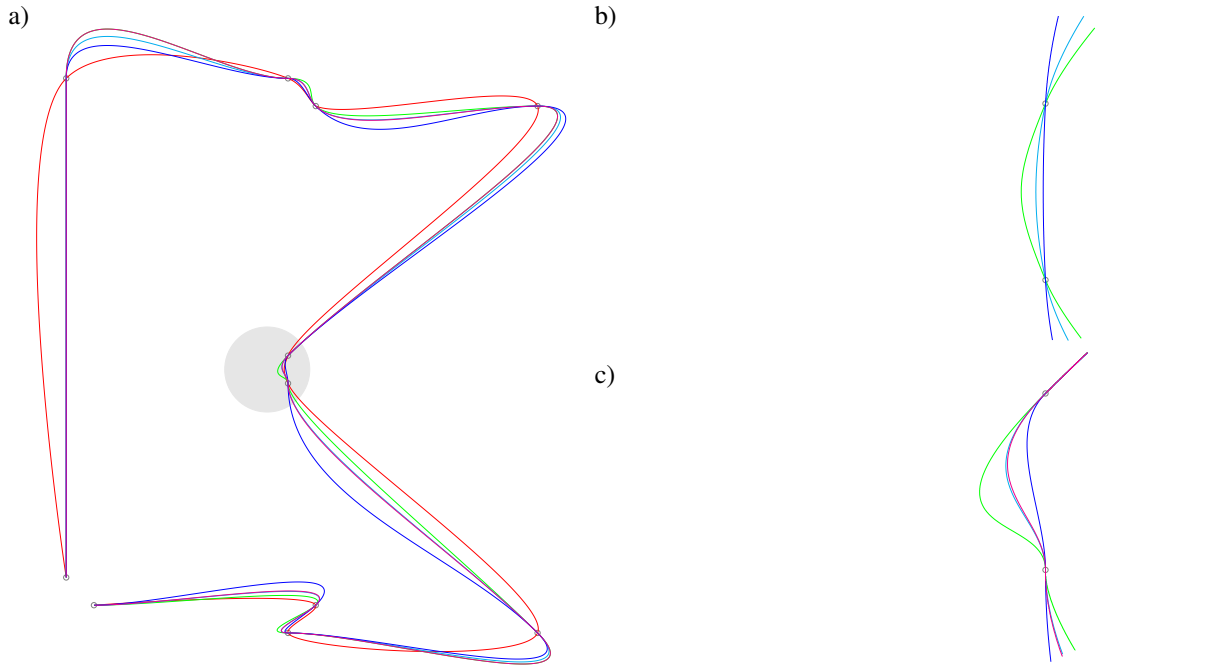


Figure 5: Comparison of centripetal CR and the recursive TPS with different parameterizations:
Left: centripetal CR (red), TPS with $\beta = 0.5$ (green), $\beta = 0.7$ (cyan) and $\beta = 1$ (blue)
a) Right: zoomed in on the light-gray circle:
b) Top right: CR and c) Bottom right: TPS, both with parameterizations of $\beta = 0.5$ (green), $\beta = 0.7$ (cyan) and $\beta = 1$ (blue)
An additional curve was added for a TPS with a variable parameterization, which uses $\beta = 0.5$ for the long segments and $\beta = 1$ for the short segment. Note that it in this example follows (and overlaps) very closely either the curve created by the TPS with $\beta = 0.5$ or the one with $\beta = 0.7$, depending on the segment.

equation, as already discussed in section 2. Thus, TPS will generate a self-intersection in two neighboring segments if the next control-point after the current segment lies in between $\mathbf{l}_{(t)}$ and the interpolation curve of the current segment and on the plane spanned by the three control-points used during the current segment.
Finally, since the recursive formulation of TPS behaves similar as the uniform version with $\alpha = 1$, the derivative of the recursive formulation of TPS is a blending function itself, of the derivatives at point $\mathbf{P}_r$ and $\mathbf{P}_{r+1}$ (see also the appendix 7.2.2):

$$\mathbf{F}'_{TPS,rec(v)} = (1 - h_{(v)}) \frac{\mathbf{D}_{r,r-1}}{||\mathbf{D}_{r,r-1}||^{\beta}} + h_{(v)} \frac{\mathbf{D}_{r+1,r}}{||\mathbf{D}_{r+1,r}||^{\beta}}$$
$$\text{with } h_{(v)} = 4\frac{v}{s_r} - 3(\frac{v}{s_r})^2 \qquad (11)$$

For a comparison in computation time both the uniform and the recursive version of both TPS and CR were implemented. For a fair comparison, each was implemented in different, hand-optimized ways and the respective fastest version was used; the reached optimizations should be comparable. The same $\alpha = \beta = 0.5$ were used for all the different interpolation methods and the same 1 000 000 randomly chosen control-points between which 100 intermediate points for each segment were interpolated. This was done on an Intel Core i7-6700 CPU running at 3.40GHz with 16 GBytes of RAM and gcc 5.4.0 with full optimization. The results are given in table 1.
It is counter-intuitive that the recursive version of TPS is faster than the uniform version, especially since it involves square roots - however with modern computers and compilers square roots can be computed very efficiently (in fact it took less than 4.5 s to calculate all the s values needed during the comparison) and need to be calculated only once for each interpolation segment. The remaining computation of the recursive version of TPS contains very few operations, less than needed for the uniform version. However, for the uniform version the minimal number of operations needed depends highly on the value chosen for α, and in fact for $\alpha = 1$ the uniform version needs fewer operations and thus performs marginally faster than the recursive version.

Computation time	uniform CR	uniform TPS	centripetal CR	recursive TPS
in seconds	1295	908	2009	699
in percent	65	45	100	35

Table 1: Timing results for the different methods.

ISBN 978-80-86943-37-4

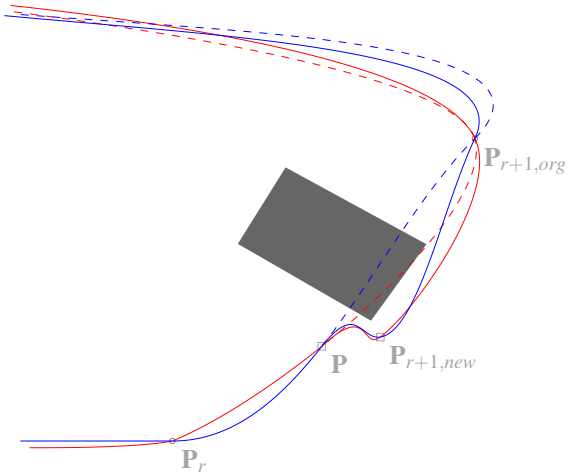


Figure 6: Example of a change of control-points during an ongoing interpolation:
A course is planned for a robot with centripetal CR (red) and recursive TPS (with $\beta = 0.5$, in blue) through a number of points given by the circles. However, during its run the robot notices a new obstacle (visualized by the gray rectangle) in point $\mathbf{P}$. The robot plots an evasion course through point $\mathbf{P}_{r+1,new}$ and thus is able to reach its destination without the need to stop for a replanning. The course it actually took is shown with solid lines, the planed one with dotted lines (note that it is partly overlapping with the robots actual course.)

4 CHANGE OF CONTROL POINTS DURING AN ONGOING INTERPOLATION

Here, it will be shown how it is possible to change the next control-point ($\mathbf{P}_{r+1}$) during an ongoing interpolation, without losing any of the mathematical properties, for both TPS and CR. This is useful in many real-time scenarios, an example might be a robot which encounters a previously unknown obstacle that it needs to circumvent.
The basic idea is to stop the ongoing interpolation, and start a new one in the current point $\mathbf{P}$, which thus becomes $\mathbf{P}_{r,new}$. The derivative in this point $\mathbf{P}' = \mathbf{P}'_{r,new}$ is calculated using the current interpolation interval, and a new new destination point $\mathbf{P}_{r+1,new}$ is chosen. The derivative of the recursive formulation of TPS in point $v = 0$ is:

$$\mathbf{F}'_{TPS,rec(0)} = \frac{1}{||\mathbf{D}_{0,r-1}||^{\beta_0}} \mathbf{D}_{0,r-1} \tag{12}$$

Thus, setting $\mathbf{D}_{0,r-1} = \mathbf{P}'$ and $\beta_0 = 0$ for the new interpolation segment maintains C^1 continuity. (Alternatively, if β_0 is chosen freely the generated curve will be G^1 continuous in point $\mathbf{P}$.)
In a similar way the next control-point can be changed during an ongoing interpolation using recursive CR. Equation 9 describes the derivative for CR and $v = 0$. We set again the β_0 for segment $\mathbf{D}_{r,r-1}$ to zero, as well as equation 9 $\mathbf{F}'_{CR,rec(0)} = \mathbf{P}'$, and solve for $\mathbf{D}_{r,r-1}$:

$$\mathbf{D}_{r,r-1} = \frac{(1+s_r)\mathbf{P}' - \frac{\mathbf{D}_{r+1,r}}{s_r}}{s_r} \tag{13}$$

Note that in contrast to TPS, in CR setting β_0 to another value but zero will not lead to G^1 continuity, but will rather have no higher continuity than C^0.
Setting $\beta_0 = 0$ however means using an uniform parameterization, albeit for only one segment, which might lead to unwanted behavior (especially loops in the case of CR). However, although no proof for the existence or non-existence of such unwanted behavior can be presented, none were found during the experiments with neither CR nor TPS (i.e. no large overshoots either). An example of this method can be found in figure 6.
Of course, if the necessity to change the control-points becomes known beforehand (i.e. before entering the segment containing an afflicted control-point), no measures need to be taken in case of TPS. For CR, this is only the case if both the current and the next segment are not affected.

5 CONCLUSION / FUTURE WORK

Three-Point-Splines (TPS) have been introduced which have similar mathematical characteristics as Catmull-Rom (CR). However, in contrast to CR and similar splines TPS do not rely on the knowledge of future points. Thus TPS enable spline interpolation even in real-time scenarios where no future values are known. Also, it was shown that TPS can be computed faster than CR.
TPS are more rigid than CR and because of that are trivially free of self-intersection (inside the interpolation segment) without the need for a particular parameterization to avoid this. Thus the parameterization of TPS can be adapted to the application, but ideal values for different applications still need to be found. For CR, only one parameterization, centripetal, fulfills this property. However, centripetal CR is also free from self-intersections in directly consecutive segments, a property that cannot be introduced to TPS due to the fact that they do not include knowledge of future points.
Also, a method was presented how to change the next control-point during an ongoing interpolation, for both TPS and CR, rather than having to do a computational expensive replanning. However, although no such behavior was observed in the experiments, it is not completely clear yet if this can lead to unwanted behavior like self-intersections (in the case of CR) or overshots (in the case of TPS).

ISBN 978-80-86943-37-4

6 ACKNOWLEDGMENTS

My special gratitude to Jan-Åke Larsson, Per-Erik Forssén, Ingmar Ragnemalm and Robert Forchheimer for their valuable help in writing this paper.

7 APPENDIX - PROOFS AND DERIVATIONS

In the following excessive use is made of the fact that $||\mathbf{P}|| = \sqrt{\mathbf{PP}}$ (note that since this describes a norm always the positive root has to be selected), and especially $(\frac{\mathbf{P}}{||\mathbf{P}||})^2 = \frac{\mathbf{PP}}{(\sqrt{\mathbf{PP}})^2} = 1$.

7.1 Properties of the Uniform Version

7.1.1 Formulation and Derivative

Separating equation 2 in two terms, one blending $\mathbf{D}_{r,r-1}$ and one blending $\mathbf{D}_{r+1,r}$ yields (with $0 <= u <= 1$):

$$\begin{aligned}\mathbf{F}_{CR(u)} &= \alpha u(u-1)^2\mathbf{D}_{r,r-1} \\ &+ u^2((\alpha-2)u+(3-\alpha))\mathbf{D}_{r+1,r}+\mathbf{P}_r \\ &= \alpha a_{(u)}\mathbf{D}_{r,r-1}+b_{(u)}\mathbf{D}_{r+1,r}+\mathbf{P}_r\end{aligned}$$

With the derivative:

$$\begin{aligned}\mathbf{F}'_{CR(u)} &= \alpha(3u^2-4u+1)\mathbf{D}_{r,r-1} \\ &+ (3(\alpha-2)u^2+2(3-\alpha)u)\mathbf{D}_{r+1,r} \quad (14) \\ &= \alpha a'_{(u)}\mathbf{D}_{r,r-1}+b'_{(u)}\mathbf{D}_{r+1,r}\end{aligned}$$

7.1.2 Distance to $l_{(t)}$

Inserting $\mathbf{F}_{CR(u)}$ into the point-line-distance function becomes:

$$\begin{aligned}d_{(u)} &= ||(\mathbf{F}_{CR(u)}-\mathbf{P}_r)-((\mathbf{F}_{CR(u)}-\mathbf{P}_r)\mathbf{D}_{r+1,r})\mathbf{D}_{r+1,r}|| \\ &= || \quad (\alpha a_{(u)}\mathbf{D}_{r,r-1}+b_{(u)}\mathbf{D}_{r+1,r}) \\ &\quad - ((\alpha a_{(u)}\mathbf{D}_{r,r-1}+b_{(u)}\mathbf{D}_{r+1,r})\tfrac{\mathbf{D}_{r+1,r}}{||\mathbf{D}_{r+1,r}||}) \\ &\quad \cdot \tfrac{\mathbf{D}_{r+1,r}}{||\mathbf{D}_{r+1,r}||}||\end{aligned}$$

Simplifying this equation leads to:

$$\begin{aligned}d_{(u)} &= || \quad \alpha a_{(u)}\mathbf{D}_{r,r-1}+b_{(u)}\mathbf{D}_{r+1,r} \\ &\quad - \alpha a_{(u)}\tfrac{\mathbf{D}_{r,r-1}\cdot\mathbf{D}_{r+1,r}}{\mathbf{D}_{r+1,r}\cdot\mathbf{D}_{r+1,r}}\mathbf{D}_{r+1,r} \\ &\quad - b_{(u)}\tfrac{\mathbf{D}_{r+1,r}\cdot\mathbf{D}_{r+1,r}}{||\mathbf{D}_{r+1,r}||\,||\mathbf{D}_{r+1,r}||}\mathbf{D}_{r+1,r}|| \\ &= \alpha a_{(u)}||\mathbf{D}_{r,r-1}-\tfrac{\mathbf{D}_{r,r-1}\cdot\mathbf{D}_{r+1,r}}{||\mathbf{D}_{r+1,r}||\,||\mathbf{D}_{r+1,r}||}\mathbf{D}_{r+1,r}|| \\ &= ||\mathbf{D}_{r,r-1}||\alpha a_{(u)} \\ &\quad \cdot ||\tfrac{\mathbf{D}_{r,r-1}}{||\mathbf{D}_{r,r-1}||}-\tfrac{\mathbf{D}_{r+1,r}\cdot\mathbf{D}_{r,r-1}}{||\mathbf{D}_{r+1,r}||\,||\mathbf{D}_{r,r-1}||}\tfrac{\mathbf{D}_{r+1,r}}{||\mathbf{D}_{r+1,r}||}|| \\ &= ||\mathbf{D}_{r,r-1}||\alpha a_{(u)}\,(\tfrac{\mathbf{D}_{r,r-1}\cdot\mathbf{D}_{r,r-1}}{||\mathbf{D}_{r,r-1}||\,||\mathbf{D}_{r,r-1}||} \\ &\quad - 2\tfrac{\mathbf{D}_{r+1,r}\cdot\mathbf{D}_{r,r-1}}{||\mathbf{D}_{r+1,r}||\,||\mathbf{D}_{r,r-1}||}\tfrac{\mathbf{D_{r+1,r}}}{||\mathbf{D_{r+1,r}}||}\tfrac{\mathbf{D_{r,r-1}}}{||\mathbf{D_{r,r-1}}||} \\ &\quad + (\tfrac{\mathbf{D}_{r+1,r}\cdot\mathbf{D}_{r,r-1}}{||\mathbf{D}_{r+1,r}||\,||\mathbf{D}_{r,r-1}||})^2\tfrac{\mathbf{D}_{r+1,r}\cdot\mathbf{D}_{r+1,r}}{||\mathbf{D}_{r+1,r}||\,||\mathbf{D}_{r+1,r}||})^{\frac{1}{2}} \\ &= ||\mathbf{D}_{r,r-1}||\alpha a_{(u)}(1-\cos^2\theta)^{\frac{1}{2}}\end{aligned}$$

with θ the angle between $\mathbf{D}_{r,r-1}$ and $\mathbf{D}_{r+1,r}$ (4)

7.2 Properties of the Recursive Version

7.2.1 Formulation and Derivative

From figure 4 the equation for the recursive version of TPS can be derived, for $0 \le v \le s_r = ||\mathbf{D}_{r+1,r}||^\beta$:

$$\begin{aligned}\mathbf{F}_{TPS,rec(v)} &= -\tfrac{v}{s_{r-1}}(1-\tfrac{v}{s_r})^2\mathbf{P}_{r-1}+\tfrac{v}{s_{r-1}}(1-\tfrac{v}{s_r})^2\mathbf{P}_r \\ &\quad + (1-\tfrac{v}{s_r})\mathbf{P}_r+\tfrac{v}{s_r}\mathbf{P}_r \\ &\quad - \tfrac{v}{s_r}(2\tfrac{v}{s_r}-\tfrac{v^2}{s_r^2})\mathbf{P}_r+\tfrac{v}{s_r}(2\tfrac{v}{s_r}-\tfrac{v^2}{s_r^2})\mathbf{P}_{r+1} \\ &= \tfrac{v}{s_{r-1}}(1-\tfrac{v}{s_r})^2(\mathbf{D}_{r,r-1}) \\ &\quad + \tfrac{v}{s_r}(2\tfrac{v}{s_r}-\tfrac{v^2}{s_r^2})(\mathbf{D}_{r+1,r})+\mathbf{P}_r \\ &= s_r((\tfrac{v}{s_r}-2(\tfrac{v}{s_r})^2+(\tfrac{v}{s_r})^3)\tfrac{\mathbf{D}_{r,r-1}}{s_{r-1}} \\ &\quad + (2(\tfrac{v}{s_r})^2-(\tfrac{v}{s_r})^3)\tfrac{\mathbf{D}_{r+1,r}}{s_r}+\mathbf{P}_r\end{aligned}$$

And from that the derivative:

$$\begin{aligned}\mathbf{F}'_{TPS,rec(v)} &= (1-4\tfrac{v}{s_r}+3(\tfrac{v}{s_r})^2)\tfrac{\mathbf{D}_{r,r-1}}{s_{r-1}} \\ &\quad + (4\tfrac{v}{s_r}-3(\tfrac{v}{s_r})^2)\tfrac{\mathbf{D}_{r+1,r}}{s_r} \\ &= (4\tfrac{v}{s_r}-3(\tfrac{v}{s_r})^2)(\tfrac{\mathbf{D}_{r+1,r}}{s_r}-\tfrac{\mathbf{D}_{r,r-1}}{s_{r-1}}) \\ &\quad + \tfrac{\mathbf{D}_{r,r-1}}{s_{r-1}}\end{aligned}$$

7.2.2 Distance to $l_{(t)}$

Inserting $\mathbf{F}_{TPS,rec(v)}$ in equation 4 (with $a_{(v)} = \frac{s_r}{s_{r-1}}(\frac{v}{s_r}(\frac{v}{s_r}-1)^2)$) leads to:

ISBN 978-80-86943-37-4

$$d_{(v)} = ||\mathbf{D}_{r,r-1}|| \frac{||\mathbf{D}_{r+1,r}||^{\beta}}{||\mathbf{D}_{r,r-1}||^{\beta}} (\frac{v}{s_r}(\frac{v}{s_r}-1)^2)(1-\cos^2\theta)^{\frac{1}{2}}$$
$$= ||\mathbf{D}_{r+1,r}||^{\beta} ||\mathbf{D}_{r,r-1}||^{1-\beta} a'_{(v)} (1-\cos^2\theta)^{\frac{1}{2}}$$
with $0 \leq a'_{(v)} = \frac{v}{s_r}(\frac{v}{s_r}-1)^2 \leq 1$

7.3 Equation and Derivative of recursive Catmull-Rom

The following is needed to be able to change the control-points during an ongoing interpolation as described in section 4. Again we use $0 \leq v \leq s_r = ||\mathbf{D}_{r+1,r}||^{\beta}$.

7.3.1 Formulation

$$\begin{aligned}\mathbf{F}_{CR(v)} = & \frac{s_1^2 v - 2 s_r v^2 + v^3}{s_{r-1} s_r (s_{r-1}+s_r)} \mathbf{D}_{r,r-1} \\ & + (\frac{s_{r-1} s_r v + 2 s_r v^2 - v^3}{s_r s_r (s_{r-1}+s_r)} + \frac{s_r v^2 - v^3}{s_r s_r (s_r + s_{r+1})}) \mathbf{D}_{r+1,r} \\ & + \frac{v^3 - s_r v^2}{s_r s_{r+1} (s_r + s_{r+1})} \mathbf{D}_{r+2,r+1} \\ & + \mathbf{P}_r\end{aligned}$$

7.3.2 First Derivative

$$\begin{aligned}\mathbf{F'}_{CR(v)} = & \frac{s_r^2 - 4 s_r v + 3 v^2}{s_{r-1} s_r (s_{r-1}+s_r)} \mathbf{D}_{r,r-1} \\ & + (\frac{s_{r-1} s_r + 4 s_r v - 3 v^2}{s_r s_r (s_{r-1}+s_r)} + \frac{2 s_r v - 3 v^2}{s_r s_r (s_r + s_{r+1})}) \mathbf{D}_{r+1,r} \\ & + \frac{3 v^2 - s_r 2 v}{s_r s_{r+1} (s_r + s_{r+1})} \mathbf{D}_{r+2,r+1}\end{aligned}$$

REFERENCES

[CR74] E. Catmull and R. Rom. A class of local interpolating splines. In *Computer Aided Geometric Design*, 1974.

[CYK11] S. Schaefer C. Yuksel and J. Keyser. Parameterization and applications of catmull-rom curves. In *Computer-Aided Design*, volume 43, pages 747–755, 2011.

[DB88] T. DeRose and B.A. Barsky. Geometric continuity, shape parameters, and geometric constructions for catmull-rom splines. In *ACM Transactions on Graphics*, volume 7, 1988.

[DM96] F. Canny D. Manocha. Detecting cusps and inflection points in curves. In *Computer-Aided Geometric Design*, volume 12, 1996.

[Dod97] N.A. Dodgson. Quadratic interpolation for image resampling. In *IEEE Transactions on Image Processing*, volume 6, 1997.

[ET14] P. Englert and M. Toussaint. Reactive phase and task space adaption for robust motion execution. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[Flo08] M.S. Floater. On the deviation of a parametric cubic spline interpolant from its data polygon. In *Computer-Aided Geometric Design*, volume 25, 2008.

[IDS99] I. Guskov I. Daubechies and W. Sweldens. Regularity of irregular subdivision. In *Constructive Approximation*, volume 15, 1999.

[JAW67] E.N. Nilson J.H. Ahlberg and J.L. Walsh. *The Theory of Splines and Their Applications*. Academic Press, 1967.

[KB84] D. H. U. Kochanek and R. H. Bartels. Interpolating splines with local tension, continuity, and bias control. In *ACM SIGGRAPH*, volume 18, 1984.

[Lee89] E.T.Y. Lee. Choosing nodes in parametric curve interpolation. In *Computer-Aided Geometric Design*, volume 21, 1989.

[ML88] R. Moore and J. Lopes. A recursive evaluation algorithm for a class of catmull-rom splines. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, volume 22, 1988.

[NDH09] M.S. Floater N. Dyn and K. Hormann. Four-point curve subdivision based on iterated chrordal and centripetal parameterization. In *Computer-Aided Geometric Design*, volume 26, 2009.

[Ogn13] Jens Ogniewski. C1-continuous, low-complex splines using 3 control points. In *Motion in Games*, 2013.

[Yua96] J.T. Yuan. Asymmetric interpolation lattice. In *IEEE Transactions on Signal Processing*, volume 44, 1996.

Creating digital models of paleoanthrological sample by photogrammetry and computed tomography

Novikov, Mikhail
FSRC "Crystallography and Photonics" RAS
Svyatoozerskaya, 1. Russia, 140700, Shatura
novikov@rambler.ru

Knyaz, Vladimir
State Research Institute of Aviation System (GosNIIAS)
7, Victorenko str., Russia, 125319, Moscow
knyaz@gosniias.ru

Galeev,Ravil
Institute of Ethnology and Anthropology RAS
Leninskiy prospect 32a, Russia, 119991, Moscow
ravil.galeev@gmail.com

ABSTRACT

The specificity of paleoanthropological researchs and accurate anthropological documentation is due to the complexity of the form of anthropological objects, their uniqueness, high historical and scientific value. The main purpose of this work is to develop methods for creating 3D models with a high degree of informatively for the development of digital documentation systems of paleoanthropological objects. The article presents a comparative analysis of digital models of anthropological object create using photogrammetry and computed tomography. It is shown that the combined use of non-contact methods of photogrammetry and computed tomography allows to create high-precision three-dimensional models with photorealistic texture and accurate internal and hidden geometry. The proposed approach allows not only to create virtual collections for wide sharing of specialists, but also using modern methods of additive manufacturing to make exact copies of unique artifacts.

Keywords

paleoanthropology; non-contact measurements; digital model; computed tomography; photogrammetry; additive technologies; three-dimensional documentation

1 INTRODUCTION

One of the main objectives of paleoanthropology is the morphological description of bone remains, using a number of measuring methods (craniometry, osteometry, etc.), allowing for extensive comparative studies, the purpose of which is to describe the diversity of anthropological types, both in ancient times and in modern times [Jur16a].

Despite the significant development of modern measuring equipment, hand tools are used in modern physical anthropology. In part, this is due to the fact that the developed measuring techniques of paleoanthropological objects can be used with a specific set of measurement instrument – stout, jig, sliding compass and other tools (Figure 1), in addition, the accuracy of manual measurements is in full compliance with the required accuracy (the measurement accuracy is up to 0.1 cm).

At the same time, manual measurement has a number of obvious shortcomings - it is, first of all, the complexity, time-consuming, often damage to the object of study, and, surprisingly, the extreme high cost of anthropological tools.

Modern non-contact methods of measurement allow to obtain huge arrays of geometric data of paleontological samples and to create on their basis high-precision

Figure 1: Mechanical tools for measuring anthropological objects

digital models for documenting and providing wide remote access to these data for specialists [Jur16a].

This is largely able not only to solve the above problems of manual measurement and eliminate their shortcomings, but also to expand the creative research horizons in general.

Paleoanthropology imposes special requirements on the created digital models [Web15], [Dav17]. Together with precision and the desired level of detail of the

paleoanthropological models must be exactly are textured with high resolution. In addition, it is often necessary to use digital models with different levels of detail in different parts of the object [Sub02].

In geometric morphometry, the change in shape, defined as the spatial distribution of benchmark, can be analyzed independently of size using a set of morphometric features. The coordinates of the labels can be obtained either by using the digitizer directly from the object, or on digital models created by computer tomography (CT) [Nys15] or surface scanning [Heg16]. Computer tomography also offers the possibility of measuring internal structures.

Anthropology has a long history of measuring biological form. Currently, there are several applications that provide the ability to include new 3D benchmarks and geometric morphometry methods in the study of changes in size and shape. This approach allows for original research [Cun14].

2 PHOTOGRAMMETRY

The paleoanthropological material for this study was obtained as a result of the excavation of Chernovaya VIII, carried out in 1962–1963. G.P. Maksimenkov, and dates back to II millennium BC. [Mak80]. More than 50 skulls suitable for research were found in 14 burial mounds of the burial ground. For studying the paleoanthropology of the Okunev culture, the collected series of skulls is one of the most representative in terms of both the number and the preservation of anthropological material [Rav16].

Textured surface 3D models of paleoanthropological objects were obtained using an original photogrammetric system developed in the State Research Institute of Aviation System [Kny16]. For purposes of paleoanthropological analysis 3D models of paleoanthropological objects have to be of high geometric accuracy and of high resolution (detailing). Additionally some special requirements are imposed for the 3D reconstruction system. They are as following:

- Short processing time
- Producing 3D model of a given object
- Texturing of the 3D model to capture visual features of scanning area

To meet these requirements the system configuration shown in Figure 2 is chosen.

It includes two high-resolution digital cameras, a computer-controlled table for rotating the object (optional), a special structured light projector to illuminate the sample, and a digital SLR camera for high-resolution texture generation. Original software supports in automated a set of the functions needed for creating accurate photorealistic textured 3D models. The main functions are:

- Automatic system calibration based on coded targets
- Scanning 2.5D fragments of an object
- Merging all partial 2.5D patches in single 3D mesh
- Accurate automatic mapping of the high resolution texture

The developed system is based on photogrammetric principle of spatial measurements, which allows determining 3D coordinates for any point of the object if its image correspondence for two oriented photographs is established. The system provides automated images orientation (calibration) and supports a set of methods of automated correspondence problem solution based on various structured light patterns such as scanning stripe, coded light, phase shift.

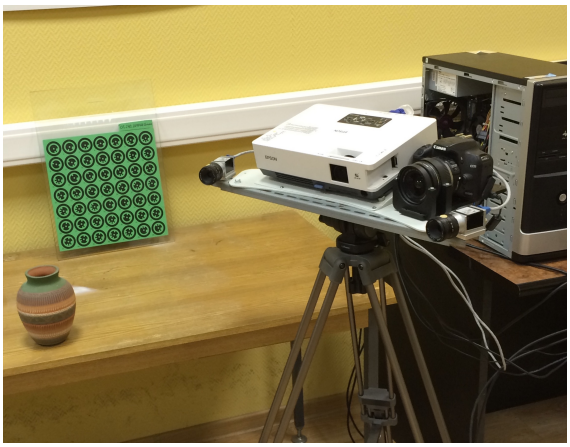


Figure 2: Photogrammetric system

The original calibration procedure [Kny10] is used to compensate geometrical distortions of imaging and to provide high accuracy of generated 3D models. It starts from the basic model of image formation – the collinearity equation, expressing the condition that the point of the scene G, the center of the projection of O and the image of this point g lie on one straight line:

$$\mathbf{X}_G = \mathbf{X}_0 - \mu \mathbf{A}^T \cdot (\mathbf{x}_g - \mathbf{x}_p) \quad (1)$$

Here
$\mathbf{X}_0 = (X_0, Y_0, Z_0)$ – coordinates of the center of the projection,
$\mathbf{X}_G = (X, Y, Z)$ – scene point coordinates,
$\mathbf{x}_g = (x, y, -f)$ – the corresponding coordinates of the scene point in the image,
$\mathbf{A}$ – coordinate system transformation matrix,
$\mathbf{x}_p$ – coordinates of the main point of the snapshot,
μ – scale factor.

In a real imaging system, an image is formed with the distortions introduced by the elements of the optical

ISBN 978-80-86943-37-4

system of the lens and inaccuracies in the manufacture of the camera. To account for distortions, additional terms are introduced into the collinearity equations Δx and Δy, describing various distorting factors. Then the real (distorted) x_d, y_d coordinates of a point in the image are defined as:

$$x_d = x + \Delta x; \quad (2)$$
$$y_d = y + \Delta y; \quad (3)$$

The practice of photogrammetric measurements have proved the good description for nonlinear distortion is the following model [Bey92]:

$$\Delta x = a_0 \cdot y_a + x_a(a_1 r^2 + a_2 r^4 + a_3 r^6) + a_4(r^2 + 2x_a^2) + 2a_5 x_a y_a; \quad (4)$$

$$\Delta y = a_0 \cdot x_a + y_a(a_1 r^2 + a_2 r^4 + a_3 r^6) + a_5(r^2 + 2y_a^2) + 2a_4 x_a y_a; \quad (5)$$

here $r^2 = x_a^2 + y_a^2$

Here
x_a, y_a – coordinates of a point on the image,
$a_0, ..., a_5$ – camera interior orientation parameters:
a_0 – coefficient of affine distortion;
a_1, a_2, a_3 – coefficients of radial distortion;
a_4, a_5 – coefficients of tangential distortion.

Firstly laboratory calibration was performed using a precise test field. The test field contains 49 reference points with known 3D coordinates. The reference points are marked by coded targets providing automated detection, identification and precise sub-pixel measurement of reference points in the image. The vector of estimated parameters $v_e^l = (x_p, y_p, m_x, m_y, a_0, ..., a_5)^T$ for test field calibration includes coordinates of principal point, image scales and additional parameters correspondingly, spatial coordinates of reference points being known by independent precise measurements.

The results of calibration procedure are presented in Table 1.

Camera rmse, mm			
	Left	Right	Color
σ_x	0.022	0.024	0.014
σ_y	0.023	0.021	0.011
σ_z	0.033	0.035	0.020
Max error, mm			
δ	0.083	0.078	0.044

Table 1: Results of system calibration

Table 1 ensures the accuracy of the generated 3D models at the level of 0.05 mm. This accuracy is sufficient for the task of anthropological analysis.

The textured digital model provides the expert with more information, as some features can only be found on the color image of the object. Since the photogrammetric system has been calibrated for all three cameras using a single calibration field, this ensures that the texture is accurately superimposed on the geometric coordinates of the digital model.

Figure 3: Object image

Figure 4: Object textured 3D model

Figure 3 presents an image of a skull. Figure 4 presents textured 3D model acquired with high resolution ("micro" configuration of the photogrammetric system).

These models can be used for paleoanthropological research and solving such problems as:

ISBN 978-80-86943-37-4

- measurements of morphometric anthropological parameters (distances, angles, coordinates));
- virtual and plastic reconstruction of the external appearance;
- restoration of paleoanthropological objects-digital and manual;
- creation and documentation of specialized databases of paleoanthropological objects.

As an alternative to the traditional method, a set of algorithms for automatic detection and recognition of landmarks is proposed. They use both an object digital model and high-resolution textures to recognize a set of necessary landmarks.

3 COMPUTER TOMOGRAPHY

The study of the paleontological object was carried out on a modern multispiral computed tomography. The data obtained in DICOM [DICOM] format (433 slices with 0.572 mm step and 0.5723 mm X and Y resolution) were transmitted over the network to ILIT RAS for processing and conversion into a three-dimensional digital model.

Tomograms are a set of gray images of arrays of sections (layers) of the object under study at the z coordinate. Each element of the tomogram is a function of the density of the object at the corresponding point $q(x,y,z)$. To convert the tomogram into a digital model, it is necessary to build a mathematical model of the object as a solid. In general, such a model can be represented as:

$$F(x,y,z) = \begin{cases} 0 & : \textit{no object} \\ 1 & \textit{object present} \end{cases} \quad (6)$$

the function $F(x_i,y_j,z_k)$ has the form of a three-dimensional image with two gradations (0 or 1 - bits per pixel). Thus, if you represent an image pixel as a cube with dimensions $dx \times dy \times dz$, the transformation of tomographic data to a digital model can be performed by converting $Q(i,j,k)$ to $F(x_i,y_j,z_k)$.

For the conversion to be correct, it is necessary to correctly determine the boundary of the real object in the tomogram. In the simplest case, the ratio (1) takes the form

$$F(x_i,y_j,z_k) = \begin{cases} 0 & \text{if } Q(i,j,k) \le Q_t \\ 1 & Q(i,j,k) > Q_t \end{cases} \quad (7)$$

The correct definition of the object boundary is possible only if you know all the nuances of the mechanism of tomographic scanning of specific types of objects and representation of their images on the tomogram. After defining the boundaries of the object, the accuracy of the manufactured model is completely determined by the number of image points and the number of layers in the source data.

Tomographic data of the anthropological sample was processed in the programs Inobitec DICOM Viewer (Inobitec, Russia). Comparison of computer models is shown in (Figure 5).

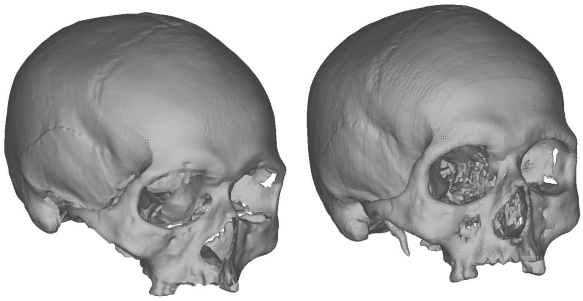


Figure 5: Left to right: anthropological model creating by photogrammetry and tomographic 3D model by programs Inobitec.

4 TEXTURED THREE-DIMENSIONAL MODEL WITH INTERNAL STRUCTURE

The main objective of this work is to create a high-precision three-dimensional textured model of paleoanthropological sample. To do this, a combination of geometric data of the photogrammetric and tomographic models was carried out. Step-by-step combination of shift and rotation was carried out according to STL data in the program Magics (Materialise, Belgium). The results of this alignment and standard deviations are shown in Figure 6.

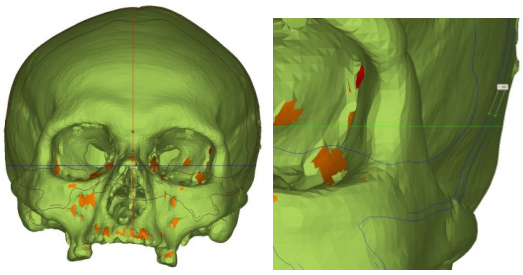


Figure 6: Step-by-step combination of STL files in the program Magics

The average deviation of the photogrammetric and tomographic models was 1.6 mm.

In addition, the models were automatically combined in a specialized program for working with point data (Figure 7).

The obtained results showed a good coincidence of the geometric data created by photogrammetric and tomographic methods. This allows you to combine textured

ISBN 978-80-86943-37-4

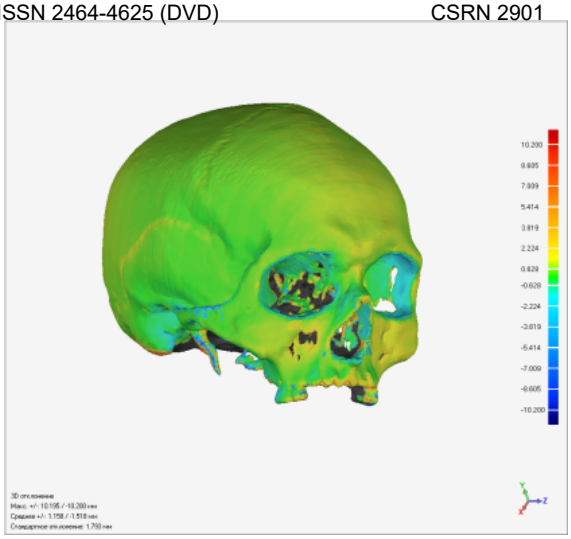


Figure 7: Automatic combination of photogrammetric and tomographic models by minimum mean deviation

geometrically bound data with the internal structure of the tomographic data and obtain a highly informative digital model (Figure 8).

Figure 8: Combination of photogrammetric and tomographic models with high-resolution textures

5 EXPERIMENTAL RESULTS. RECONSTRUCTION WITH THE USE OF ADDITIVE TECHNOLOGIES

Another advantage of the introduction of digital models in the practice of paleoanthropological research is the possibility of obtaining their exact copies with the help of additive technologies. These copies of paleoanthropological objects can be used for artistic reconstruction of appearance, for creation of reference collections for wide access.

The digital model created by the photogrammetric system represents only the external surfaces of the object can not be directly used for reconstruction on 3D printers that use only solid-state models. The 3D surface model was converted into a solid model by setting an equidistant thickness in the Magics software. This process required additional time and the resulting internal structure do not correspond to the real object.In the case of a digital model obtained by converting tomographic data, the entire structure of the object is reproduced in full, but information about the texture and appearance of the object is lost.

Plastic copies of paleoanthropological digital models were made on a laser stereolithograph LC250 [Che15], developed in ILIT RAS (Figure 9).

Figure 9: The result of the reconstruction of the paleontological object

The main purpose of creating real copies of digital models is to check the quality and accuracy of 3D printing and to study the possibility of using such copies in paleoanthropological studies. To assess the accuracy of the stereolithographic model it was scanned photogrammetric system. The resulting digital model was compared with the original model. The results of comparison of these two models are shown in (Figure 10).

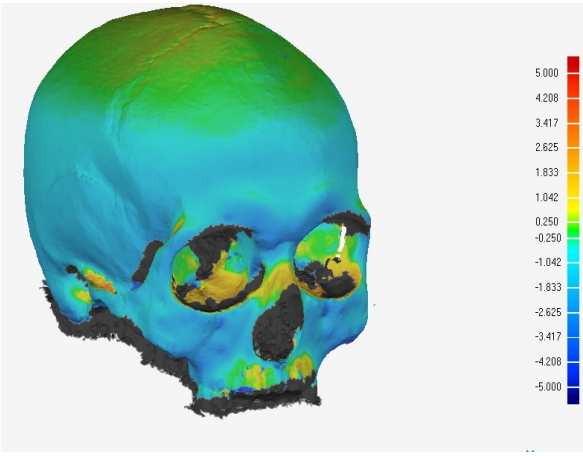


Figure10: The comparison results for two models

The results of estimates of the accuracy of stereolithography reconstruction show that SLA models can be used to solve such anthropological problems as reconstruction of human appearance and object restoration.

CONCLUSION

The problem of anthropological research and accurate documentation of the studied objects is associated with

ISBN 978-80-86943-37-4

the complexity of their geometric shape, including the presence of hidden from view areas, as well as the fragility of high – value paleoanthropological materials. In this regard, the possibility of measurements by contact methods and their accuracy are significantly limited. The methods and means of non-contact mea-surements developing today create high-precision dig-ital models of the studied objects, to provide measure-ments with high accuracy, as well as to create conditions for the preservation of the objects of research, which is of great importance when working with paleoanthropological materials. Methods of creating digital three-dimensional models of complex spatial form provide opportunities for automation of measurement processes and application of data mining methods. The resulting digital models are the basis for creating exact copies of unique objects of cultural heritage with the help of additive technologies necessary for visualization and training.

Photogrammetric methods of collection, processing and presentation of paleoanthropological data have been de-veloped. The main means of reconstruction of digital paleoanthropological 3D models is a photogrammetric system that provides accurate and photorealistic 3D and 2D data.

A study of the same anthropological object was carried out using computed tomography. A digital model based on tomographic data was constructed and compared with a photogrammetric model. The resulting difference was due to the calibration of the photogrammetric system can be compensated by a scale factor.

The applicability of the use of stereolithographic copies of paleoanthropological objects for research and exchange of rare data is studied. Evaluation of the accuracy of the stereolithographic copies shows that they can be successfully used for educational purposes and for the reconstruction of the appearance.

6 CONFLICT OF INTEREST

The authors declare that they have no conflict of interest.

7 ACKNOWLEDGMENTS

This work was done with the financial support of the RFBR grant OFI-m N 17-29-04509.

8 REFERENCES

[Bey92] Beyer, H., Advances in characterization and calibration of digital imaging systems. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. XXIX, pp. 545–555, 1992.

[Che15] Cherebylo S.A., Evseev A.V., Ippolitov E.V., Novikov M.M., Applying laser stereolithography in medicine. At: Modern laser information technology. Ed.: V. Ya. Panchenko, F.V.Lebedeva, M.: Interkontakt Nauka, 358-373б 2015.

[Cun14] John A. Cunningham, Imran A. Rahman, Stephan Lautenschlager, Emily J. Ray- field, and Philip C.J. Donoghue. A virtual world of paleontology. Trends in Ecology & Evolution, 29(6):347 – 357, 2014.

[Dav17] Davies T.G. et al. Open data and dig-ital morphology, Proc. R. Soc. B 284:20170194 http://dx.doi.org/10.1098/rspb.2017.0194, 2017

[DICOM] Digital Imaging and Communications in Medicine (DICOM), National Electrical Manufacturers Association, Rosslyn, USA, https://www.dicomstandard.org/current/.

[Heg16] Hegna Thomas A., Johnson Robert E. , Preparation of fossil and osteological 3d-printable models from freely available CT-scan movies, Journal of Paleontological Techniques, Number 16, pp. 1–10, 2016.

[Jur16a] Mikolas Jurda and Petra Urbanova. Three-dimensional documentation of dolni vestonice skeletal remains: can photogrammetry substitute laser scanning? Anthropologie vol. 54, no. 2, pp. 109 –118, 2016.

[Jur16b] Mikolas Jurda and Petra Urbanova. Sex and ancestry assessment of Brazilian crania using semi-automatic mesh processing tools Legal Medicine 23, 34 – 43, 2016.

[Kny10] Knyaz V., "Multi-media Projector – Single Camera Photogrammetric System For Fast 3D Reconstruction," *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* **XXXVIII-5**, 343-348 (2010)

[Kny16] Knyaz, V. A. and Chibunichev, A. G., "Photogrammetric techniques for road surface analysis," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **XLI-B5**, 515–520 (2016).

[Mak80] Maksimenkov G.A. Burial ground Chernovaya VIII - reference monument of Okunev culture // Monuments Okunev culture. L., 1980. pp. 3-22

[Nys15] Nysjö,J., Malmberg,F., Sintorn,I., Nyström,I.: BoneSplit - A 3D Texture Painting Tool For Interactive Bone Separation in CT Images, Journal of WSCG Vol.23, No.2, p.157, (2015).

[Rav10] Galeev R.M. Craniotriganometric research of turns from a burial ground Chernovaya VIII, Bulletin of archeology, anthropology and ethnography. 2010. № 2 (13) 109.

[Sub02] Subsol Gerard et al. Three-Dimensional Imaging in Paleoanthropology and Prehistoric Archaeology, pages 37 – 45. BAR International Series 1049, 2002.

[Web15] Gerhard W. Weber. Virtual Anthropology and Biomechanics, pages 937 – 968. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

Deep Learning-based Overlapping-Pig Separation by Balancing Accuracy and Execution Time

Hanhaesol Lee
Dept. of Computer Convergence Software
Korea University
Korea 30019, Sejong
maxsoribada@korea.ac.kr

Jaewon Sa
Dept. of Computer Convergence Software
Korea University
Korea 30019, Sejong
sjwon92@korea.ac.kr

Yongwha Chung
Dept. of Computer Convergence Software
Korea University
Korea 30019, Sejong
ychungy@korea.ac.kr

Daihee Park
Dept. of Computer Convergence Software
Korea University
Korea 30019, Sejong
dhpark@korea.ac.kr

Hakjae Kim
Class Act, Co., Ltd.
Digital-ro, Geumcheon-gu
Korea 08589, Seoul
krunivs@gmail.com

ABSTRACT

The crowded environment of a pig farm is highly vulnerable to the spread of infectious diseases such as foot-and-mouth disease, and studies have been conducted to automatically analyze behavior of pigs in a crowded pig farm through a video surveillance system using a top-view camera. Although it is required to correctly separate overlapping-pigs for tracking each individual pigs, extracting the boundaries of each pig fast and accurately is a challenging issue due to the complicated occlusion patterns such as X shape and T shape. In this study, we propose a fast and accurate method to separate overlapping-pigs not only by exploiting the advantage (i.e., one of the *fast* deep learning-based object detectors) of You Only Look Once, YOLO, but also by overcoming the disadvantage (i.e., the *axis aligned* bounding box-based object detector) of YOLO with the test-time data augmentation of rotation. Experimental results with the occlusion patterns between the overlapping-pigs show that the proposed method can provide better accuracy and faster processing speed than one of the state-of-the-art deep learning-based segmentation techniques such as Mask R-CNN (i.e., the performance improvement over Mask R-CNN was about 11 times, in terms of the accuracy/processing speed performance metrics).

Keywords

Pig monitoring, overlapping-pigs, separation, deep learning, YOLO.

1. INTRODUCTION

The early detection of management problems related to health and welfare is an important aspect of caring for group-housed livestock. In particular, caring for individual animals is necessary to minimize the possible damage caused by infectious diseases or other health and welfare problems. However, it is impossible to carefully manage pigs because each farm worker in Korea has to manage 1,800 pigs on the average. In order to solve these problems, various researches have been reported to automatically manage the behavior of individual pigs using surveillance cameras [1-5].

For surveillance systems using camera, separating dense pigs into individual pigs in a crowded environment is an essential element in the behavioral analysis of individual pigs. Recently, a faster deep learning-based object detector, such as You Only Look Once (YOLO) [6], has been developed, and thus method of the results of pig detection by using YOLO have been reported. Furthermore, separating of touching-pigs into individual pigs in crowded environments have been reported by using YOLO [7, 8]. Since YOLO has some limitation in detecting small objects, however, separating overlapping-pigs with YOLO is challenging and has not been reported yet. In complicated occlusion patterns such as X shape and T shape, for example, the small areas of the occluded pig may not be detected accurately with YOLO only.

In this study, we propose a separation method for overlapping-pigs by considering accuracy and

execution time. Our proposed method consists of the foreground detection step as the preprocessing and separation step for the overlapping-pigs. In the first step as the preprocessing, we consider data fusion between the infrared and depth information which is obtained from an Intel RealSense camera [9]. Because the infrared information is more accurate than the depth information, the infrared information can compensate for the defect of the depth information such as inaccurate pixels. Also, the depth information is less affected by illumination conditions, so that it can be complementary with the infrared information for precisely detecting pigs. Second, simple and effective image processing techniques are exploited for fast detection of pigs to satisfy the real-time execution.

In the separation of overlapping-pigs steps, we propose a detection and separation method of overlapping-pigs using YOLO and image processing techniques. First, the test image is augmented by rotating it at various angles such that the YOLO boundary box contains the occluding pig only, regardless of the occlusion pattern of the test image. In particular, we use a pre-computed lookup table for the rotation in real-time. Then, by performing YOLO on the augmented data including the input data, a bounding box for each rotation is obtained, and the optimal angle of the rotation is determined. Finally, we separate the overlapping-pigs by using the YOLO result and some image processing techniques.

The rest of the paper is structured as follows: we describe background in Section 2, and Section 3 explains the proposed method for detecting pigs from the various factors in the pig room and for separating overlapping-pigs from the various occlusion shapes. Section 4 shows the experimental results of the proposed method, and the paper is finally concluded in Section 5.

2. BACKGROUND

Recently, a variety of methods have been proposed for separating touching objects using images acquired from a camera. As an example, a separation method for two touching pigs has been proposed by analyzing the outline of touching pigs into two-dimensional image data as one-dimensional time series data [10]. By searching for two concave points between touching pigs and connecting the corresponding concave points, the touching pigs could be separated as individual pigs. However, there is a problem where the pigs cannot be separated as individual ones with a conventional separation method when the pigs are overlapped between them. In the case of the overlapping pigs, for example, the shape of the overlapping pigs appears to complex shapes such as X shape and T shape. Because more than two concave points may be obtained in such a complex overlapping shapes, accurate concave points cannot be obtained which are used to separate the overlapping pigs.

In another studies, a separation method for overlapping people or vehicles in surveillance camera environment has been proposed by using shape information and color information of the objects. In the case of overlapping among people, for example, the overlapping people can be separated as the individuals by using the outer shape of their head [11,12].

In the case of vehicle overlapping problem, some studies have been reported on separating overlapping situations between vehicles using associative tracking method using spatio-temporal information through color information [13]. However, it is difficult to apply this method for the overlapping situations between pigs because this method uses the shape information and the color information of the vehicles and people. For example, the shape information of the head and body of a pig cannot be applied to separate the overlapping pigs because the pig's head and body are not clearly distinguished. Since the color of the pigs is very similar, moreover, the separation method using color information is difficult to be simply exploited for the occlusion situations.

Meanwhile, a separation method using the depth information was proposed by analyzing differences the depth values among the pigs [14-16]. Because of inaccurate depth information, however, the previous methods have a difficulty where the threshold for separating the overlapping pigs should be set continuously according to their posture or position. In this paper, we propose a separation method for overlapping pigs by using a deep-learning technique and some simple image processing techniques.

3. PROPOSED METHOD

In order to separate the overlapping pigs, we perform the foreground detection from the pig room and data augmentation with YOLO in real-time. First, we perform the foreground detection in the pig room by using data fusion of both infrared and depth information before separating the overlapping-pigs. Initially, the region of interest (ROI) in both the input infrared and depth frames is set to only concentrate the activity region of pigs. Then, spatiotemporal interpolation technique is applied to the depth image for removing noises generated from inaccurate depth information. In the next step, the background (e.g., floor and wall) is subtracted from the foreground (i.e., the pigs) by analyzing the depth information, and the contrast of the infrared information is improved with a contrast enhancement technique to roughly localize the pigs. After that, simple image processing techniques are applied to precisely detect the pigs from data fusion (i.e., integrating infrared and depth

 ISBN 978-80-86943-37-4

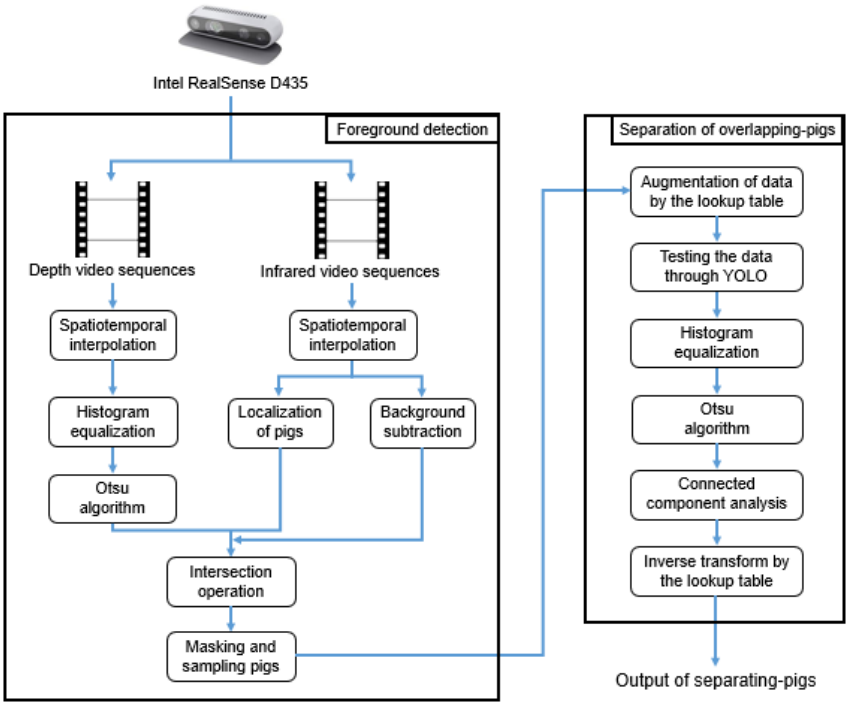


Figure 1. The overall structure of the proposed method.

information). Using the advantages of the data fusion, the pigs can be detected effectively in the low-contrast.

Second, we use YOLO to separate the overlapping-pigs from the result of the foreground detection. Firstly, the input data sampled from the result of the foreground detection is rotated and augmented by the lookup table. In fact, a trigonometric function can be used to rotate the sampled data; however, it is time-consuming to simply utilize the function when rotating the sampled data. By using the pre-computed lookup table for rotation angle, thus, the input data can be rotated and augmented in real-time. Then, YOLO can be used to detect the overlapping-pigs from a bunch of the rotated and augmented data, and some image processing techniques are applied to separate the overlapping-pigs from the YOLO results. Finally, the lookup table can be exploited to inversely transform the result of separated pigs into its original angle. Figure 1 shows the overall structure of the proposed method.

3.1 Preprocessing

First of all, we localize the pigs from the depth information. From an input depth frame, we set the ROI to only focus on the necessary regions from the whole pig room. As preprocessing step, spatiotemporal interpolation [17] with 4×4 window is then applied for removing noises (i.e., undefined pixels). Note that the spatiotemporal interpolation technique should be iteratively performed until the undefined pixels are removed. In order to realize the locations of each pig in the depth frame, the frequencies of the depth values from both background and foreground are calculated from histogram analysis. We can confirm that the background region is larger than that of the foreground (i.e., the pigs) in the pig room. In other words, the most frequent depth value can be defined as a threshold for background subtraction. Then, the pigs in the pig room can be roughly localized by using the defined threshold. However, some part of the background is not subtracted which has the same depth values with the pigs. For instance, the wall in the pig room may remain through the defined threshold because the height of the wall may be similar to the height of the pig.

In order to remove the remaining wall, the background depth map is modeled from the depth video sequences recorded during 24-h period. First, the floor and other regions (i.e., the wall and pigs) for every frame are respectively divided by using the threshold that is defined by histogram analysis. Here, we define the floor and the other parts as the floor depth map and the other depth map, respectively. Then, the floor depth map and the other depth map are independently updated with the depth values of the floor and the other parts during the 24 h videos. After updating each depth map, the depth values of the other depth map are overlapped to the not updated regions of the floor depth map because the depth values of the floor are only updated from the interpolated depth frame through the threshold.

After the background depth map is completely modeled, frame difference is performed between the background depth map and every interpolated depth frame. Then, the localized pigs can be obtained by using histogram equalization (HE) [18] and Otsu

ISBN 978-80-86943-37-4

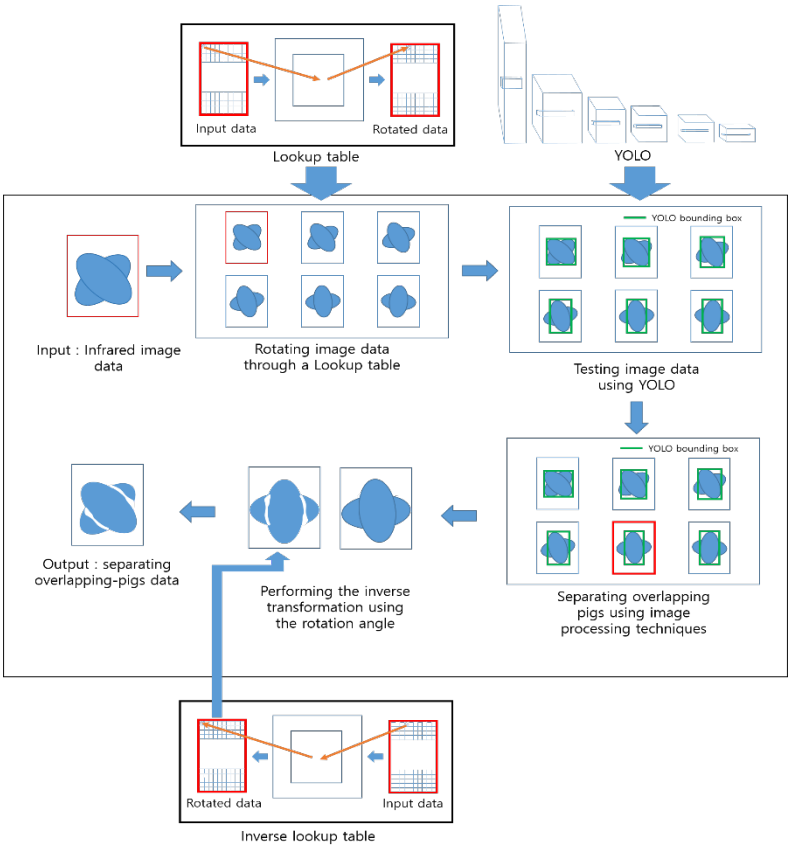


Figure 2. Outline of the separation method.

algorithm [19] on the depth frame. Finally, the depth frame with the localized pigs is used into the infrared frame for robustly detecting the pigs in the low-contrast conditions.

The pigs from infrared information can be detected by analyzing its accurate gray values. Despite of the accurate values of the infrared information, there is a problem that the pigs cannot be detected according to various illumination conditions such as the low-contrast. Because the infrared information may be affected by various illuminations, the depth frame with the localized pigs can be utilized for accurately detecting the pigs regardless of the illumination conditions. First, the ROI of the input infrared frame is also set as the same manner of the depth frame. Then, the histogram equalization (HE) is conducted to solve the low-contrast conditions. In the next step, the Otsu algorithm is applied to roughly localize the pigs after conducting HE into the infrared frame.

Because the contrast of the floor and wall in the infrared frame is also coordinated consistently by applying HE, however, the localized pigs cannot be identified between the floor and wall. Therefore, the pigs localized from each information can be detected with data fusion between the infrared and depth information.

In order to detect the pigs in the pig room, the intersection operation is performed between the infrared and depth frame where the pigs are localized. Even if the pigs can be detected from both the data, however, the background in the pig room are still detected. For example, the wall and floor are still detected as the foreground because of roughly localized frame with some remaining noises.

In an attempt to only detect the pigs, the frame difference between the input depth frame and the background depth map is used. Only the pigs can be then detected from the intersection operation between the roughly localized frame and the depth frame where the wall and floor are removed through the frame difference. Given the detected frame, the post-processing using some image processing techniques is performed to accurately detect the pigs. In order to remove the remaining noise, an erosion operation is performed to remove and minimize small noises that are adjacent to the objects or generated from the intersection operation. Then, the connected component analysis (CCA) is applied into the detected frame for labeling all of the objects, where the small labeled objects are removed. After removing the noises, the pigs can be detected by using a dilation operation to recover the shapes of the pigs.

3.2 Separation of Overlapping-Pigs

The outline of the separation method for overlapping-pigs consists of four steps as shown in Figure 2: the input data augmentation, the bounding

ISBN 978-80-86943-37-4

box selection, the overlapping-pigs separation, and the inverse transform to its original angle.

We use YOLO for separating the overlapping-pigs. YOLO is an object detector that detects objects in real time and separates them with high accuracy. However, since YOLO is a bounding box-based object detector, it has a characteristic to create a bounding box parallel to the x-axis. This characteristic causes some problems when separating the overlapping-pigs because of the shapes of the various occlusion. As shown in Figure 3, for example, a bounding box is generated on the overlapping-pigs through YOLO for X shape and T shape. In this type of occlusion, we tried to detect only occluding pig, but the area of the interest of the bounding box also include occluded pig. To solve this problem, we augment the data at the test-time. There are methods such as shifting, flipping, and rotation to augment data. Among such the augmentation categories, we use data rotation from the categories to overcome the characteristic of the bounding box.

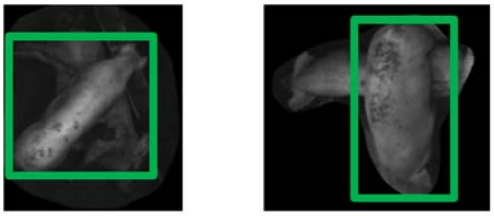


Figure 3. Various bounding boxes obtained from YOLO in case of occlusion.

The image can be rotated through the trigonometric function. In this case, however, it is hard to satisfy the real time because of a relatively large amount of computing loads. In order to satisfy the real-time processing, we rotate the data using a pre-defined lookup table. The lookup table rotates the data clockwise. The rotation angle is from 0 ° to 50 °, and it generates six data including the input data (0°). The six images are combined into one image in order to perform YOLO efficiently.

In the next step, the optimally-rotated YOLO result is obtained according to the area of the margin. Here, a "margin" is defined as the area excluding the occluding pig in the bounding box. The area of the margin can be calculated using the following equation (1):

$$margin = \frac{P_{total} - P_{occluding\_pig}}{P_{total}} \quad (1)$$

where P_{total} means the number of all pixels in the bounding box, and $P_{occluding\_pig}$ means the number of occluding pig pixels. Margins are calculated on all the augmented data to obtain the area of the margin for each of the six data including the input data. The area of the six margins obtained are compared and the bounding box of the largest area of margin is selected as the optimally-rotated YOLO result. That is, the data, which is the area of the largest margin, can be determined as the representative data among the six augment data.

Then, we apply some image processing techniques to the optimally-rotated YOLO result in order to separate the overlapping-pigs. First, histogram equalization (HE) technique [18] is applied to the optimally-rotated YOLO result in order to clearly distinguish the boundary line between the overlapping pigs. When the histogram smoothing is applied, the intensity distribution of the whole image using the smoothing becomes uniform, and thus the contrast between the two pigs can be improved. As a next step, Otsu technique [19] is applied in order to separate the two pigs. Then, the binary result from Otsu is labeled with connected component analysis technique [20], and the number of foreground pixels is calculated to derive the size of each connected component. By selecting the largest and the second largest connected component, we can obtain the occluding pig and the occluded pig as a final result of image processing techniques (i.e., some image noise can be removed).

Finally, we use an inverse lookup table to rotate the image processing result back to its original angle. That is, the inverse lookup table is a pre-computed table that rotates the pixel locations of the image processing result in the opposite direction with the lookup table. As a result, we obtain the final separation result of the overlapping-pigs. Algorithm 1 describes the overall proposed method.

Algorithm 1 Separation of overlapping-pigs

Input: Infrared and depth information sequences

I_{mask} = FG_Detect($I_{infrared}, I_{depth}, I_{bg}$);
I_{sample} = SamplingPigs($I_{mask}, I_{infrared}$);

for i = 1 to 5 :
  $I^{i}_{rotate} = lookup_i(I_{sample})$;

$I_{optimal}$ = CompareMargines(I_{rotate});
I_{bbox} = YOLO($I_{optimal}$);
Apply histogram equalization into I_{bbox};
Separate overlapping pigs in I_{bbox} with Otsu;
Post-process I_{bbox} through CCA;
$I_{sep.} = lookup^{inverse}(I_{bbox})$;

Output: Separating overlapping pigs into individual pigs

ISBN 978-80-86943-37-4

4. EXPERIMENTAL RESULTS

The following experimental setup was used to conduct the proposed method: Intel ® Core™ i7-7700K, NVIDIA GeForce GTX1080 Ti, 32GB RAM, and OpenCV 3.4. For the experiment, an Intel RealSense D435 camera was installed on the ceiling of 3.2m above the bottom. There were 9 pigs in the room, and we obtained video frame images having resolution size of 1280×720 pixels. In the case of YOLO, we produced a model through the training data, which was composed of 774 frames. The YOLO learning parameters were 0.001 for learning rate, 0.9 for momentum, and 0.0005 for decay. The activation function was leaky ReLU function. We then used 47 test frames that were not used as training data.

We used the depth and infrared video sequences obtained from Intel RealSense camera during a 24 h period. As explained in Section 2, various illumination conditions in the infrared and depth videos were confirmed such as low contrast. In particular, the low-contrast conditions were evidently identified when the pigs were located at the corners in the pen. Thus, we detected the pigs while considering the conditions.

Initially, we modeled the background depth map as an independent procedure for conducting the frame difference into the input depth frame. Then, the spatiotemporal interpolation technique was applied to the 1296 frames extracted from each video. Note that because the spatiotemporal interpolation technique was interpolated from three frames to one frame, 1296 frames were needed to detect the pigs in 432 frames. With each interpolated frame, simple image processing techniques were conducted to each domain.

In the case of depth information, a histogram analysis was performed for background subtraction in the input depth frame. The frequency of the depth value corresponding to the background converged to 53, and the depth threshold for segmenting the background was defined as 53. The background-subtracted depth frame was then derived using the threshold defined through histogram analysis. In addition, the frame difference between the background depth map and the interpolated depth frame was carried out. In the case of the procedure of infrared information, the Otsu algorithm was used to define the parameter for segmenting the background for roughly localizing the pigs. Using the background-subtracted depth frame and each localized frame from each information, the pigs could be detected by performing the intersection operations among the frame attributes.

For the detected frame, the morphology operation and CCA were conducted as the post-processing steps for refining the detected pigs. As the size of each noise calculated by CCA was less than 100, the noises were simply removed with the threshold defined as 100. After that, a dilation operation was conducted three times to sufficiently recover the shape of the pigs, and as a result, all of the pigs in the pen could be accurately detected.

Then, we sampled the input data from the results of the foreground detection, which was used to rotate counterclockwise for selecting an optimal bounding box. The optimal bounding box was selected when the occluding pig was aligned with x-axis or y-axis. Accordingly, the rotation angle of the test-time augmentation was set to 0 ° ~ 50 ° with 10 ° interval. Then, the six rotated data were merged into a single synthetic image. Through this synthesis, six YOLO executions could be reduced to one YOLO execution (i.e., YOLO is You Only Look Once, independent of the number of objects to be detected). In addition, we used a pre-computed lookup table, instead of applying a trigonometric function directly, to reduce the execution time. In the case of rotating one data using the trigonometric function, the execution time of 27 msec was required. On the other hand, the lookup table could reduce the rotation time to less than 1msec.

Fig. 4 shows the YOLO results after rotating 0 °, 10 °, 20 °, 30 °, 40 °, and 50 ° for X-shaped and T-shaped overlapping pigs, respectively.

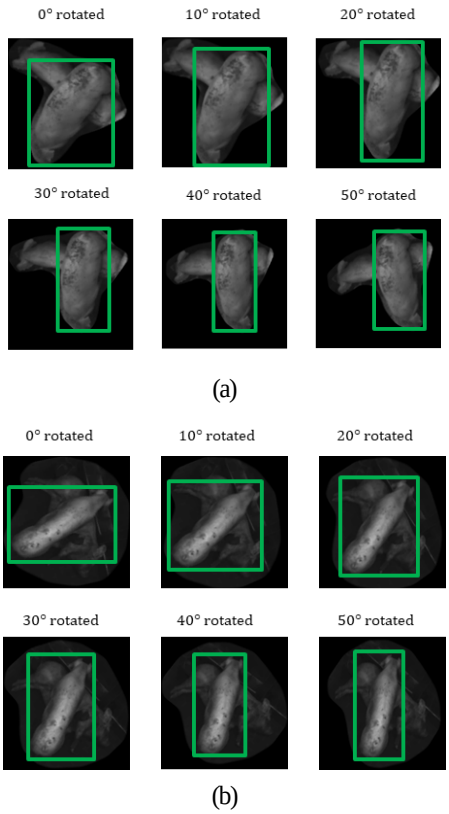


Figure 4. Results of YOLO from the rotated data. (a) Results of YOLO from the rotated T-shaped data and (b) Results of YOLO from the rotated X-shaped data.

ISBN 978-80-86943-37-4

Then, we measured the horizontal and vertical lengths of the six bounding boxes, and obtained the horizontal and vertical ratios of the bounding boxes. The YOLO result with the largest ratio was selected as the optimally-rotated YOLO result. In Figure 4(b), for example, the ratios of each rotation 0 ° rotation) were 1.05 (for 0 ° rotation), 1.29 (for 10 ° rotation), 1.62 (for 20 ° rotation), 1.80 (for 30 ° rotation), 2.05 (for 40 ° rotation), and 2.27 (for 50 ° rotation), respectively. Then, the largest ratio was 2.27, and thus the YOLO result rotated by 50 ° rotation was selected as the optimally-rotated YOLO result.

With the optimally-rotated YOLO result, we applied some image processing techniques to separate the overlapping pigs. Histogram smoothing was first applied to the optimally-rotated YOLO result in order to clarify the intensity value of the overlapping-pigs by improving the contrast of the optimally-rotated YOLO result. Then, Otsu was applied to separate the individual pigs from the overlapping-pigs. After the connected component analysis with the Otsu's result, we computed the size of each connected component. Finally, we selected two largest connected components as the occluding pig and the occluded pig, and rotated the result to the original angle by using the inverse lookup table.

In order to evaluate the separation result of the proposed method qualitatively, we compared it with the state-of-the-art deep learning-based instance segmenter (i.e., Mask R-CNN [21]) and YOLO [6] only (i.e., applying YOLO to overlapping-pigs directly without any rotation). As shown in Figure 5, Mask R-CNN could not distinguish the overlapping-pigs depending on the overlapping patterns, whereas YOLO only could not distinguish the overlapping-pigs at all with the selected frames. On the contrary, the proposed method could distinguish the overlapping-pigs with the selected frames.

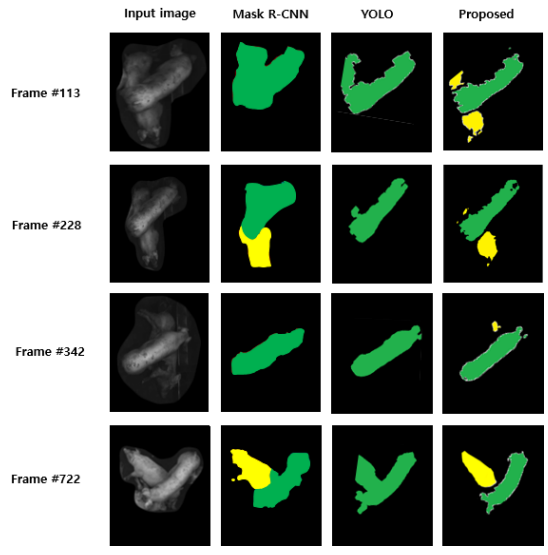


Figure 5. Separation results of overlapping-pigs.

In order to evaluate the separation performance of overlapping-pigs quantitatively, the results of the three methods were compared. As a performance metric for accuracy, we compared the separation results with ground-truth at pixel-level. Each accuracy of the methods was calculated by equation (2):

$$Accuracy = \frac{N_{TP}}{N_{TP}+N_{FP}+N_{FN}} \quad (2)$$

where true positive (N_{TP}) is defined to a pixel on the separated pigs, false positive (N_{FP}) is defined to a pixel on the background as the separated pigs, and false negative (N_{FN}) is defined to a pixel on the separated pigs as the background.

Since real-time performance is important in any surveillance system, we compared the execution times. Finally, these two performance metrics (i.e., accuracy and time) are generally conflicting, we computed a "collective" performance metric defined as accuracy/time.

Table 1 shows the comparison results of Mask R-CNN, YOLO only, and the proposed method. The proposed method could provide better accuracy than Mask R-CNN and much better accuracy than YOLO. Also, the proposed method could provide much faster execution time than Mask R-CNN, while it could provide slightly slower execution time than YOLO due to the additional image processing steps. With the accuracy/time performance metric, the proposed method could improve the collective performance of Mask R-CNN by a factor of 11, and that of YOLO by 27%.

Even though the overlapping pigs were separated through the proposed method in real-time, we should consider the occlusion issue for separating among more than two pigs. The occlusion among the numerous pigs in a complex pig room should be solved with the extended separation method in order to manage the pigs' health care. Therefore, we will extend the proposed method to precisely separate occlusion among more than two pigs.

Table 1. Performance comparison

Methods	Accuracy (%)	Time (msec)	Accuracy / Time (%/msec)
Mask R-CNN[21]	79.87	254.35	0.31
YOLO[6]	55.61	20.42	2.72
Proposed	83.33	24.01	3.47

5. CONCLUSION

Separation of dense pigs in a crowded environment is an important issue to automatically manage pig

ISBN 978-80-86943-37-4

farms. Recently, many studies have been reported to detect pigs with YOLO (i.e., one of the fast deep learning-based object detectors). With the axis aligned bounding box-based method, however, separating overlapping-pigs is difficult, depending on the complicated occlusion patterns.

In this study, we proposed the real-time separation method for overlapping-pigs using test-time augmentation with the effective foreground detection in the pig room. First of all, we applied the spatiotemporal interpolation into the depth information for removing noises and roughly localizing the pigs in the pig room. Then, the infrared information was concurrently utilized to the depth information as data fusion, and we simply applied effective image processing techniques to precisely detect the pigs in real-time.

Through the results of the foreground detection, we used YOLO and image processing techniques by using the pre-computed lookup table for test-time augmentation. As a result of these procedures, the overlapping-pigs could be separated fast and accurately regardless of the complicated occlusion patterns.

Experimental results show that the overlapping-pigs could be separated with accuracy of 83.33% and the execution speed of 24.01msec. These results show that the accuracy/time performance of the proposed method was 1,119% higher than that of Mask R-CNN, and it was 27% higher than that of YOLO only. Future studies will be carried out on a parallel-processing method that can handle the whole process of individual pig analysis in real-time.

6. ACKNOWLEDGMENTS

This research was supported by the Basic Science Research Program through the NRF funded by the MEST (2018R1D1A1A09081924) and the Leading Human Resource Training Program of Regional Neo Industry through the NRF funded by the MSIP (2016H1D5A1910730).

7. REFERENCES

[1] Y. Chung, S. Oh, J. Lee, D. Park, H. Chang, and S. Kim, “Automatic Detection and Recognition of Pig Wasting Diseases Using Sound Data in Audio Surveillance Systems,” *Sensors*, Vol. 13, No. 10, pp. 12929-12942, 2013.

[2] A. Wongsriworaphon, B. Arnonkijpanich, and S. Pathumnakul, “An Approach Based on Digital Image Analysis to Estimate the Live Weights of Pigs in Farm Environments,” *Computers and Electronics in Agriculture*, Vol. 115, No. C, pp. 26-33, 2015.

[3] M. Oczak, K. Maschat, D. Berckmans, E. Vranken, and J. Baumgartner, “Automatic Estimation of Number of Piglets in a Pen During Farrowing, Using Image Analysis,” *Biosystems Engineering*, Vol. 151, pp. 81-89, 2016.

[4] M. Ju, H. Baek, J. Sa, H. Kim, Y. Chung, and D. Park, “Real-Time Pig Segmentation for Individual Pig Monitoring in a Weaning Pig Room,” *Journal of Korea Multimedia Society*, Vol. 19, No. 2, pp. 215-223, 2016.

[5] L. Lee, L. Jin, D. Park, and Y. Chung, “Automatic Recognition of Aggressive Behavior in Pigs Using a Kinect Depth Sensor,” *Sensors*, Vol. 16, No. 5, pp. 631, 2016.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real Time Object Detection,” *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779-788, 2016.

[7] J. Seo, M. Ju, Y. Choi, J. Lee, Y. Chung, and D. Park, “Separation of Touching Pigs Using YOLO-Based Bounding Box,” *Journal of Korea Multimedia Society*, Vol. 21, No. 2, pp. 77-86, 2018.

[8] M. Ju, Y. Choi, J. Seo, J. Sa, S. Lee, Y. Chung, and D. Park, “A Kinect-based Segmentation of Touching-Pigs for Real-Time Monitoring,” *Sensors*, Vol. 18, No. 6, pp. 1746, 2018.

[9] Intel RealSense D435, Intel. Available online: https://click.intel.com/intelr-realsensetm-depth-camera-d435.html (accessed on 28 Feb. 2018).

[10] H. Baek, Y. Chung, M. Ju, Y. Chung, D. Park, and H. Kim, “Pig Segmentation using Concave Points and Edge Information,” *Journal of Korea Multimedia Society*, Vol. 19, No. 8, pp. 1361-1370, 2016.

[11] Y. Do, “Dividing Occluded Humans Based on an Artificial Neural Network for the Vision of a Surveillance Robot,” *Journal of Institute of Control, Robotics and Systems*, Vol. 15, No. 5, pp. 505-510, 2009.

[12] H. Choi, S. Hong, and J. Ko, “Merge and Split of Players under MeanShift Tracking in Baseball Videos,” *Journal of Advanced Navigation Technology*, Vol. 21, No. 1, pp. 119-125, 2017.

[13] J. Lim, S. Kim, C. Lee, and M. Lee, “Overlap Removal and Background Updating for Associative Tracking of Multiple Vehicle,” *Journal of KIISE*, Vol. 16, No. 1 pp. 90-94, 2010.

[14] H. Lee, Y. Choi, J. Sa, Y. Chung, and D. Park, “Detection of Occluding Pigs Using Depth Information in a Pigsty,” *Proceeding of the Fall Conference of the Korea Multimedia Society*, Vol. 25, No. 2, pp. 833-835, 2018.

 ISBN 978-80-86943-37-4

[15] H. Lee, H, Lee, J. Kim, Y. Choi, H. Kim, Y. Chung, D. Park, and H. Kim, "Occluded-Pigs Detection and Separation Using Depth Information," *Proceeding of the Fall Conference of the Korea Multimedia Society*, Vol. 20, No. 2, pp. 813-815, 2017.

[16] H. Seo, H. Lee, C. Park, J. Seo, Y. Chung, D. Park, and H. Kim, "Occluding Pigs Individual Detection Using Depth Information," *Proceeding of the Conference of the Workshop on Image Processing and Image Understanding*, 2018.

[17] J. Kim, Y. Chung, Y. Choi, J. Sa, H. Kim, Y. Chung, D. Park, and H. Kim, "Depth-based Detection of Standing-Pigs in Moving Noise Environments," *Sensors*, Vol. 17, No. 12, pp. 2757, 2017.

[18] M. Eramian and D. Mould, "Histogram Equalization using Neighborhood Metrics," In *Proc. of the 2nd Canadian Conference on Computer and Robot Vision (CRV'05)*, pp. 397-404, 2005.

[19] N. Otsu, "A Threshold Selection Method from Gray-level Histograms," *IEEE Trans. Syst. Man Cybern*, Vol. 9, No. 1, pp. 62–66, 1979.

[20] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov, "Real-time Computer Vision with OpenCV," *Communications of the ACM*, Vol. 55, No. 6, pp. 61-69, 2012.

[21] K. He, G. Gkoxari, P. Dollár, and R. Girshick, "Mask R-CNN," *Proceeding of the IEEE International Conference on Computer Vision*, pp. 2980-2988, 2017.

 ISBN 978-80-86943-37-4

 ISBN 978-80-86943-37-4

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

Performance Assessment of Convolutional Neural Networks for Semantic Image Segmentation

Alexander Leipnitz Tilo Strutz Oliver Jokisch
Institute of Communications Engineering
Leipzig University of Telecommunications (HfTL)
Gustav-Freytag-Str. 43-45
04277, Leipzig, Germany
{leipnitz,strutz,jokisch}@hft-leipzig.de

Abstract

Convolutional neural networks are applied successfully for image classification and object detection. Recently, they have been adopted to semantic segmentation tasks and several new network architectures have been proposed. With respect to automotive applications, the Cityscapes dataset is often used as a benchmark. It is one of the biggest datasets in this field and consists of a training, a validation, and a test set. While training and validation allow the optimisation of these nets, the test dataset can be used to evaluate their performance.
Our investigations have shown that while these networks perform well for images of the Cityscapes dataset, their segmentation quality significantly drops when applied to new data. It seems that they have limited generalisation abilities. In order to find out whether the image content itself or other image properties cause this effect, we have carried out systematic investigations with modified Cityscapes data. We have found that camera-dependent image properties like brightness, contrast, or saturation can significantly influence the segmentation quality. This papers presents the results of these tests including eight state-of-the-art CNNs. It can be concluded that the out-of-the-box usage of CNNs in real-world environments is not recommended.

Keywords

Convolutional neural network, Semantic segmentation, Generalisation abilities

1 INTRODUCTION

Recent developments of convolutional neural networks for semantic segmentation led to impressive results on validation and test datasets. However, the datasets for this performance measurement and the dataset on which the training procedure was based share the same image characteristics as, for example, the lighting conditions and the acquisition environment (e.g. camera type and settings). A very prominent and one of the largest datasets in the field of semantic segmentation is the Cityscapes dataset [1]. It shows urban scenes of 50 different cities. Common features between the training, validation and test datasets have been prevented by having no city being doubly represented in one of the sub datasets. Nevertheless, dependencies still exist between them due to the standardised capture settings. Images from real-world scenarios can be much more diverse, especially when using different cameras or settings, and state-of-the-art CNNs are expected to cope with these image variations. So far it was not known how well CNNs trained with the Cityscapes dataset perform under various lighting conditions or in rural areas.

Domain adaptation and transfer-learning are well-known methods to adjust trained models to new conditions or type of scenes. However, they are typically not used to increase the generalisation abilities but to shift the application range. The use of CNNs in real-world applications such as autonomous driving, on the other hand, requires them to function optimally under all kinds of conditions.

In this paper, eight state-of-the-art CNNs are compared using out-of-the-box models available on the Internet in order to assess their generalisation abilities. The investigations have revealed that most of the evaluated nets do not cope well with images having varying characteristics which can be caused by different camera systems. These variations have been simulated by modifying brightness, saturation, or contrast of the images. In a second test, images that do not belong to the Cityscapes dataset have been presented to the CNNs in order to visually evaluate the resulting segmentation masks.

2 RELATED WORK

The focus on a specified dataset and therefore overfitting and limited generalisation abilities of neural net-

works are a known issue and have already been covered in literature.

Adversarial examples are images that can trick a neural network into a false classification by slightly modifying an otherwise correctly classified image. In [2] these cases are described, searched and systematically provoked. Wang et al. presents a theoretical analysis of the functionality of adversarial examples and possible countermeasures [3].

Even small image transformations like object translation can drastically influence CNNs for object recognition [4], which is validated in [5]. Rosenfeld et al. additionally showed that the object position in an image and "object transplanting" from one image to another has not only an influence on its own detection rate but also the detection of other objects in the image.

Global image modifications can also have an effect on the CNN results. In [6], the robustness of classifiers against added random and *semi-random* noise in the samples has been researched and their impact on the classification rate has been proven. In addition, the effects of blur, noise, contrast, JPEG and JPEG2000 related compression artefacts are evaluated for image classification tasks for deep neural networks in [7]. Vasiljevic et al. extend this in [8] by investigating the impact of blurred images to semantic segmentation tasks.

Solutions have already been proposed to overcome these problems. In [9], the robustness of a classification deep neural networks has been improved by including distorted copies of the original images in the training process. They use downsampling, JPEG compression and random cropping for stability training. Random cropping is one form of data augmentation that is used for some CNNs compared in this paper.

An alternative solution to make CNNs robust against these type of modifications is proposed in [10]. Based on the image quality, different paths inside of the network are selected to maximize the classification result.

The impact of image modifications on the task of image classification has already been comprehensively studied in the present literature. The effects are expected to carry over to semantic segmentation tasks. This paper examines the influence of image modifications with real application background (image brightness, contrast, and saturation). Additionally, the segmentation of unknown images is evaluated. This allows a comparison of different network architectures beyond their test dataset segmentation scores.

3 INVESTIGATIONS

CNNs for image classification and object detection use a cascade of convolutional layers and downsampling to compute a vector containing the class scores from the resulting feature maps via fully connected layers. Changes to this classic CNN architecture have been made to cope with the task of image segmentation. The class scores are computed for each pixel on low resolution features maps and different upsampling techniques have been developed to obtain a score map in the original image resolution. This is known as an encoder-decoder network. The basic structure can be seen in **Fig. 1**a) with the feature maps being the possible intermediate result after many different modules or layers (convolution, downsampling or upsampling).

CNN	Test-Set. $mIoU[\%]$	Val.-Set $mIoU[\%]$	Fig. 1
DeepLabv3+	82.1	78.7	a)
PSPNet	81.2	77.0	a)
TuSimple-DUC	77.6	83.7	a)
RefineNet	73.6	75.3	b)
LRR	71.9	72.5	a)
ICNet	69.5	67.7	b)
ESPNet	60.3	59.1	a)
ENet	58.3	53.5	a)

Table 1: Ranking of the CNNs based on their official Cityscapes test dataset results (Test-Set. *mIoU*), the measured results on the validation dataset (Val.-Set. *mIoU*) and the categorization of their network architecture

Every network architecture takes a different approach on this structure with more or less drastic modifications to the encoder or decoder. A major change is the introduction of additional branches on the encoder site that process downsampled versions of the original image in parallel. This is called a multi-path encoder-decoder structure (**Fig. 1**b) and is explained in more detail in the following subsections.

3.1 Selected Convolutional Neural Networks

On the Cityscapes website [11] a lot of CNNs are ranked regarding their segmentation performance on the test dataset. The **Tab. 1** lists some of these network architectures along with their performance ratings (mean Intersection over Union metric [12], *mIoU*) for the Cityscapes test and validation dataset. This selection is based on code availability and covers a broad range of segmentation capabilities (highest, middle and also low *mIoU* scores) and objectives. DeepLabv3+, PSPNet, TuSimple-DUC, RefineNet, and LRR focus on the highest segmentation score possible, while ICNet, ESPNet and ENet also have real-time inference in mind. They all can be categorized as an encoder-decoder (Fig. 1a) or multi-path encoder-decoder (Fig. 1b) network architecture.

 ISBN 978-80-86943-37-4

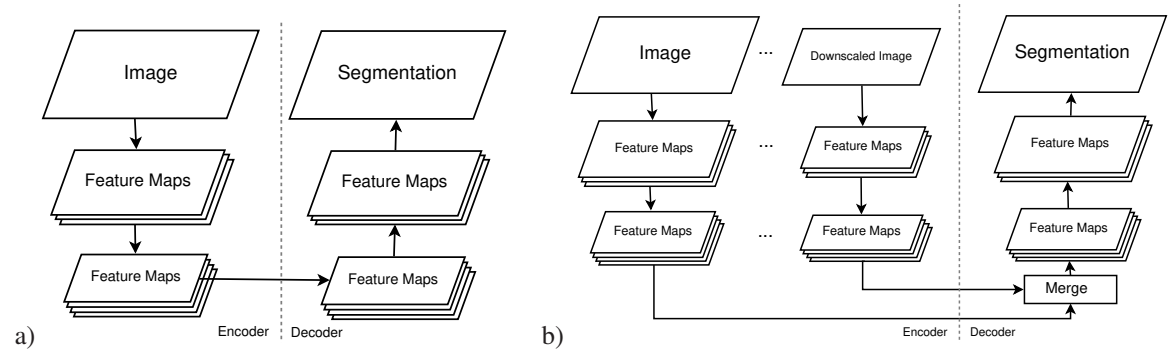


Figure 1: Basic CNN architecture for semantic segmentation; a) encoder-decoder; b) multi-path encoder-decoder

3.1.1 DeepLabv3+

The Deep Labelling Network (DeepLabv3+) [13] is the highest ranked network on the test dataset but only the second best network on the validation dataset of the examined networks (see Table 1). It is based on DeepLabv3 [14] and introduces a modified Xception module [15] as network backbone. In general, the network architecture is based on a complex and powerful encoder that relies on parallel atrous convolution with different rates to enlarge the field-of-view (Atrous Spatial Pyramid Pooling). The decoder module handles the upsampling and combines the final encoder output with low-level features from previous layers with the same spatial size to recover object segmentation details.

3.1.2 PSPNet

The Pyramid Scene Parsing Network (PSPNet) [16] is the second highest ranked network on the test dataset and third highest network on the validation dataset. Its encoder is ResNet-based [27] while the decoding is performed by a pyramid pooling module. This proposed module uses parallel pooling and convolution with different sized kernels/filters. This aims at a broader receptive field by including local and global context information. The resulting feature maps are then upsampled and concatenated before a final convolutional layer generates the segmentation-output.

3.1.3 TuSimple-DUC

The ResNet-DUC-HDC alias TuSimple-DUC [17] yields the highest segmentation performance on the Cityscapes validation dataset and third highest segmentation performance on the test dataset. It introduces a combination of Dense Upsampling Convolution (DUC) and Hybrid Dilated Convolution (HDC) as an addition to the ResNet-based architecture. The encoder consists of the ResNet and HDC layers while the decoding is performed by the DUC layers.

3.1.4 RefineNet

The Multi-Path Refinement Network (RefineNet) [18] uses parallel processing of the original and downsampled version of the input image. It is a multi-path encoder-cecoder network architecture. The RefineNet-block consists of two ResNet-based Residual Convolution Unit (RCU) for each input, Multi-resolution Fusion, Chained Residual Pooling and final RCU to compute the output feature map.

3.1.5 LRR

The LRR architecture (Laplacian Pyramid Reconstruction and Refinement) [19] introduced the two name-giving techniques. The low-resolution segmentation map is upsampled and refined with the help of higher-resolution feature maps in areas with high uncertainty.

3.1.6 ICNet

The Image Cascade Network (ICNet) [20] is a variation of the PSPNet with focus on real-time inference and is also a multi-path encoder-decoder network architecture with three encoder branches. The PSPNet architecture is only used for a downsampled version of the input image to save computational time. The resulting small spatial sized feature map gets upsampled and merged with feature maps that originated from a higher sampled and later the original sized input image. They both only have passed through a limited number of convolutional layers. After these two Cascade Feature Fusion (CFF) modules, only upsampling and a final convolutional layer is applied to get the final segmentation output.

3.1.7 ESPNet

The Efficient Spatial Pyramid of Dilated Convolutions Network (ESPNet) [21] has introduced an ESP module replacing the standard convolutional layer. It consists of a point-wise convolution and a spatial pyramid of dilated convolutions that result in an computational efficient and bigger receptive field. The network architecture consists of normal convolutional layers, ESP modules and deconvolutional layers [22] for upsampling.

3.1.8 ENet

The Efficient Neural Network (ENet) [23] is especially designed for real-time inference. For this reason, the

 ISBN 978-80-86943-37-4

network architecture is very small compared to the other presented networks by limiting the number of layers and size of the feature maps. It is based on the ResNet architecture.

3.2 Experimental Setup

The generalisation abilities of CNNs can be evaluated in two ways. Firstly, the images of a known dataset can be modified and the change in segmentation performance is measured. Secondly, the segmentation output for unknown images can be visually demonstrated and discussed. The out-of-the-box performance of each network architecture has been tested with the provided and here named models:

- DeepLabv3+: *deeplabv3_cityscapes_train*; model-variant: xception_65;
- TuSimple-DUC: *ResNet_DUC_HDC_CityScapes*
- RefineNet: *refinenet_res101_cityscapes.mat*
- LRR: *LRR4x-VGG16-CityScapes-coarse-and-fine*
- ESPNet: *espnet_p_2_q_8.pth*
- ENet: *cityscapes_weights.caffemodel*.

The official implementation of the PSPNet and ICNet could not be used due to Hard- and Software incompatibilities. Instead, the Tensorflow implementations [24, 25] have been utilized with these models:

- PSPNet: *pspnet101-cityscapes*
- ICNet: *icnet_cityscapes_train_30k.npy*.

The Cityscapes test dataset is not public available so all further tests have to be performed on the validation dataset. Changing the brightness, contrast and saturation of its images represents realistic scenarios in a real-world environment.

3.2.1 Brightness

A brightness modification can be described by an offset b to the R, G and B values of each pixel:

$$R' = \max(\min(R+b,255),0) \quad (1)$$
$$G' = \max(\min(G+b,255),0) \quad (2)$$
$$B' = \max(\min(B+b,255),0) \quad (3)$$

with b ranging between $[-50;50]$ in increments of 10 in our tests. The **Fig. 2** shows the effect of the maximum brightness changes on a Cityscapes validation dataset image. The range of b is chosen so that the resulting images still look realistic and can arise by under- or overexposing the camera sensor. It is to be expected that the influence of this modification on the segmentation results is rather low because the gradients are not affected. Only pixels that have to be clipped to 0 or 255 change their properties in a non-linear way.

3.2.2 Contrast

A contrast modification corresponds to the multiplication with a factor c:

$$R' = \max(c \cdot R,255) \quad (4)$$
$$G' = \max(c \cdot G,255) \quad (5)$$
$$B' = \max(c \cdot B,255) \quad (6)$$

with c ranging between $[0.5;2]$ in our tests. We choose the values $c \in \{0.5;0.7;1;1.4;2\}$. The influence of the maximum contrast modification can be seen in **Fig. 3**. This modification is more drastic compared to the brightness change because it is affecting the neighbouring relations between pixels and compresses or stretches the accompanying histogram.

3.2.3 Saturation

The saturation of an image can easily be modified by transforming the image from the RGB to the HSV colour-space first. According to [26], the saturation s_{hsv} is defined by:

$$s_{hsv} = \begin{cases} 0 & \text{if } R=G=B \\ 255 \cdot \frac{max(R,G,B)-min(R,G,B)}{max(R,G,B)} \end{cases} . \quad (7)$$

Analogous to the brightness, the saturation modification can be applied by an offset s:

$$s'_{hsv} = \max(\min(s_{hsv}+s,255),0) \quad (8)$$

with s ranging between $[-40;40]$ in increments of 10 in our tests. **Fig. 4** shows the maximum saturation modification. Due to the transformation, the pixel values change non-linear and the neighbourhood-relations between pixel get distorted the most. It is expected that this modification has the most influence on the segmentation performance.

4 RESULTS AND DISCUSSION

4.1 Modified Cityscapes Validation Dataset

The graphs in **Fig. 5** show the influence of the image modification on the *mIoU* score for each CNN applied to the entire Cityscapes validation dataset.

The images in the Cityscapes dataset are rather dark, which is indicated by the low mean of the colour channels for the images of the training dataset (R: 73.19, G: 82.91 B: 72.39). Therefore, the reduction of brightness can make many objects black. They become indistinguishable resulting in a drastic decrease of the segmentation performance of all nets for a negative b in Fig. 5a). Increasing the brightness does not seem to affect the CNNs output much except for the ICNet.

The influence of the contrast modification in Fig. 5b) surprisingly has the least affect on the segmentation

 ISBN 978-80-86943-37-4

Figure 2: Brightness modification on a Cityscapes validation image; a) original image; b) $b = -50$; c) $b = 50$

Figure 3: Contrast modification on a Cityscapes validation image; a) original image; b) $c = 0.5$; c) $c = 2$

Figure 4: Saturation modification on a Cityscapes validation image; a) original image; b) $s = -40$; c) $s = 40$

performance, although the images in Fig. 3 appear to be the darkest or brightest of all image modifications in their extreme points. Only a very big c leads to a significant decrease probably due to clipping of the pixel values to 255.

As expected, the change in saturation has the greatest impact on the segmentation performance. The curves in Fig. 5c) drop off rather rapidly with an increasing $|s|$. DeepLabv3+ and RefineNet seem to be the least affected by this modification as the flat curves indicate.

Some curves intersect with others. This indicates that some network architectures have a lower segmentation performance on the "default" images but a higher robustness against image modification and therefore less over-fitting. The DeepLabv3+ shows in all three graphs the best generalisation abilities. It has the second highest *mIoU* score and flatter curves compared to TuSimple-DUC which it also intersects. Therefore, the DeepLabv3+ network architecture has the best compromise between *mIoU* score and generalisation abilities in this test.

4.2 Unknown Images Dataset

In a real-world application, a CNN is exposed to various different scenes. The validation and test datasets are usually from the same source and feature the same bias (camera settings, preference by the photographer etc.). This bias is also learned by the CNNs and prevents them from having good generalisation capabilities. To test this further, a visual segmentation evaluation has been performed with unknown images from a completely different source. **Fig. 6** to **Fig. 9** (each a)) show four example images with similar content to the Cityscapes images and exclusively known objects/classes. They only tend to be a bit brighter and originate from a different camera. Even without having ground truth data for these images available, the segmentation outputs produced by the nets in b) - i) show remarkable differences that allow a subjective comparison. The relation between the colours and the classes of the Cityscapes dataset can be seen in **Fig. 10**. The conclusion from the previous section is confirmed with DeepLabv3+ producing the best looking segmentation output and showing the best generalisation performance. The segmentation is almost perfect with only the semantic meaning being wrong in some cases. The biggest problem seems to be the semantic segmentation of the grassland where the border is inexact and the classes "vegetation", "terrain" and "sidewalk" are assigned in an inconsistent way. The second best network architecture in this test appears to be the RefineNet, whose segmentation has the same but more obvious problems. The others CNNs often drastically fail to segment the objects correctly and make fundamental errors regarding the classification. The class "building" stands out by being assigned incorrectly to different areas in the images.

ESPNet and ENet have the biggest problems with the images. The segmentation of objects is mostly wrong and often the segments are classified into the wrong class. Especially Fig. 7 is negatively noticeable here.

4.3 Comparison of Results Between Both Datasets

In our tests, DeepLabv3+, first place on the test dataset but only second place on the validation dataset, seems to be the least influenced by the image modifications and showed the best segmentation output for the four unknown images.

ISBN 978-80-86943-37-4

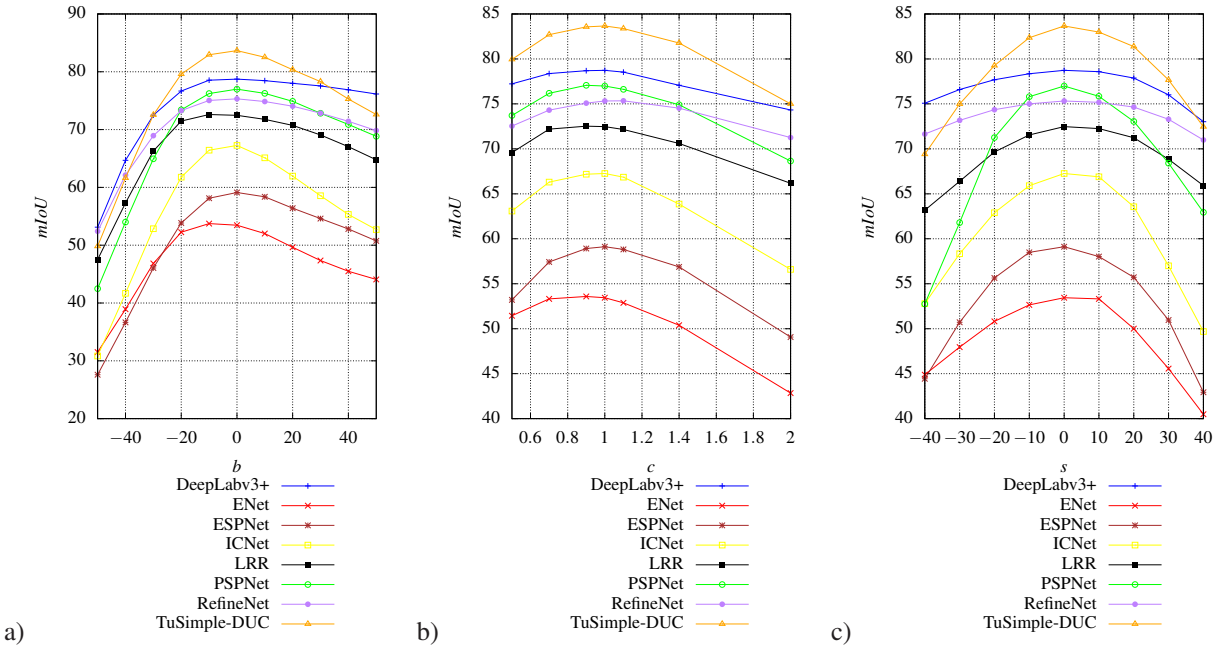


Figure 5: Influence on the *mIoU* score a) brightness; b) contrast; c) saturation

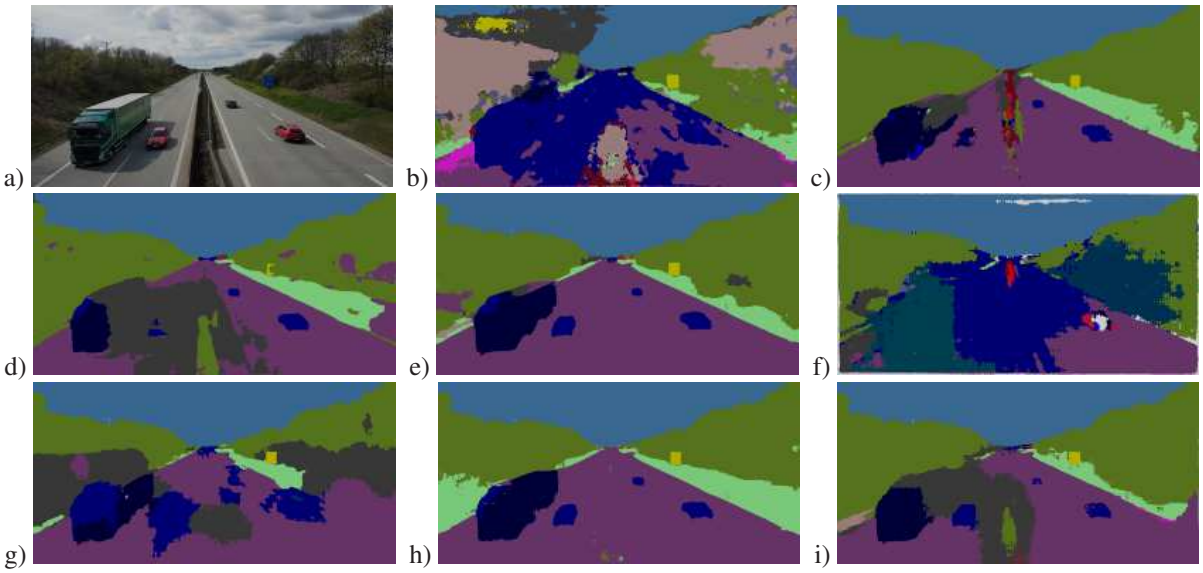


Figure 6: Segmentation-output; a) original image; b) ENet; c) ICNet; d) PSPNet; e) RefineNet; f) ESPNet; g) LRR; h) DeepLabv3+; i) TuSimple-DUC

The investigated shifts of brightness, saturation, and contrast are realistic modifications that can occur under various practical conditions, and convolutional neural networks should be able to cope with them.

The surveyed CNNs use a variety of different preprocessing steps but there does not seem to be a correlation between them and the results in this paper. ICNet, PSPNet, TuSimple-DUC and LRR subtract a fixed value for each colour channel to distribute the pixel values around zero, while ESPNet normalizes the input image with the Cityscapes dataset mean and its standard deviation. DeepLabv3+ also normalizes the pixel values x to $[-1;1]$ by $x' = (2/255) \cdot x - 1.0$. The other network architectures (ENet and RefineNet) do not use any image preprocessing steps. The DeepLabv3+ and RefineNet showed the best generalisation abilities in both tests despite their fundamentally different network architectures and preprocessing methods. The reasons for the divers robustness against varying image characteristics could not be clarified with our experimental set-up yet.

ISBN 978-80-86943-37-4

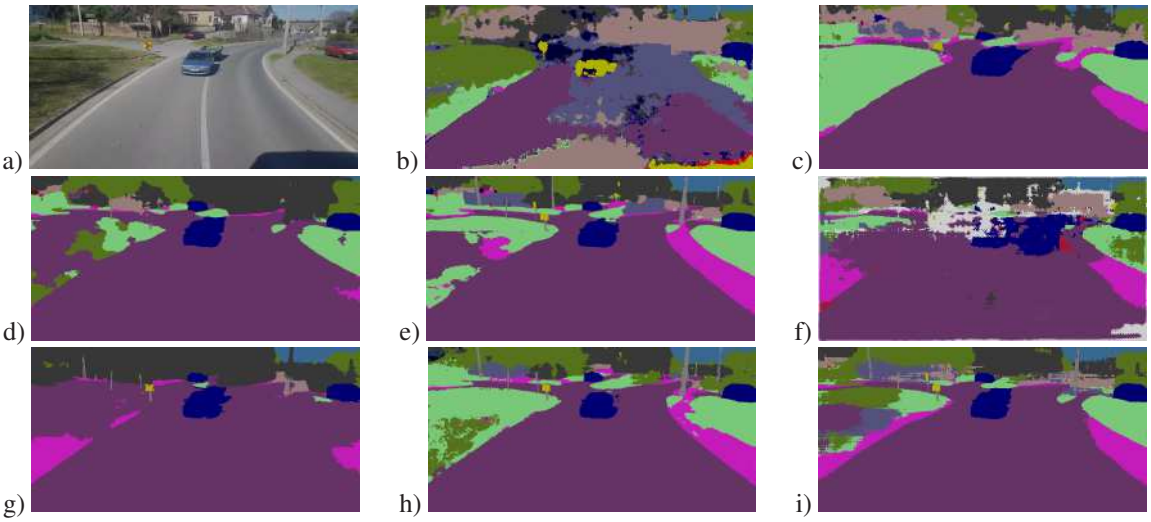


Figure 7: Segmentation-output; a) original image; b) ENet; c) ICNet; d) PSPNet; e) RefineNet; f) ESPNet; g) LRR; h) DeepLabv3+; i) TuSimple-DUC

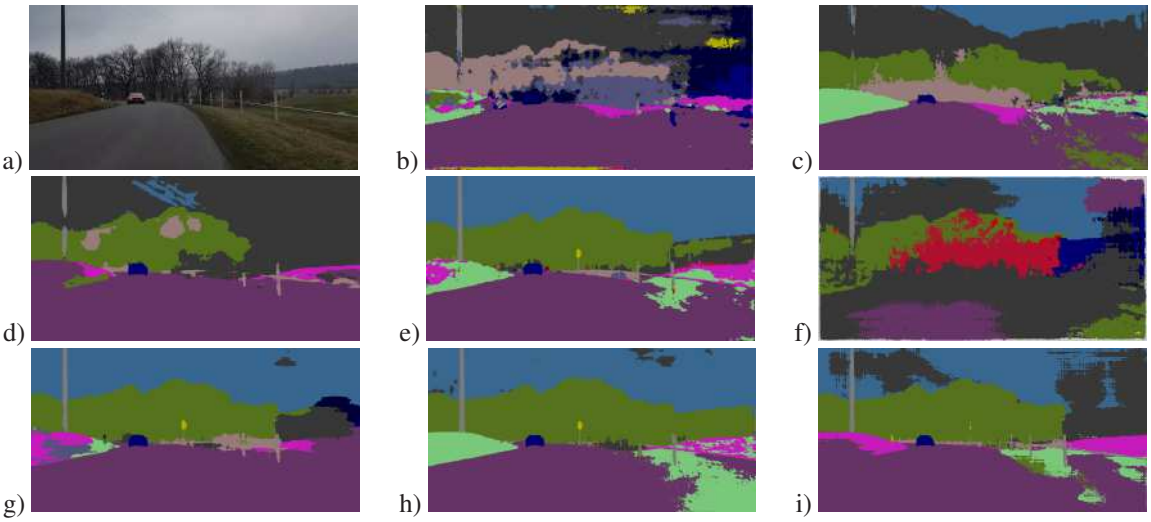


Figure 8: Segmentation-output; a) original image; b) ENet; c) ICNet; d) PSPNet; e) RefineNet; f) ESPNet; g) LRR; h) DeepLabv3+; i) TuSimple-DUC

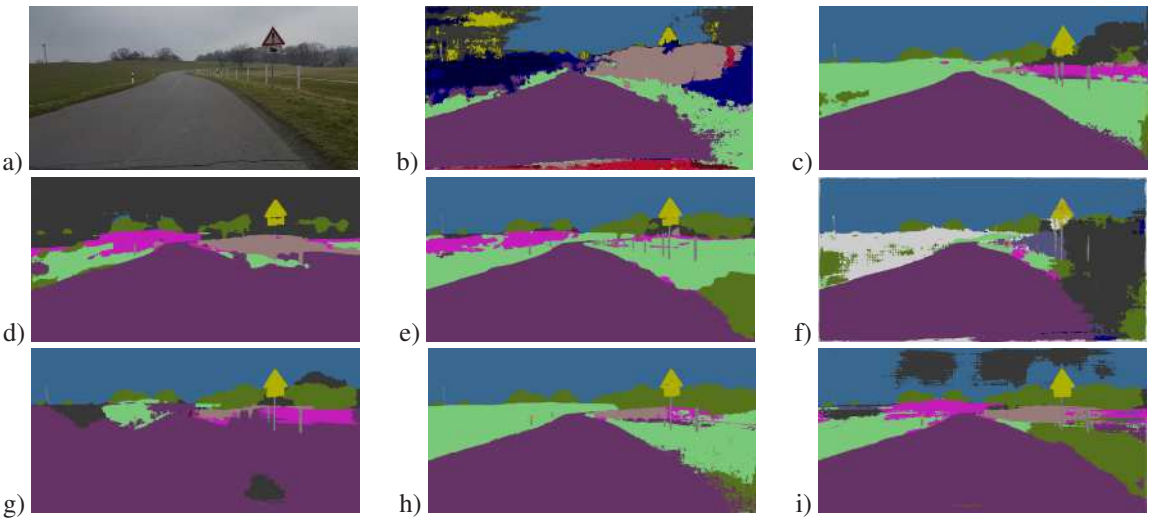


Figure 9: Segmentation-output; a) original image; b) ENet; c) ICNet; d) PSPNet; e) RefineNet; f) ESPNet; g) LRR; h) DeepLabv3+; i) TuSimple-DUC

ISBN 978-80-86943-37-4

Figure 10: Colourmap of the Cityscapes classes

5 CONCLUSIONS

Our investigations show that (i) modern CNNs are sensitive to simple image modifications in the validation dataset and that (ii) a high segmentation score on the validation or test dataset is not necessarily an indicator for a good generalisation capability of network architectures. We assume that the compared neural networks did not primarily learn the structural properties of objects in the scene, but some colour properties which coincide with objects. Consequently, segmentation scores on validation and test data are not sufficient as a benchmark test. To select a powerful network architecture, also the generalisation capability in a real-world application need to be considered.

To support reproducible research, all scripts, CNN models and images are provided in [29].

6 ACKNOWLEDGMENTS

We acknowledge the financial support by the Federal Ministry of Education and Research of Germany in the framework of the Era.Net HARMONIC project (project number 01DJ18011). Further thank goes to the unknown reviewers, whose comments allowed us to improve the initial manuscript.

7 REFERENCES

[1] Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B.: The Cityscapes Dataset for Semantic Urban Scene Understanding. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; Fergus, R.: Intriguing properties of neural networks. *arXiv preprint* arXiv: 1312.6199, 2013.

[3] Wang, B.; Gao, J.; Qi, Y.: A Theoretical Framework for Robustness of (Deep) Classifiers against Adversarial Examples. *arXiv preprint* arXiv: 1612.00334, 2016.

[4] Azulay, A.; Weiss, Y.: Why do deep convolutional networks generalize so poorly to small image transformations?. *arXiv preprint* arXiv: 1805.12177, 2018.

[5] Rosenfeld, A.; Zemel, R.; Tsotsos, J. K.: The Elephant in the Room. *arXiv preprint* arXiv: 1808.03305, 2018.

[6] Fawzi, A.; Moosavi-Dezfooli, S.M.; Frossard, P.: Robustness of classifiers: from adversarial to random noise. *Advances in Neural Information Processing Systems*, 2016, 1632–1640.

[7] Dodge, S.; Karam, L.: Understanding how image quality affects deep neural networks. *Eighth International Conference on Quality of Multimedia Experience, QoMEX*, Lisbon, Portugal, Jun. 2016, 1–6.

[8] Vasiljevic, I.; Chakrabarti, A.; Shakhnarovich, G.: Examining the Impact of Blur on Recognition by Convolutional Networks. *arXiv preprint* arXiv: 1611.05760, 2016.

[9] Zheng, S.; Song, Y.; Leung, T.; Goodfellow, I.: Improving the robustness of deep neural networks via stability training. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, 4480–4488.

[10] Ghosh, S.; Shet, R.; Amon, P.; Hutter, A.; Kaup, A.: Robustness of Deep Convolutional Neural Networks for Image Degradations. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2018, 2916–2920.

[11] Cityscapes Dataset: https://www.cityscapes-dataset.com/benchmarks/#scene-labeling-task/ last visited 25th April 2019.

[12] Everingham, M.; Eslami, S. A.; Van Gool, L.; Williams, C. K.; Winn, J.; Zisserman, A.: The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111.1, 2015, 98–136.

[13] Chen, L. C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H.: Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. *European Conference on Computer Vision (ECCV)*, 2018.

[14] Chen, L.C.; Papandreou, G.; Schroff, F.; Adam, H.: Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint* arXiv:1706.05587, 2017.

[15] Chollet, F.: Xception: Deep learning with depthwise separable convolutions. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, 1800–1807.

[16] Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J.: Pyramid Scene Parsing Network. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, 2881–2890.

[17] Wang, P.; Chen, P.; Yuan, Y.; Liu, D.; Huang, Z.; Hou, X.; Cottrell, G.: Understanding Convo-

ISBN 978-80-86943-37-4

lution for Semantic Segmentation. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018, 1451–1460.

[18] Lin, G.; Milan; A., Shen, C.; Reid, I.D.: RefineNet: Multi-path Refinement Networks for High-Resolution Semantic Segmentation. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, 5168–5177.

[19] Ghiasi, G.; Fowlkes, C.C.: Laplacian Pyramid Reconstruction and Refinement for Semantic Segmentation. *Proc. of the European Conference on Computer Vision (ECCV)*, 2016, 519–534.

[20] Zhao, H.; Qi, X.; Shen, X.; Shi, J.; Jia, J.: ICNet for Real-Time Semantic Segmentation on High-Resolution Images. *Proc. of the European Conference on Computer Vision (ECCV)*, 2017, 405–420.

[21] Mehta, S.; Rastegari, M.; Caspi, A.; Shapiro, L.; Hajishirzi, H.: ESPNet: Efficient Spatial Pyramid of Dilated Convolutions for Semantic Segmentation. *Proc. of the European Conference on Computer Vision (ECCV)*, 2018.

[22] Noh, H.; Hong, S.; Han, B.: Learning Deconvolution Network for Semantic Segmentation. *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2015, 1520–1528.

[23] Paszke, A.; Chaurasia, A.; Kim, S.; Culurciello, E.: ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation. *arXiv preprint* arXiv: 1606.02147, 2016.

[24] PSPNet: https://github.com/hellochick/PSPNet-tensorflow last visited 25th April 2019.

[25] ICNet: https://github.com/hellochick/ICNet-tensorflow last visited 25th April 2019.

[26] Smith, A.R.: Color gamut transform pairs. *ACM Siggraph Computer Graphics*, 12.3, 1978, 12–19.

[27] He, K.; Zhang, X.; Ren, S.; Sun, J.: Deep residual learning for image recognition. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, 770–778.

[28] Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A.: The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88.2, 2010, 303–338.

[29] http://www1.hft-leipzig.de/leipnitz/papers/CNNrobustness-resources/ last visited 25th April 2019.

 ISBN 978-80-86943-37-4

 ISBN 978-80-86943-37-4

Stylized Sketch Generation using Convolutional Networks

Mayur Hemani
Adobe Inc.
Adobe Systems India Pvt. Ltd.
A-05, Sector-132
Noida, U.P.-201304, India
mayur@adobe.com

Abhishek Sinha
Adobe Inc.
Adobe Systems India Pvt. Ltd.
A-05, Sector-132
Noida, U.P.-201304, India
abhsinha@adobe.com

Balaji Krishnamurthy
Adobe Inc.
Adobe Systems India Pvt. Ltd.
A-05, Sector-132
Noida, U.P.-201304, India
kbalaji@adobe.com

ABSTRACT

The task of synthesizing sketches from photographs has been pursued with image processing methods and supervised learning based approaches. The former lack flexibility and the latter require large quantities of ground-truth data which is hard to obtain because of the manual effort required. We present a convolutional neural network based framework for sketch generation that does not require ground-truth data for training and produces various styles of sketches. The method combines simple analytic loss functions that correspond to characteristics of the sketch. The network is trained on and evaluated for human face images. Several stylized variations of sketches are obtained by varying the parameters of the loss functions. The paper also discusses the implicit abstraction afforded by the deep convolutional network approach which results in high quality sketch output.

Keywords

neural-networks, image manipulation, image stylization, sketch style

1 INTRODUCTION

Stylized sketch synthesis from a photograph is an important problem in image stylization. In this paper we propose a simple framework for stylized sketch generation from images using a convolutional neural network used as an image-to-image optimization framework. We present results and analysis for human face images.

Sketches are representational artwork characterized by minimal marks (strokes, shading, etc.) on paper or other substrates. They can be hand-drawn or digitally constructed using software, and in this work we limit our scope to digitally styled sketches only. These sketches usually start from a photographic image, and transform it by means of an image processing pipeline.

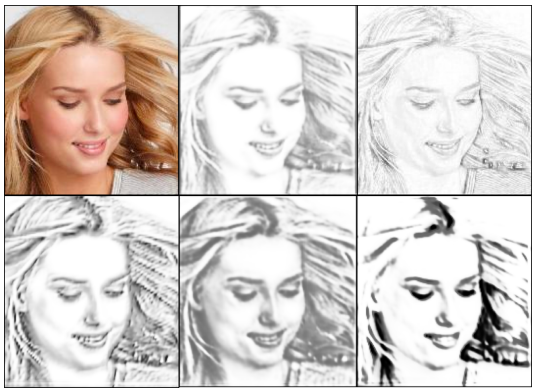


Figure 1: Sketching results using our method. The different styles of sketches are produced by varying the parameters and structure of the loss function at the top of an image-to-image neural network.

Static filter pipelines (for example in [11]) are a fast method for producing sketch-like effects. The drawback of such an approach is that its ability to capture features of a sketch drawing depends on complex heuristics that do not always work. Also, these methods lack control over the characteristics of the output sketch which depends heavily on the input image. Our method addresses both of these concerns by learning a function from image data using an optimization objective that captures the semantics of the desired sketch output. Different stylized sketch output can be constructed simply by changing the parameters and structure of the loss functions of the underlying neural network.

Neural networks are now commonly used for image stylization, starting with [6]. Sketch generation could be posed as a style-transfer problem. The primary drawback of this approach is that examples images of each sketch style are required. Our method addresses

this problem by not requiring any specifically collected paired-data. Also, the results vary significantly with style-transfer.
Stylized sketch generation can also be set up as a supervised learning problem, but the data for stylized sketches is scarce. Generative approaches using adversarial training, such as in [20], have also been proposed but they too lack control over the characteristics of the sketch output.

Our method for photograph to sketch generation comprises of a U-Net [12] like feed-forward convolutional neural-network which is trained on different loss functions that capture the semantics of the sketch output - its similarity to the input photograph, its overall brightness and contrast, and its sharpness, as determined by the amount of variation captured. We fuse a set of non-trainable classical image-processing filters at the top of the convolutional network that enables the styles to be varied either by direct combination with the output, or by enforcing a semantic constraint on it. Different loss function parameters produce different stylizations. The data used for training is a set of regular, non-sketch photographs. The network trains quickly, for fewer than ten thousand iterations, and produces high quality sketch output.

This paper makes the following contributions:

- A generalized method for fusing classical image processing techniques into a learnable filter mechanism as constraints.
- Loss functions for producing stylized sketch output, that are also extensible to other kinds of image stylization tasks.

2 RELATED WORK

Our work on sketch generation is closest to those based on style-transfer because both methods use end-to-end training of neural networks without the need for extensive data. Supervised learning methods like [1, 14, 13, 21, 18, 25, 26, 24, 19, 17, 4] learn from large sets of paired data and are directed towards producing hand-drawn sketches, and are therefore not directly related to our work.

Style Transfer Neural Style Transfer methods like [6], [8] are closely related to our work because they too train a deep neural network without the use of paired data. [8] extended the initial algorithm in [6] by training the image transformer network for each style image, thus making it possible to generate stylized images for a single style in a single pass. [2] proposed to use conditional instance normalization in image transformer networks. Their method reduces each style to a point in the embedding space and makes it possible to train a single network for several different styles. Recently, [3] obviated the need to retrain the image transformer networks for every new style by proposing a meta network that takes a style image and produces the transformer network directly. All of these methods rely on matching the low- and mid-level features of deep neural network for transferring style from a style source to another image. Our method differs from style transfer in that the style source is itself expressed as semantic constraints on the output, rather than another image.

Deep Image Prior An important reason why the method is able to produce stylized sketch images with ease is that the underlying neural network affords it a certain level of abstraction of the representational qualities of the input images. This ability of the network is discussed in the work by Ulyanov et al. [15]. They discuss how a minimally trained U-Net network ([12]) provides a good prior for the input itself and impedes noise (details). This can be considered a means of implicit abstraction of photographic images. We use this impedance of the network to noise and detail to the effect of producing a pre-styled, abstract state. The constraints imposed by the filters then enable directing the output towards the desired sketch characteristics.

3 METHOD

An overview of our framework is depicted in Figure 2. The learning pipeline comprises of a U-Net like convolutional neural network with a set of fixed, non-trainable filters at the top. The network learns to optimize a composite loss function which is constructed on the input image, the raw unfiltered output from the network and the filtered output. The sketch output is either a filtered output from the fixed filters or a linear combination of the raw unfiltered and filtered outputs. Combinations of different fixed filters and different loss functions lead to distinct sketch stylizations. For contrast independence, we employ the instance normalization trick from [16].

3.1 Dataset

The training of our network does not require any ground-truth data. We trained and validated our framework on human face images as input. We use the Celeb-A [10] aligned and cropped images for our experiments. The dataset has over 200,000 images of human faces. We use around 30,000 of these images for training, and the rest for testing. The images are center-cropped to a size 192x192 pixels for training. We also validate our method with images from the CUHK Face Sketch Database [22].

3.2 Network Architecture

The network is a U-Net [12] like fully convolutional neural network. We chose the U-Net like architecture

 ISBN 978-80-86943-37-4

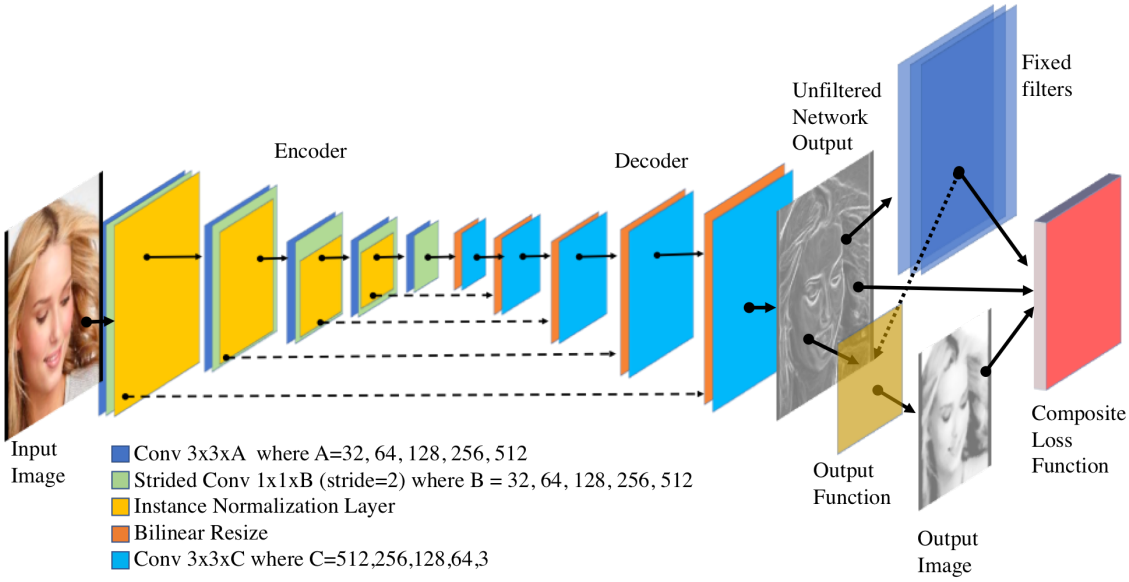


Figure 2: Network architecture.

because of the implicit prior that it affords, as discussed in [15]. In our experiments, this network trained really quickly as long as the expected output as close to the input image itself, the closeness measure itself being codified by the loss functions. All training sessions were run for fewer than ten thousand iterations. The last layer of the network has a sigmoid activation and produces a 3-channel image output. This output is then fed into a set of fixed filters including edge detection, and blur filters. The resulting output image, as well as the raw, unfiltered output from the network, along with the input images are all passed into the composite loss function. The output is selected as either the filtered output, or a combination of the filtered and unfiltered output images.

3.3 Loss Formulation

We consider sketches as an interpolate between the grayscale image and the inverted edge response image, along with some modified textural characteristics. This transformation is achieved by means of the neural network working as an image-to-image filter that is trained to optimize certain functions of the output image, and satisfy constraints defined on classical image processing filters over the output. For example, pencil or charcoal sketches can be considered as gray-scale image representations of a photographic image which have the following properties:

- The shapes and edges of the sketch match with those of the photographic image.
- The substrate color (white for the paper, for instance) is higher in mass than the pencil/charcoal marks.

Output Characteristic	Loss Representations (Mean)
Brightness Similarity	$L_s = \|\|G(Y) - G(X)\|\|^2$
Edge Similarity	$L_e = \|\|\nabla Y - \nabla X\|\|^2$
Overall Brightness	$L_b = \|\|G(Y)\|\|$
Local Brightness	$L_{lb} = \|\|\mu_{n\times n}(G(Y)) - \mu_{n\times n}(G(X))\|\|^2$
Overall Contrast	$L_c = \|G(Y)\|_\infty^{n\times n} +$ $\|\text{-G(Y)}\|_\infty^{n\times n}$
Total Variation	$L_{tv} = \|\|\nabla Y\|\|$

Table 1: Key image characteristics used for constructing loss functions. X is the input image, Y is the output, $G(x)$ is the grayscale conversion of x, ∇ is the gradient operator, $\mu_{n\times n}$ is the $n \times n$ average pooling operation, $||_\infty^{n\times n}$ is $n \times n$ max-pooling.

These ideas are used to formulate loss functions that produce different variants of sketch outputs. The general formulation includes loss terms for edge similarity L_e, brightness similarity L_{lb}, overall brightness L_b and contrast L_c of the output, its closeness to a binary image F_{bin}, a function of the overall brightness of the image (Luma CCIR601) that determines the degree of whiteness in the output F_{mass}, and a total variation term L_{tv}. The similarity terms are measured with respect to the original image. More explicitly, this is expressed as:

$$\begin{aligned} L(X,Y) =& \lambda_{bdif} L_s + \lambda_{edgedif} L_e + \lambda_{lbdif} L_{lb} \\ &+ \lambda_{bright} F_{mass}(G(Y)) + \lambda_{contr} L_c \\ &+ \lambda_{bin} F_{bin}(G(Y)) + \lambda_{var} L_{tv} \end{aligned} \quad (1)$$

Here, X is the RGB color input image, Y is the RGB output image, $G(Y)$ is the grayscale image corresponding to the output image and the λs are constant weights. The different variations of sketch output are produced

ISBN 978-80-86943-37-4

Figure 3: Pencil Sketch Output.


Figure 4: Charcoal sketch output.

by choosing different values for the λs, F_{mass}, F_{bin}, and the implementation of the gradient (∇) operator. Table 1 lists the different loss function components.

3.3.1 Shaded Pencil Sketch

Shaded pencil sketches (Figure 3(a)) are characterized by thin strokes and smears over the paper. The constants are $\lambda_{edgedif} = 1.0$, $\lambda_{bright} = 10^{-2}$, $\lambda_{bin} = 10^{-3}$.. The loss function components are:

$$\begin{aligned} L_e &= ||\nabla Y - \nabla X||^2 \\ F_{mass}(x) &= (1.0 - G(x) \cdot x) \\ F_{bin}(x) &= ||(1.0 - 4(x-0.5)^2)|| \end{aligned} \tag{2}$$

3.3.2 Traced Pencil Sketch Output

A contoured sketch output (Figure 3(b)) can be produced where instead of edge-matching with the Sobel operator, for the input image X, we employ a formulation which is similar to the Difference of Gaussians method for edge detection as used in [23] and implementing a selection operation on the edges in the input image. The image predicted by the network (Y), is a selection mask over the edges. The objective function itself is to minimize the difference between the Y_{out} image and the grayscale vector of the input image $G(X)$, as well as reduce the overall edge response of the predicted mask Y.

$$\begin{aligned} G_{pooled} &= |G(X)|_{\infty}^{4\times4} \\ G_{shifted} &= R_{4\times4}(G_{pooled}, w, h) \\ \nabla X &= G_{shifted} - G(X) \end{aligned} \tag{3}$$

Where $R_{4\times4}$ is a re-scaling operation that restores the size of the max-pooled image to the original size. The output image is computed as:

$$Y_{out} = 1.0 - \nabla G(X) \cdot G(Y) \tag{4}$$

$$L(X,Y) = \lambda_{lbdif} ||Y_{out} - G(X)||^2 + \lambda_{tv} L_{tv} \tag{5}$$

Where $\lambda_{lbdif} = 1.0$, and $\lambda_{tv} = 0.01$

3.3.3 Charcoal Drawing Generation

Charcoal drawings (Figure 4) have larger smears and deeper black shades. The corresponding loss function components are the same as for shaded pencil sketches 2 except for the additional contrast terms and the λ values:

$$L_c = |G(Y)|_{\infty}^{4\times4} + |-G(Y)|_{\infty}^{4\times4} \tag{6}$$

with $\lambda_{edgedif} = 1.0$, $\lambda_{bright} = 10^{-3}$, $\lambda_{contr} = 0.1$, and $\lambda_{bin} = 0.1$.

3.4 Contrast Invariant Output

One major reason why we employ a learning method for sketch generation is that it can be used to learn the map from a color image to a sketch image irrespective of the contrast of the input. For this, instance-normalization (as described in [16]) layers are inserted into the encoder part of the network. Figure 5 shows the results without and with the instance normalization trick.

3.5 Results on other datasets

Figure 6 depicts some results from the CUHK Face Database. These results are not really comparable to the hand-drawn ground-truth images, but they are produced for comparison with other contemporary methods.

3.6 Training and Stability

We use the Adam optimizer [9] for training the network. Many of the loss functions we use have stable local minima which poses a challenge in training. For example, if the network is initialized randomly, and trained with L_{bin} as part of the loss function, sometimes it results in an inverted initial output. And then it takes very long for the network to converge towards the global optimum (or even a good local optimum). We could reduce the learning rate or reduce the weights of these losses considerably, but that too results in delay in convergence. We found that training with the L_2 loss between the input (grayscale) and output images (reconstruction task) for only a thousand iterations initially, produces a very good prior and adds to the stability of

 ISBN 978-80-86943-37-4

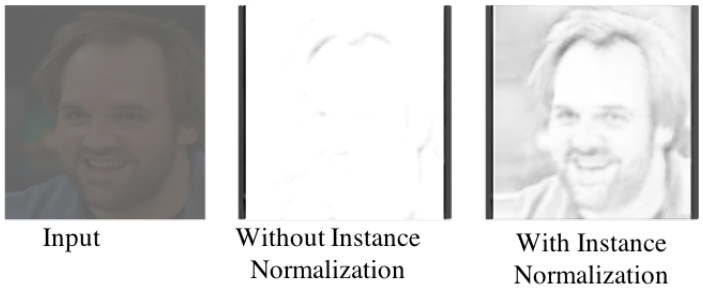


Figure 5: Output without and with instance normalization.

Figure 6: Pencil and charcoal sketch output for the CUHK dataset. Top row: Charcoal sketches, Bottom row: pencil sketch output.

Figure 7: Quilt-like sketch output produced by matching the inverted edge image against the grayscale input image.

the training. In our experiments, with an initial training for reconstruction, the network never converged to a stable local optimum.

Just as in illustration workflows, where an abstract shape or form serves as a good starting point for producing styled art, this abstraction implicit in the network serves as a good starting point for learning automatic stylization. We can use this fact to produce artistic renditions of the input image that abstract away certain representational qualities but retain the semantics. Sketch generation is an instance of this sort of abstraction.

To further corroborate the insight that the network does learn a representation close to the original image even when it is not fully trained, we train our network with a combination of three conflicting loss function with clear trade-offs. This ensures that the network never fully converged to the minimum of either of the loss function components:

$$\begin{aligned} L(X,Y) &= L_{edge} + L_{bdif} \\ L_{edge}(X,Y) &= ||(1 - F((G(Y)) - G(X)||_2 \\ L_{bdif}(X,Y) &= ||\mu_{4\times4}(G(Y)) - \mu_{4\times4}(G(X))||_2 \end{aligned} \tag{7}$$

where the filter function F(Y) is the average of the gradient responses (computed using the Sobel operator) in the x- and y- directions.

$$F(Y) = 0.5 \cdot (\nabla_x(Y) + \nabla_y(Y)) \tag{8}$$

The loss functions pit the (inverted) edge response against the grayscale value of the input image. For photographic images, the gradient response is rarely equal to the input image G(X), because the intensities of the pixels are not exponential in their coordinates. Therefore, this is a very hard optimization problem for the network. As a result, the network learns to abstract out details and produces only the most necessary edge responses. Training with this loss function produces a quilt-like pattern(Figure 7) from the network.

3.7 Comparison with Style Transfer

Our results are compared with those from style transfer methods. To produce results from style transfer, we took a synthesized sketch image from our method as a style image and used it to transfer style over to a photograph in the dataset via the approach of neural style transfer as in [6]. The results are depicted in Figure 8.

3.8 Extended Results

The versatility of our approach is indicated by the diversity of results that can be obtained by simply changing the underlying loss function parameters. Figure 9 demonstrates the results on high-resolution images. The last output is obtained by capturing the color output times the inverted edge response, while keeping identical objectives as a pencil sketch, besides an L_2 distance between the color output and the input image.

ISBN 978-80-86943-37-4

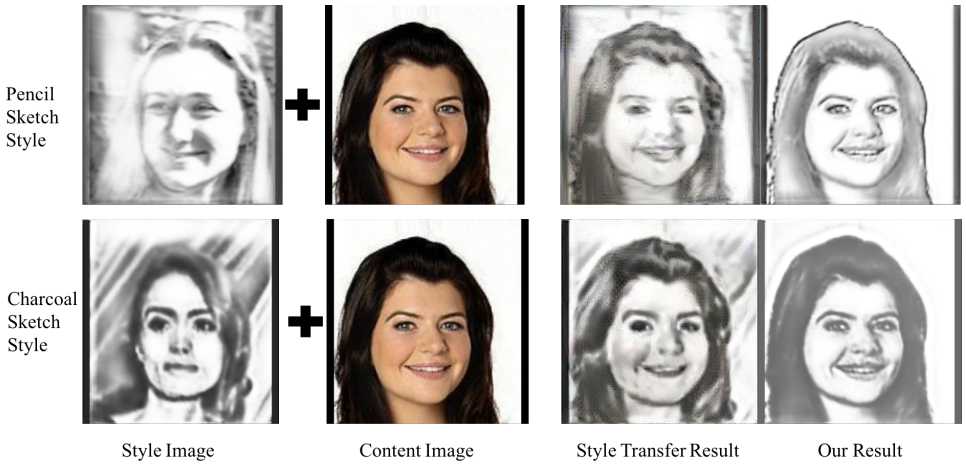


Figure 8: Comparison with style transfer outputs. The style transfer output loses semblance with the content image, while our method produces better looking sketch output.

Figure 9: High Resolution Results. The last result is obtained by capturing the color output from the network times the inverted edge response as the output.

4 CONCLUSIONS

This paper presents a framework for producing stylized images without the need to train with paired style and ground-truth data. It focuses on producing stylized sketch output by means of fusing classical image processing filters into a learning framework based on an intuitive loss function. It also demonstrates how the underlying abstraction afforded by the deep neural network aids in producing stylizations.

5 REFERENCES

[1] Chaofeng Chen, Xiao Tan, and Kwan-Yee K. Wong. Face sketch synthesis with style transfer using pyramid column feature. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 485–493. IEEE, 2018.

[2] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. 2017.

[3] Shuicheng Yan Falong Shen and Gang Zeng. Neural style transfer via meta networks. In *CVPR2018*, 2018.

[4] Fei Gao, Shengjie Shi, Jun Yu, and Qingming Huang. Composition-aided sketch-realistic portrait generation. *arXiv preprint arXiv:1712.00899*, 2017.

[5] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[6] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional

ISBN 978-80-86943-37-4

neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[7] Xun Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *CoRR*, abs/1703.06868, 2017.

[8] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016.

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[11] Cewu Lu, Li Xu, and Jiaya Jia. Combining sketch and tone for pencil drawing production. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, NPAR '12, pages 65–73, Goslar Germany, Germany, 2012. Eurographics Association.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[13] Yibing Song, Linchao Bao, Qingxiong Yang, and Ming-Hsuan Yang. Real-time exemplar-based face sketch synthesis. In *European Conference on Computer Vision*, pages 800–813. Springer, 2014.

[14] Yibing Song, Jiawei Zhang, Linchao Bao, and Qingxiong Yang. Fast preprocessing for robust face sketch synthesis. *arXiv preprint arXiv:1708.00224*, 2017.

[15] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. *arXiv:1711.10925*, 2017.

[16] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

[17] Lidan Wang, Vishwanath Sindagi, and Vishal Patel. High-quality facial photo-sketch synthesis using multi-adversarial networks. In *Automatic Face & Gesture Recognition (FG 2018), 2018 13th IEEE International Conference on*, pages 83–90. IEEE, 2018.

[18] Nannan Wang, Dacheng Tao, Xinbo Gao, Xuelong Li, and Jie Li. Transductive face sketch-photo synthesis. *IEEE transactions on neural networks and learning systems*, 24(9):1364–1376, 2013.

[19] Nannan Wang, Wenjin Zha, Jie Li, and Xinbo Gao. Back projection: an effective postprocessing method for GAN-based face sketch synthesis. *Pattern Recognition Letters*, 107:59–65, 2018.

[20] Nannan Wang, Mingrui Zhu, Jie Li, Bin Song, and Zan Li. Data-driven vs. model-driven: Fast face sketch synthesis. *Neurocomputing*, 257:214–221, 2017.

[21] Shenlong Wang, Lei Zhang, Yan Liang, and Quan Pan. Semi-coupled dictionary learning with applications to image super-resolution and photo-sketch synthesis. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2216–2223. IEEE, 2012.

[22] X. Wang and X. Tang. Face photo-sketch synthesis and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):1955–1967, Nov 2009.

[23] Holger Winnemöller, Jan Eric Kyprianidis, and Sven C. Olsen. Xdog: an extended difference-of-gaussians compendium including advanced image stylization. *Computers & Graphics*, 36(6):740–753, 2012.

[24] Dongyu Zhang, Liang Lin, Tianshui Chen, Xian Wu, Wenwei Tan, and Ebroul Izquierdo. Content-adaptive sketch portrait generation by decompositional representation learning. *learning*, 2:111.

[25] Shengchuan Zhang, Xinbo Gao, Nannan Wang, and Jie Li. Robust face sketch style synthesis. *IEEE Transactions on Image Processing*, 25(1):220–232, 2016.

[26] Shengchuan Zhang, Xinbo Gao, Nannan Wang, Jie Li, and Mingjin Zhang. Face sketch synthesis via sparse representation-based greedy search. *IEEE transactions on image processing*, 24(8):2466–2477, 2015.

 ISBN 978-80-86943-37-4

 ISBN 978-80-86943-37-4

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

SyCaT-Vis: Visualization-Based Support of Analyzing System Behavior based on System Call Traces

Alrik Hausdorf[1]
hausdorf@informatik.uni-leipzig.de

Nicole Hinzmann[1]
hinzmann@informatik.uni-leipzig.de

Dirk Zeckzer[1]
zeckzer@informatik.uni-leipzig.de

[1]Image and Signal Processing Group,
Leipzig University

ABSTRACT

Detecting anomalies in the behavior of a computer system is crucial for determining its security. One way of detecting these anomalies is based on the assessment of the amount and sequence of system calls issued by processes. While the number of processes on a computer can become very large, the number of system calls issued during the lifespan of such a process and its subprocesses can be humongous. In order to decide whether these anomalies are due to the intended system usage or if they are caused by malicious actions, this humongous amount of data needs being analyzed. Thus, a careful analysis of the system calls' types, their amount, and their temporal sequence requires sophisticated support. Visualization is frequently used for this type of tasks. Starting with a carefully aggregation of the data presented in an overview representation, the quest for information is supported by carefully crafted interactions. These allow filtering the tremendous amount of data, thus removing the standard behavior data and leaving the potentially suspicious one. The latter can then be investigated on increasingly finer levels. Supporting this goal-oriented analysis, we propose novel interactive visualizations implemented in the tool SyCaT-Vis. SyCaT-Vis fosters obtaining important insights into the behavior of computer systems, the processes executed, and the system call sequences issued.

Keywords

Security visualization, system call traces, security analysis, behavior analysis

1 INTRODUCTION

Detecting anomalies in the behavior of a computer system is crucial for determining its security. The analysts need to decide if these anomalies are due to intended system usage or if they are caused by malicious actions. To do so, the behavioral analysis can be based on observing the system calls issued by processes like memory reads and writes, network usage, and CPU usage, among others. Besides the types of the system calls, their amount and their sequence are important clues for separating intended from malicious behavior.

Overall, this behavioral system analysis is quite involved due to the huge amount of individual system calls and the already large number of processes that run in parallel. To date, only some approaches for automated analysis of system call traces were proposed (Section 2.1). Regarding visual support of these analyses, only three approaches were found in the literature [16, 14, 2] (Section 2.2). Our contribution is the tool SyCaT-Vis (SystemCallTrace-Visualization) providing three views containing interactive visualizations adapted to this situation:

1. A Context View showing all traces that were captured.
2. A Process view proving more detailed, temporal information about programs and program groups.
3. A detailed thread view showing groups of system calls as well as individual system calls themselves.

This realizes a semantic zooming environment that together with filtering as well as additional interactions fosters the analysts insights into the process behavior recorded and thus eases analyzing the security of the computer system under investigation (Section 4).

2 RELATED WORK

2.1 Automated Analysis of System Call Traces

Analyzing the behavior of a system is often performed based on data about which processes are running and

Data Set	t	CA	E	SC	SCT	SCC	C	U	P	T
1	1s	i	21,362	10,756	23	9	4	6	22	75
2	5s	i	62,353	31,339	72	12	1	5	84	143
3	8s	i	105,384	52,900	81	12	1	7	101	165
4	∼60s	i	784,167	392,896	88	12	1	7	304	516
5	∼27s	au	3,674,663	1,838,122	106	13	7	14	145	715

Table 1: Basic values of the benchmark data (t: time span; CA: computer activity (i: idle, au: actively used); and the contained number of elements (see also Table 2; E: events (without switch events); SC: system calls; SCT: system call types; SCC: system call categories; C: containers; U: users; P: processes; T: threads)

Information Type	Result-Set	sysdig Parameters
event information	E_{inf}	evt.num, evt.rawtime, evt.latency, evt.dir
system call type	S_{type}	syscall.type
event category	E_{cat}	evt.category
container	C	container.id, container.name
user	U	user.uid, user.name
process	P	proc.pid, proc.exe, proc.args, proc.pname, proc.name
thread	T	thread.tid

Table 2: Attributes extracted from the complete sysdig trace.

which system calls are issued by them. To obtain low false-positive rates in detecting malicious system behavior, several different models supporting the automated analysis of system call traces have been developed.

Forrest et al. [6, 4, 5] developed an n-gram pattern approach. Sequences of system calls are analyzed and those stemming from normal process execution are used to form pattern that are stored in a database. The system call sequences to be analyzed are then compared to the pattern stored in the database and mismatches are reported. Sekar et al. [10] and Yu et al. [17] use finite state automata instead of the n-grams for this analysis. Ghosh et al. [7] use neural networks to determine whether or not the system behavior is malicious, while Liao et al. [8] use k-nearest neighbor classification for pattern detection overcoming the need of learning sequence pattern for individual programs.

Warrender et al. [15] compared several methods to detect intrusions: enumeration of observed sequences, a rule induction technique, and Hidden Markov Models. They conclude that "weaker methods than Hidden Markov Models are likely sufficient" for these tasks.

Coull et al. [1] use an approach that is based on techniques used in bioinformatics to uncover masquerade attacks. The trace to be analyzed is compared to previous traces using semi-global alignment. A similarity score is computed and based on this score the similarity or dissimilarity between the traces that are compared can be assessed.

Mazeroff et al. [9] employ an approach based on probabilistic models. They construct probabilistic suffix trees and translate them into probabilistic suffix automata. The resulting models are then used for monitoring data in real-time.

Shu et al. [13] analyze the correlation among events using long-span behavior anomaly detection based on mildly context-sensitive grammar verification.

All these approaches rely on machine learning approaches, generating models and assessing the systems' behavior according to these models. No visual support for analyzing the results is provided.

2.2 Visual Analysis of System Call Traces

Visualizations supporting system behavior analysis based on system call traces are less common. Wu et al. [16] propose "lviz", a tool visualizing Microsoft Windows system call traces. Their visualization shows two call traces and the contained events in a dotplot matrix. Tandon et al. [14] visualize distances of motifs found in system calls using scatterplots. Lately, the tool Csysdig [2] showing the latency of system calls per time in a spectrogram was proposed.

Our tool provides three visualizations allowing on the one hand to obtain a general overview and on the other hand analyzing and comparing several process execution traces at the same time. Moreover, the user can smoothly change between these views in a top-down or bottom-up manner according to her workflow.

3 DATA SETS

For testing the performance and showing the features of our visualization, we use self-generated data sets (Table 1). The behavioral data was collected using the tool "sysdig" [3]. This tool does not provide system calls directly. Therefore, these have to be extracted from the

 ISBN 978-80-86943-37-4

behavioral data comprising the events associated with the system calls. Sysdig stores the data obtained in compressed, binary files, the proposed file extension being “scap”.

As the amount of attributes collected by sysdig is very large, it is useful to extract those attributes that should be analyzed before subsequent operations. This is done calling sysdig using a set of parameters indicating the attributes to be extracted. The information we are interested in together with the corresponding sysdig parameters is given in Table 2.

"switch"-events are not related to system calls and just indicate a context switch, i.e., when a thread is put to sleep by the process scheduler, while another thread will be executed. Therefore, they are filtered from the data. Data selection and filtering are performed while loading the original data into our tool.

Data Set 5 contains the largest amount of collected system calls and therefore will be used as example for presenting our approach.

4 SYCAT-VIS

4.1 System Overview

To support analyzing a system’s behavior based on its system call traces obtained using sysdig [3], we propose SyCaT-Vis (SystemCallTrace-Visualization). All data to be analyzed is stored in a PostgreSQL database. SyCaT-Vis imports the data collected into this database by running the sysdig command as described in the previous section for selecting and filtering the original system traces (scap-file) followed by parsing the results and writing them into the database. The importer itself is written in the Java programming language.

A ReSTful API written in NodeJS connects the SyCaT-Vis user interface to this database. It also serves as a cache for database requests needing a long computing time. The user interface itself is written in JavaScript using the AngularJS library (version 1.7.7) for data and interaction handling, and the D3.js library for creating the visualizations.

Currently, three views are provided by the user interface for analyzing the system behavior following Shneiderman’s mantra “Overview first, zoom and filter, then details on demand” [12]. The Context View (Section 4.2) provides an *overview*: information about the main attributes and their values in the information areas, a user-selected hierarchy of attributes in the main area, and a configuration area for selecting and deselecting the attributes and their hierarchy shown in the main area realizing a *filter*. A contextual popup-menu provides additional information (*details on demand*) and allows using the currently selected element as a *filter* in the Process View.

The Process View (Section 4.3) is a more detailed, *intermediate view* (*zoom*) showing the time evolution of all processes or of those processes that match the filter conditions selected. Various interactions allow modifying the display of the information as well as adapting the *filters*. Altogether, the filter conditions can be selected in the context view and they can be selected and changed in the process view.

Selecting individual processes in the Process View can in turn be used as a filter for the most *detailed view*, the Thread View (Section 4.4). This view shows the system calls for the selected threads of a process from a coarse grained aggregated view summarizing system calls to a fine grained view showing individual system calls, thus realizing several zooming levels.

We describe these views in the subsequent sections using Data Set number 5 (Table 1) as a running example.

4.2 Context View

The *main area* of the context view (Figure 1) contains a circle-based hierarchical view showing an overview of the data (Figure 1 (c), Section 4.2.3). It is complemented by a *configuration area* allowing selecting the elements displayed in the hierarchy and their order (Figure 1 (b), Section 4.2.2). Moreover, the configuration area provides selecting scaling the attribute values being visualized linearly or logarithmically. The *information areas* to the left and to the right of the visualization and interaction areas provide information about the entries for the main attributes (container: upper left, users: lower left, processes: upper right, and system call types: lower right) (Figure 1 (a), Section 4.2.1). Finally, a *contextual popup menu* provides further interaction facilities (Figures 1 (d) and 2, Section 4.2.4). Next, we describe each of these areas in the order of the work flow of an end user.

4.2.1 Information Areas

The *information areas* (Figure 1 (a)) are to the left and to the right of the visualization and interaction areas providing information about the different values of the main attributes. These correspond to the previously (Section 3) defined sets C: Container (upper left), U: Users (lower left), P: Processes (upper right), and S_{type}: System Call Types (lower right). For each of these attributes, its name and the amount of different values are given. Additionally, the eye-icon indicates that this dimension is currently part of the hierarchy displayed in the main area of this view. The attribute values are sorted by the amount of system calls they are connected to in descending order. The top eight attribute values according to this order are shown together with the amount of system calls they are related to. Finally, a hint about how many more attribute values are not shown is displayed. On mouse-over one of the attribute

 ISBN 978-80-86943-37-4

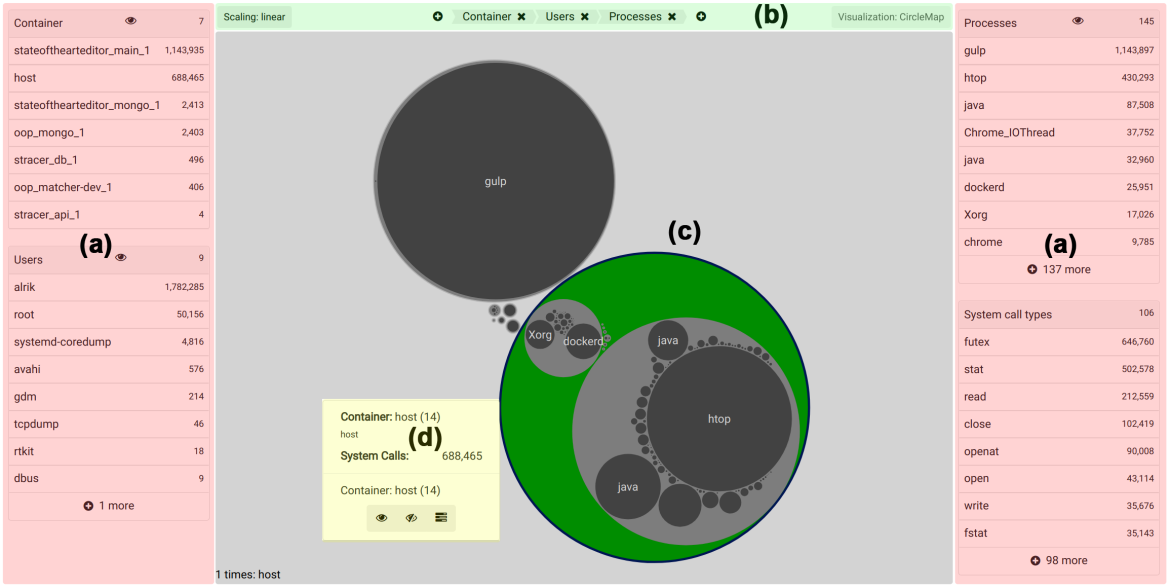


Figure 1: The Context View containing the following areas: (a) Information Areas showing the attributes that can be included into the hierarchy (b) shown in the main area (c); (b) Configuration Area allowing to construct the hierarchy to be analyzed as well as to switch between linear and logarithmic scaling; (c) Main Area showing a circle-based hierarchical visualization representing the currently active attribute hierarchy; (d) Context Menu showing information about the selected element (green) and providing interaction facilities.

values, all occurrences of this attribute value in the main area are highlighted, if this attribute is shown there.

In our example, the "Container", the "Users", and the "Processes" are part of the hierarchy shown in the middle (having an eye icon), while the "System Call Types" are not (no eye icon). All container attribute values are shown (7) in its list, while only 8 out of 10 users (2 more not displayed), 8 out of 145 processes (137 more not displayed), and 8 out of 106 system calls types (98 more not displayed) are listed in the respective areas. User "alrik" is associated with the largest number of system calls (1,782,285), followed by "root" (50,156).

4.2.2 *Configuration Area*

The configuration area is located between the two information areas at the top (Figure 1 (b)). In the middle of the configuration area, the attributes currently shown in the main area are displayed in the order in which they form the hierarchy from top (left) to bottom (right). Each of the currently shown attributes can be removed. Moreover, new attributes can be added to the top or to the bottom of the hierarchy.

On the left side of the configuration area, the user can switch between linear and logarithmic scaling of the attribute values visualized in the main area. Logarithmic scaling is especially useful if small elements should be made more visually salient.

In the example given in Figure 1, the hierarchy is "Container" (top) – "Users" – "Processes" (bottom). Moreover, linear scaling is chosen.

4.2.3 *Main Area*

In the main area of the context view (Figure 1 (c)), the currently selected hierarchy is displayed using a circle-based hierarchical view. The circles are color-coded according to the hierarchy they belong to, the lightest color representing the top attribute of the hierarchy and the darkest color representing the bottom attribute of the hierarchy. Moreover, the size of each circle reflects the number of the associated system calls.

In the example, the basic color is gray. Thus, the seven values of the top-most attribute of the hierarchy ("Con-

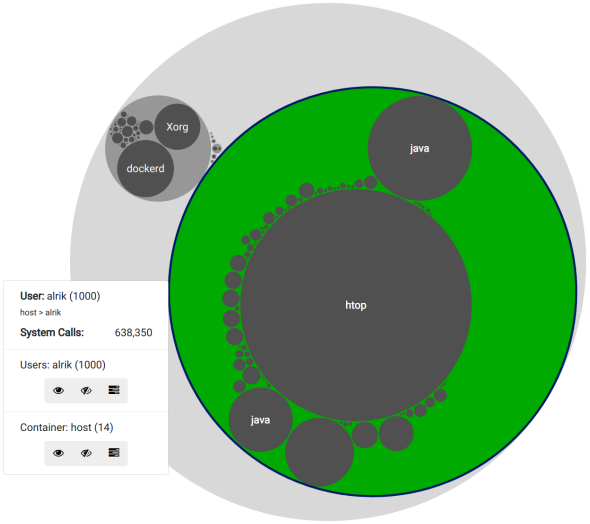


Figure 2: Context View restricted to the container "host" with user "alrik" being selected.

ISBN 978-80-86943-37-4

tainer") are mapped to seven light-gray circles that are arranged next to each other in the main area. The values of the next attribute in the hierarchy ("Users") are grouped by container value and each container-users pair is mapped to a middle-gray circle. These circles are arranged inside the circles representing their respective containers. Finally, the values of the bottom-most attribute of the hierarchy ("Processes") are grouped by container-users value pairs yielding container-users-processes triplets. Each such triplet is mapped to a dark-gray circle. These circles are arranged inside their respective container-users circles.

If an attribute value corresponding to a circle is selected, the circle is colored green. In the example, the green circle represents the container "host". Within this container, there are circles for all users active in this container (eight). Some of these circles are very small and can barely be seen. Logarithmic scaling could be used to enlarge them when needed. Within each of the circles representing the users, the dark circles represent the processes like the two "java" processes and the single "htop" process inside the circle representing the lower right user of the container "host", or the process "gulp" on the upper left.

Figure 2 shows the main area restricted to the container "host" with user "alrik" being selected (circle colored green). Only the main area is shown.

We also tested a squarified treemap [11] representation of the same data (see also `http://www.cs.umd.edu/hcil/treemap-history/`). In general, a treemap is emphasizing leaf information over hierarchy information. Also, treemaps are prone to create very small, sub-pixel wide or high stripes. Comparing the circle layout to a squarified treemap layout showing the same data (Figure 3), it can be seen that the hierarchy is more salient and that the processes with less activity are easier to discern in the circle representation (Figure 3(a)) compared to the squarified treemap representation (Figure 3(b)). Further, it can be seen, that the unused space around the circles helps to spot elements that are very small and that can not be identified in the squarified treemap representation. As both the hierarchical information and the inner nodes (and information about them) are both important for our application, the circle-based hierarchical view was chosen.

4.2.4 Context Menu

Selecting a circle in the main area results in displaying a contextual popup menu (Figure 1 (d)), Figure 2) showing information about the element associated with the circle representing a specific attribute selected (Table 3 shows the values for the examples provided in Figure 1 (d) and Figure 2):

- The *name* of the *attribute*
- The *value* of the *attribute*

Information shown	Figure 1 (d)	Figure 2
Attribute Name	Container	User
Attribute Value	host	alrik
Element ID	14	1000
Hierarchy	host	host > alrik
System Calls Amount	1,375,457	1,275,438

Table 3: Attribute values provided by the popup menu shown in Figure 1 (d) and Figure 2, respectively.

- The *id* of this *element*
- The elements of the *hierarchy* starting at the top and ending at the selected element including the selected element
- The *amount of system calls* that are associated with the selected element

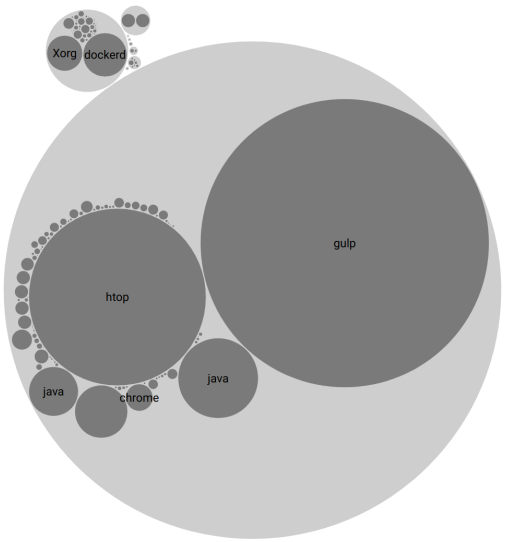
Moreover, the context menu provides several interaction facilities. For each hierarchy level above and including the current element, a filter can be selected at the bottom of the menu. The filter options are to select the element (none selected), to hide the element (strike-through eye, middle), and show only this one (eye, left). Each set of icons is annotated by the hierarchy information in the format "attribute: attribute value (id)". In Figure 1 (d), there is only one entry: "Container: host (14)", while in Figure 2, there are two entries: "Users: alrik (1000)" and "Container: host (14)" according to the two hierarchy levels involved in the selection. The third icon changes to the Process View (Section 4.3) showing the activities of the processes filtered by the selected element.

4.2.5 Interpretation

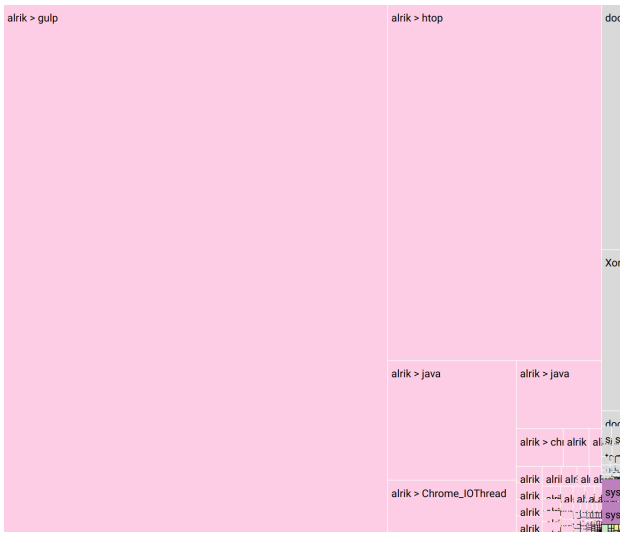
The Context View provides a general overview over the data set. Using this overview, first assumptions about the usage of the monitored computer are created. To do so, a solid knowledge is needed about which users are executing processes on this machine, which processes are expected to be running, and what the non-malicious behavior on this machine should look like. In the example shown in Figure 1, the existence of containers is expected on the machine under exam and thus should not cause an alert. However, a similar image based on the data drawn from an office only computer could be an indication of a misuse of the machine. Together with the activity of processes used by different users, this could indicate malicious behavior, e.g., by a malware infected computer.

On the other hand, if the inspected system is a server, the view of the container and the active processes inside is helpful. Based on the idea to have one container for one purpose, there should not be several processes that are equally active. If a container looks like the selected host of Figure 2, the assumption that the container was attacked could be made and that, e.g., a reverse shell

ISBN 978-80-86943-37-4



(a) Each user is represented by a light grey circle. The running processes of the user are represented by circles having a darker grey color located inside the user's circle. The size of the circles represents the amount of system calls aggregated for the respective circle. Compared to the treemap layout (b), the low activity processes are easier to discern and the hierarchy is salient.



(b) Squarified treemap representation of the "User" > 'Processes" hierarchy. Each user is mapped to an individual color. The size of the leafs representing the processes is mapped to the amount of their system calls. Compared to the circle layout (a), the low activity processes more difficult to discern and the hierarchy is less salient.

Figure 3: Comparison between the circle layout (a) and the squarified treemap layout (b) showing the Main Area of the Context View with no filters representing the hierarchy is "User" > "Processes".

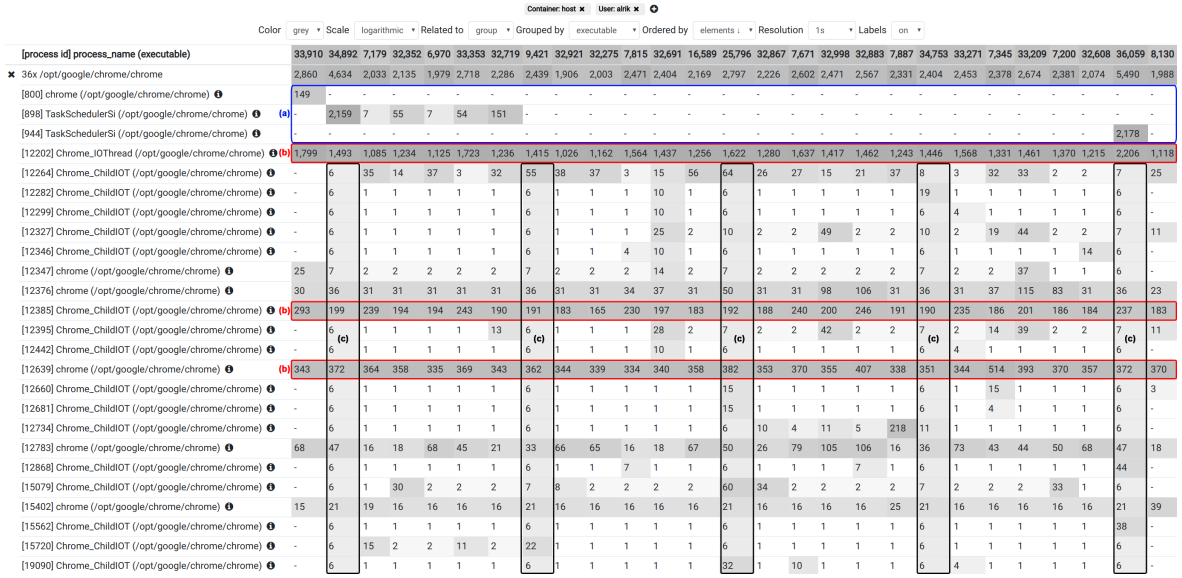


Figure 4: The Process View for user "alrik" in the container "host". The number of system calls per process overlapping a specific time step is encoded using a logarithmic grey scale. The relation is "group" and the processes are grouped by "executable". The groups are ordered by the count of "elements". The time resolution is one second.

was used to deploy malicious software inside a container.

4.3 Process View

The Process View (Figure 4) shows an overview of processes and their corresponding activities. Either all processes are shown or only the ones that were previously selected in the Context View. The filter currently applied to this view is shown at the top of the view. Filters can be removed and added there, too.

4.3.1 Design

A tabular design was chosen for displaying the processes and their evolution over time. Each process is

ISBN 978-80-86943-37-4

represented by a row and each time step is represented by a column. The number of system calls overlapping a time step is represented by the respective cell's lightness: the darker the cell, the more system calls are associated with the (sub-)process and overlap this time step.

By default, the processes are grouped by the "parent_name", a 16 bit identifier for the process. Every process in a group is spawned from the same named parent. It is possible that processes are spawned from different parent processes with the same parent_name. The parent_name should be a significant identifier for the activity or for the goal of the process. Therefore, the grouping of processes with similar named parents is interesting to us. Another option is to group the processes by the associated executable.

The processes can be ordered by the process id of the first system call in each group, or by the number of processes per group. The ordering can be either ascending or descending.

The labels shown in Figure 4 (process id, executable, and number of system calls) can be hidden such that only the visual elements are shown. Then, the information can be shown on mouse-over the respective cell.

Besides changing the filters, additional interaction facilities are provided by this view (Figure 4, below the filters; selected values are emphasized):

- Select the color hue ("Color": *grey*, blue, red)
- Select the scale ("Scale": linear, *logarithmic*)
- Select the relation ("Related to": *group*, global)
- Group by ("Grouped by": *executable*, parent_name)
- Order by ("Ordered by": *process id*, number of elements in group; *ascending*, descending)
- Select the time resolution ("Resolution": between 50ms and *1s*)
- Select if labels are shown ("Labels": *on*, off)

Here, the processes in the container "host" for user "alrik" are shown that are related to the executable "chrome". Overall, 25 processes and 27 time steps are shown in Figure 4.

Several pattern emerge from this visualization:

(a) The first three processes are different from the remaining ones. (see blue bordered area (a) in Figure 4)
(b) Several processes are constantly active, i.e., every time step. (see red bordered area (b) in Figure 4)
(c) Several processes show a regular activation (sub-)pattern. (see black bordered area (c) in Figure 4) They are at least active every six seconds.

Based on these pattern, hypotheses about the executable's behavior can be created.

4.3.2 Interpretation

Being an intermediate view between the Context View and the Thread View, the Process View provides the temporal dimension of active processes that are related to each other. One malicious behavior that is identifiable using this view is reverse-shells on servers. The possible indication for this type of malicious behavior is a long idle time followed by an extremely high amount of activity of another process directly afterwards, e.g., a php-child.

A similar example of this behavior—a process starting a new process that performs some sub-tasks—can be found in Figure 4 in the lines marked with an (a). In this case, the chrome process with process id 800 (line before) has no further activities after starting. However, directly afterwards, a high amount of system calls by its child process having the pid 898 is observed. Knowing the spawning schemata of specific processes, like using a fixed process-pool for computation, it can be assessed whether or not this behavior of creating new sub-processes and delegating tasks to them is malicious.

4.4 Thread View

4.4.1 Design

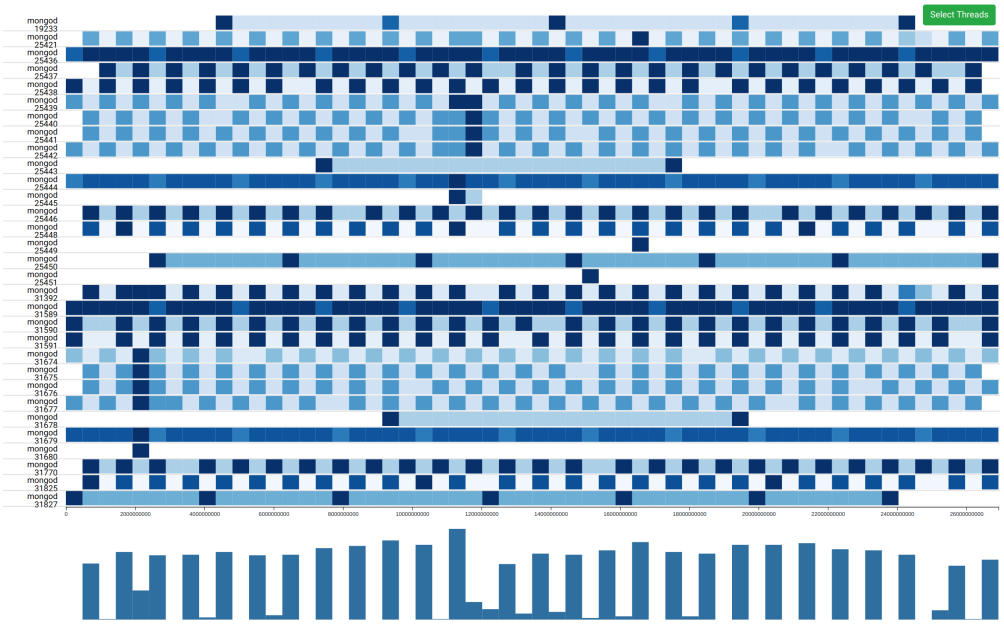
Figure 5 shows the system call traces at thread level. Each line—except the last two—represents a thread that can be selected from a list. In the last line, a barchart shows the distribution of systems calls over time. The number of all system calls from all threads overlapping the respective time interval is mapped to the height of the associated blue bar of the bar chart. On mouse over, the exact number of system calls overlapping each time interval is shown. Above this bar chart, the timeline is shown.

Figure 5(a) shows the coarsest granularity where the number of time intervals depends on the available screen space. Similar to the "Process View", a cell represents all system calls for a specific thread (row) overlapping a specific time interval (column). The saturation of the cell represents this number of system calls in relation to

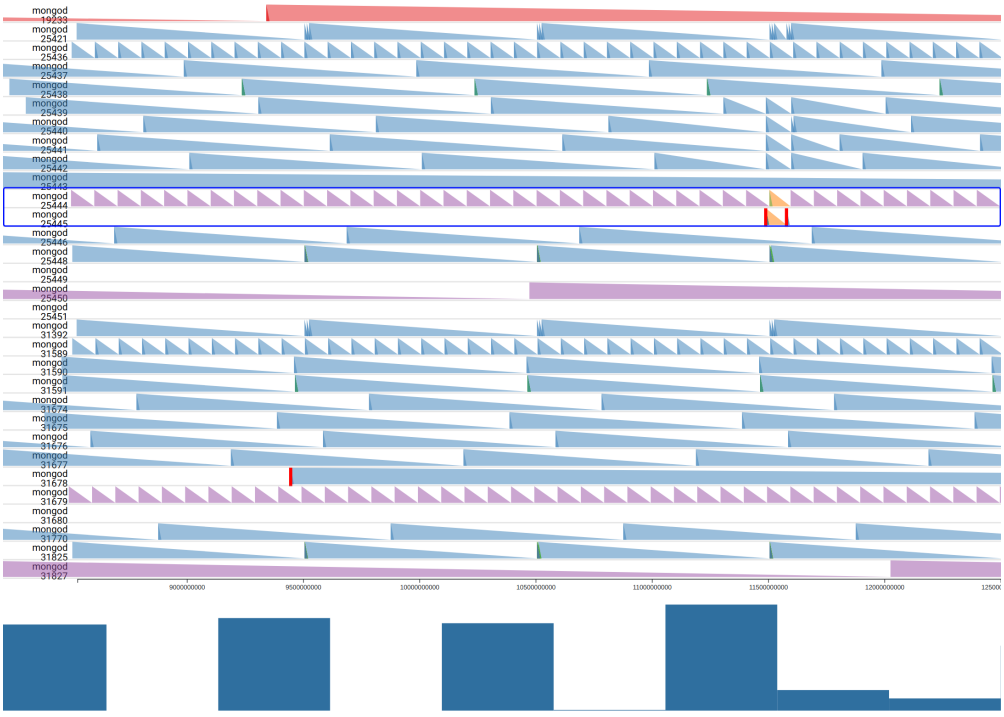
- the total number of system calls of the thread (used in Figure 5)
- the maximal number of system calls overlapping a time step over all threads
- the sum of system calls overlapping a time step over all threads
- the sum of system calls over all threads

Interaction allows to obtain more details. Zooming-in using the mouse allows to decrease the length of the time intervals. Thus, fewer, shorter time intervals are shown. These shorter time intervals are represented by cells as long as they contain more system calls than can be displayed. Otherwise, triangles and bars representing individual system calls are used. Each triangle starts at the beginning of the system call and ends at its termination. The triangle's color represents the system call category:

ISBN 978-80-86943-37-4



(a) Coarsest view showing the complete time interval selected. The saturation of each cell represents the number of system calls that are active in that interval: the higher the saturation the more system calls are contained in this thread and time interval.



(b) Finest view showing individual system calls, only. Triangles represent the system calls and their durations (called 'latency" by sysdig) from the start event to the final event of each system call. The triangle's color represents its system call category: *red*: net(work), *blue*: inter-process communication (IPC), *violet*: sleeping, *orange*: unknown (system call type: fdatasync). Single vertical red lines represent system calls with zero duration (e.g. in the marked lines, before and after the yellow system call). The irregularities in the idle threads (see marked lines) can easily be identified.

Figure 5: Visualization of the threads showing only aggregated system calls (a) and only individual system calls (b). In the last line, a barchart shows the distribution of systems calls over time. The number of all system calls from all threads overlapping the respective time interval is mapped to the height of the associated blue bar of the bar chart. On mouse over, the exact number of system calls overlapping each time interval is shown. Above this bar chart, the timeline is shown.

ISBN 978-80-86943-37-4

- *red*: net
- *blue*: inter-process communication (IPC)
- *violet*: sleeping
- *orange*: unknown (system call type: fdatasync).

A red bar represents a system call of zero duration. The finest level displaying individual system calls, only, is shown in Figure 5(b) (most detailed view).

This flexibility allows to generate as many intermediate levels between the coarsest (least detailed) and the finest (most detailed) level as necessary to provide an acceptable number of elements (bins or individual system calls).

4.4.2 Interpretation

The temporal pattern that can be observed in this visualization fosters recognizing normal as well as possibly malicious behavior. Again, a thorough understanding about the computer, its users, and the processes to be expected is needed for judging whether or not a certain visual pattern is suspected to be malicious or not. Therefore, the system calls that were issued are analyzed regarding their amount and their sequence. The detection of malicious behavior is one the one hand fosters by the side-by-side display of a number of processes performing the same task (see Figure 5 (b) line 4 to 9) such that processes with a divergent amount or sequence of system calls can be spotted. On the other hand, a sudden change in the behavior of a single process might point to a malicious irregularity. In Figure 5, the irregularities in the idle threads (lines in the blue bordered box, Figure 5) are an example of this type of anomaly.

5 CONCLUSION

We propose SyCaT-Vis for the visualization-based analysis of system call traces. Currently, three views are provided fostering understanding of the system's behavior: an *overview* showing the processes and their context (context view), an *intermediate view* showing details of selected processes (process view), and a *detailed view* showing details of selected threads of one or more processes (thread view, several levels of detail). Being able to configure and to interact with these flexible visualizations enables the security researcher to adapt them to her needs and to focus on the crucial parts for understanding whether the system's behavior is normal or not.

6 ACKNOWLEDGMENTS

This work was partially funded by the German Federal Ministry of Education and Research (BMBF) within the project Explicit Privacy-Preserving Host Intrusion Detection System EXPLOIDS (BMBF 16KIS0522K).

7 REFERENCES

[1] S. Coull, J. Branch, B. Szymanski, and E. Breimer. Intrusion detection: a bioinformatics approach. In *Proc. 19th Annual Comp. Security Appl. Conference*, pp. 24–33, Dec 2003. doi: 10.1109/CSAC.2003.1254307

[2] Draios Inc. *Csysdig Overview*, March 2017.

[3] Draios Inc. *Sysdig Overview*, March 2017.

[4] S. Forrest, S. A. Hofmeyr, and A. Somayaji. Computer Immunology. *Commun. ACM*, 40(10):88–96, Oct. 1997. doi: 10.1145/262793.262811

[5] S. Forrest, S. A. Hofmeyr, and A. Somayaji. The Evolution of System-Call Monitoring. In *Proceedings of the Annual Comp. Security Appl. Conference*, ACSAC '08, pp. 418–430. IEEE Computer Society, Washington, DC, USA, 2008. doi: 10.1109/ACSAC.2008.54

[6] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for Unix processes. In *Proceedings 1996 IEEE Symposium on Security and Privacy*, pp. 120–128, May 1996. doi: 10.1109/SECPRI.1996.502675

[7] A. K. Ghosh and A. Schwartzbard. A Study in Using Neural Networks for Anomaly and Misuse Detection. In *Proc. of the 8th Conference on USENIX Security Symposium - Volume 8*, SSYM'99, pp. 12–12. USENIX Association, Berkeley, CA, USA, 1999.

[8] Y. Liao and V. R. Vemuri. Using Text Categorization Techniques for Intrusion Detection. In *Proceedings of the 11th USENIX Security Symposium*, pp. 51–59. USENIX Association, Berkeley, CA, USA, 2002.

[9] G. Mazeroff, V. De, C. Jens, G. Michael, and G. Thomason. Probabilistic Trees and Automata for Application Behavior Modeling. In *41st ACM Southeast Regional Conference Proceedings*, pp. 435–440, 2003.

[10] R. Sekar, M. Bendre, D. Dhurjati, and P. Bollineni. A fast automaton-based method for detecting anomalous program behaviors. In *Proceedings 2001 IEEE Symposium on Security and Privacy. S P 2001*, pp. 144–155, 2001. doi: 10.1109/SECPRI.2001.924295

[11] B. Shneiderman. Tree Visualization with Treemaps: 2-d Space-filling Approach. *ACM Trans. Graph.*, 11(1):92–99, Jan. 1992. doi: 10.1145/102377.115768

[12] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proc. IEEE Symp. on Visual Languages*, pp. 336–343, 1996. doi: 10.1109/VL.1996.545307

[13] X. Shu, D. D. Yao, N. Ramakrishnan, and

 ISBN 978-80-86943-37-4

T. Jaeger. Long-Span Program Behavior Modeling and Attack Detection. *ACM Trans. Priv. Secur.*, 20(4):12:1–12:28, Sept. 2017. doi: 10.1145/3105761

[14] G. Tandon, P. Chan, and D. Mitra. MORPHEUS: Motif Oriented Representations to Purge Hostile Events from Unlabeled Sequences. In *Proc. of the ACM Workshop on Visualization and Data Mining for Computer Security*, VizSEC/DMSEC '04, pp. 16–25. ACM, New York, NY, USA, 2004. doi: 10.1145/1029208.1029212

[15] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: alternative data models. In *Proc. of the IEEE Symp. on Security and Privacy (Cat. No.99CB36344)*, pp. 133–145, 1999. doi: 10.1109/SECPRI.1999.766910

[16] Y. Wu, R. H. C. Yap, and F. Halim. Visualizing Windows System Traces. In *Proceedings of the 5th International Symposium on Software Visualization*, SOFTVIS '10, pp. 123–132. ACM, New York, NY, USA, 2010. doi: 10.1145/1879211.1879231

[17] F. Yu, C. Xu, Y. Shen, J.-Y. An, and L.-F. Zhang. Intrusion detection based on system call finite-state automation machine. In *IEEE International Conference on Industrial Technology*, pp. 63–68, Dec 2005. doi: 10.1109/ICIT.2005.1600611

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

Evaluation of 4D Light Field Compression Methods

David Barina Tomas Chlubna Marek Solony Drahomir Dlabaja Pavel Zemcik

Centre of Excellence IT4Innovations
Faculty of Information Technology
Brno University of Technology
Bozetechova 1/2, Brno
Czech Republic
{ibarina,ichlubna,isolony,zemcik}@fit.vutbr.cz

Abstract

Light field data records the amount of light at multiple points in space, captured e.g. by an array of cameras or by a light-field camera that uses microlenses. Since the storage and transmission requirements for such data are tremendous, compression techniques for light fields are gaining momentum in recent years. Although plenty of efficient compression formats do exist for still and moving images, only a little research on the impact of these methods on light field imagery is performed. In this paper, we evaluate the impact of state-of-the-art image and video compression methods on quality of images rendered from light field data. The methods include recent video compression standards, especially AV1 and XVC finalised in 2018. To fully exploit the potential of common image compression methods on four-dimensional light field imagery, we have extended these methods into three and four dimensions. In this paper, we show that the four-dimensional light field data can be compressed much more than independent still images while maintaining the same visual quality of a perceived picture. We gradually compare the compression performance of all image and video compression methods, and eventually answer the question, "What is the best compression method for light field data?".

Keywords

Light field, Plenoptic imaging, Lossy compression, Image refocusing

1 INTRODUCTION

To describe a three-dimensional scene from any possible viewing position at any viewing angle, one could define a plenoptic function $P(x,y,z,\phi,\psi)$, where the (x,y,z) is the position and (ϕ,ψ) is a viewing angle (in spherical coordinates) of a camera. Figure 1 shows the situation. The value of the P is color. The definition can be further extended with t (time) to describe a dynamic scene.

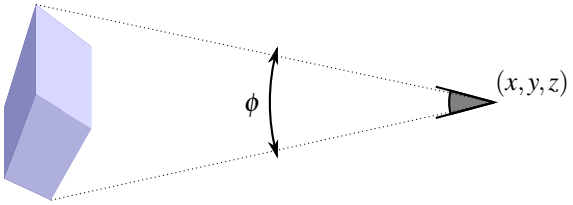


Figure 1: Plenoptic capture of a scene from a single viewing position. For simplicity, the range of viewing angles is indicated for one spherical coordinate.

Our interest here is to describe the scene by capturing either via an array of cameras or by a single compact sensor preceded by microlenses. In this case, the aperture is modeled by a grid of views (cameras) located on a two-dimensional plane. This situation is shown in Figure 2, where the baseline between individual views from the grid is described by the distance d. This representation is often referred to as 4D light field (LF) since we deal with the light field function, L, sampled in four dimensions, (k,l,m,n), where the (m,n) are pixel coordinates, and (k,l) are indices of a sub-aperture image. Light fields acquired by the single compact sensor have limited support for the viewing angle. Light fields based on the array of cameras offer larger viewing angles at the cost of missing information in between the cameras. In practice, the number of views located on

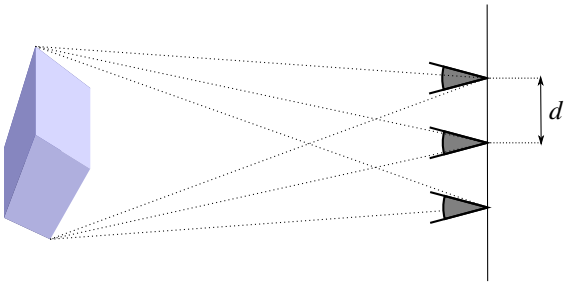


Figure 2: 4D light field capture via an array of cameras.

Figure 3: Dataset used in this paper. From top left to bottom right: Black Fence, Chessboard, Lego Bulldozer, and Palais du Luxembourg.

the two-dimensional plane ranges from a couple of units to several hundred. Considering increasing resolution sensors, it is no surprise that the light field data reach huge sizes. As an example, consider "Lego Bulldozer" light field (Figure 3) taken from the Stanford Light Field Archive. The light field is captured using a 17×17 grid of cameras having image resolution 1536×1152 (rectified and cropped). The uncompressed size easily exceeds a gigabyte. For light field videos, storage and transmission requirements are enormous.

Several methods to compress 4D light fields have been recently proposed. Some of them attempt to compress directly the data from sensors preceded by microlenses (lenslet image). Other compresses the resulting 4D light field. In this paper, we focus only on the latter ones. We compare various state-of-the-art compression methods applicable to 4D light field data. These methods include recent video compression standards, especially AV1 (validated in June 2018), and XVC (version released in July 2018). In order to evaluate the comparison, we refocus the original and decompressed light field. The evaluation is then carried out using the PSNR as a full-reference quality assessment metric.

The remainder of the paper is organized as follows. Section 2 reviews related work and compression methods. Section 3 presents our experiments in detail, and discusses the results. Section 4 concludes the paper.

2 RELATED WORK

The individual views from a light field are usually never displayed. Therefore, it is not very meaningful to compare the original and decompressed light field directly, even though such methodology is usual to asses a single view compression performance. For this reason, we adopt the compression performance assessment methodology for multi-focus rendering from [2]. This methodology basically lies in assessing the quality of the rendered views for multiple focal points. The rendered views are obtained by combining pixels from different 4D light field views for various focal planes. The average distortion is computed as the mean of the PSNR for multiple rendered focal plane views. This situation is shown in Figure 4. Note that the PSNR is computed from the MSE over all three color components.

The 4D light field comprises a two-dimensional grid of two-dimensional views. The baseline between individual views ranges from a few millimeters (microlenses) to several centimeters (camera array). It is, therefore, natural to expect a high similarity of views adjacent in any of two grid directions. This similarity opens the door to understanding the 4D light field data as a video sequence navigating between the viewpoints. Another possible point of view is to see the 4D light field as the three- or directly four-dimensional body. The above approaches can also be reflected in light field compression by using either an image, video, volumetric, or four-dimensional coding system. Although other approaches (like 3D video) are also possible, we are not aware of generally available coding systems for such cases.

In recent years, several papers compared and evaluated the compression performance of various approaches on light field imagery. The authors of [2] evaluated the performance of the main image coding standards, JPEG, JPEG 2000, H.264/AVC intra profile, and H.265/HEVC intra profile. The "intra" suffix refers to the fact the individual views were compressed independently (intra profile). The video coding approaches were not evaluated. As could be expected, the H.265/HEVC intra profile proved to be the most efficient compression method. In [17], the authors compared the compression performance of three strategies using the H.265/HEVC. Their first strategy performs compression directly on the lenslet image. Another strategy arranges 4D LF views a pseudo-temporal sequence in spiral order and subsequently compressed it. The last strategy compresses a subset of lenslet images through the transformation to 4D LF. Their results show that coding 4D LF leads to better performance when compared to coding lenslet images directly. The authors of [6] compared the performance of JPEG, JPEG 2000, and SPIHT directly on lenslet images. The comparison was performed using the same methodology as in this paper. As could be expected, the JPEG 2000 exhibits the best compression performance. In [16], the authors proposed to rearrange 4D LF views into tiles of a big rectangular image. This image is then compressed using the JPEG 2000 coder. The proposed scheme was compared against standard image coding algorithms, namely the JPEG 2000 and JPEG XR. It is, however, unclear how these standard coding algorithms were exactly applied to the 4D light field data. In [1], the author rearranges the 4D light field into a three-dimensional body. The three-dimensional volume is then encoded using the 3D DCT scheme on $8 \times 8 \times 8$ blocks, similarly as in the JPEG coding system.

ISBN 978-80-86943-37-4

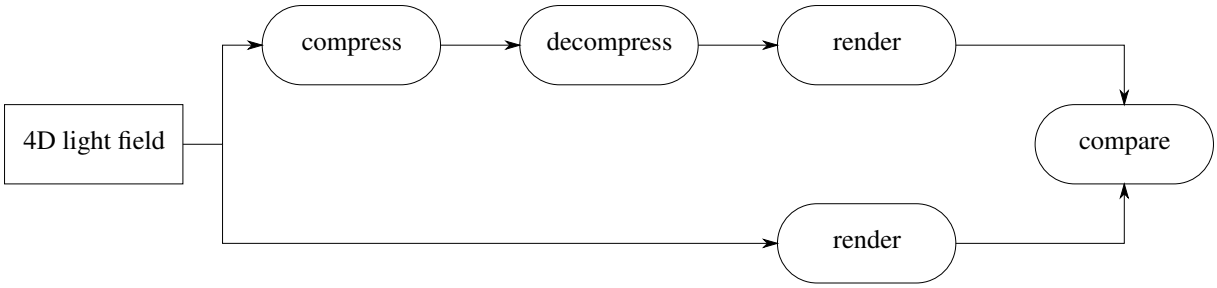


Figure 4: Data flow diagram of the compression performance assessment methodology used in this paper.

Besides conventional coding methods, also an alternative approach [3] exists that uses deep learning to estimate the 2D view from the sparse sets of 4D views. Another approach [4] proposes own sparse coding scheme for the entire 4D LF based on several optimized key views. The method in [9] decomposes the 4D light field into homography parameters and residual matrix. The matrix is then factored as the product of a matrix containing k basis vectors and a smaller matrix of coefficients. The basis vectors are then encoded using the H.265/HEVC intra profile. In [11, 12], the authors propose a hierarchical coding structure for 4D light fields. The 4D LF is decomposed into multiple views and then organized them into a coding structure according to the spatial coordinates. All of the views are encoded hierarchically. The scheme is implemented in the reference H.265/HEVC software. In [5], the authors propose a coding scheme that splits the 4D light field into several central views and remaining adjacent views. The adjacent views are subtracted from the central views, and both groups are then encoded using H.265/HEVC coder. The authors of [13, 14] feed the 4D LF into the H.265/HEVC exploiting the inter prediction mode for individual LF views. Finally, tremendous attentions have also been focused on convolutional neural network based compression approaches [7, 8].

From the above, it can be seen that the JPEG 2000 and especially the H.265/HEVC coding schemes are quite popular. In this paper, we compare the compression performance of the main state-of-the-art lossy compression methods. These methods can be divided into four groups according to the way they process the 4D LF data. The first group covers the following image coding methods—the JPEG and JPEG 2000. In the literature [10], this kind of methods is sometimes referred to as the self-similarity based methods. The second group comprises video coding methods: H.265/HEVC, AV1, VP9, and XVC. In the literature, these methods are referred to as the pseudo sequence based methods. The third group extends the image coding methods into three dimensions. This group consists of JPEG 3D (our own implementation) and JPEG 2000 3D (Part 10, JP3D). Notice that the JPEG 3D refers to a volume image rather than a pair of stereoscopic images. The fourth group extends the image coding methods into four dimensions. However, only one method in this group exists, JPEG 4D (our own implementation). To evaluate the above methods, we use the following list of encoders: OpenJPEG, x265, libaom (AV1 Codec Library), libvpx (VP8/VP9 Codec SDK), xvc codec, and our own implementation of the JPEG method.

Since our comparison also deals with the latest video compression standards, we consider it appropriate to present their short description here. The H.265/HEVC (High Efficiency Video Coding, MPEG-H Part 2) is a video compression standard designed as a successor to the widely used H.264/AVC (MPEG-4 Part 10). The standard was published in June 2013. The AV1 (AOMedia Video 1) is an open video coding format standardized in June 2018. It succeeds the VP9 video coding format developed by Google. According to [18], the AV1 outperforms the H.265/HEVC by 17 %, and VP9 by 13 % over a wide range of bitrate/resolutions. The XVC is a video coding format with a strong focus on low bitrate streaming applications. The official website claims that the codec outperforms the AV1, H.265/HEVC, and VP9.

3 EVALUATION

This section presents our dataset, multi-focus rendering method, experiments conducted on this dataset using the above methodology, and the results we achieved.

Our dataset consists of four 4D light fields based on two types of capturing devices. Two of the light fields were captured using Lytro Illum B01 plenoptic camera and the other two using conventional cameras. The first conventional camera light field was captured using a multi-camera array, and the other one using simple motorized gantry equipped with Canon Digital Rebel XTi camera. Corresponding resolutions and adjacent image disparity ranges are listed in Table 1. The value in the last column describes the pixel difference in the location of the same 3D object projected to images captured by a camera or computed from a lenslet image in the case of Lytro. The range is narrow (ca. -1 to $+1$ pixel) for the densely-sampled light field (Lytro) and wide (ca. 40 to 90 pixels) for the images captured by camera array. These values correlate to the focal length and camera baseline (distance between camera centers). For convenience, the central view for each light field is shown in Figure 3.

ISBN 978-80-86943-37-4

description	source	resolution	disparity
Black Fence	EPFL Light-field data set	$15 \times 15 \times 625 \times 434$	-1 to 1
Chessboard	Saarland University	$8 \times 8 \times 1920 \times 1080$	40 to 90
Lego Bulldozer	Stanford Computer Graphics Laboratory	$17 \times 17 \times 1536 \times 1152$	-1 to 7
Palais du Luxembourg	EPFL Light-field data set	$15 \times 15 \times 625 \times 434$	-1 to 1

Table 1: Dataset used in this paper. The first and last light field are taken using a plenoptic camera; the Chessboard is captured using a camera array; the Lego Bulldozer is captured using a motorized gantry holding a camera. The adjacent image disparity range (last column) is given in pixels.

The digital refocus of the images at the virtual focal plane is achieved using shift-sum algorithm [15]. This algorithm shifts the sub-aperture images (views) according to camera baseline with respect to the reference frame and accumulates the corresponding pixel values. The refocused image will be an average of the transformed images. The computation of the pixel value at point (m,n) of the refocused image E_d is given by the equation

$$E_d(m,n) = \frac{1}{N}\sum_{k,l} L(k,l,m+\alpha k, n+\alpha l), \qquad (1)$$

where N is the number of summed images, α is the distance of the synthetic plane from the main lens, k and l are indices of a sub-aperture image of the light field representation, and αk and αl are the shift parameters with respect to the reference frame. We performed a linear interpolation in the last two 4D dimensions to convert the sampled light field function into a continuous one.

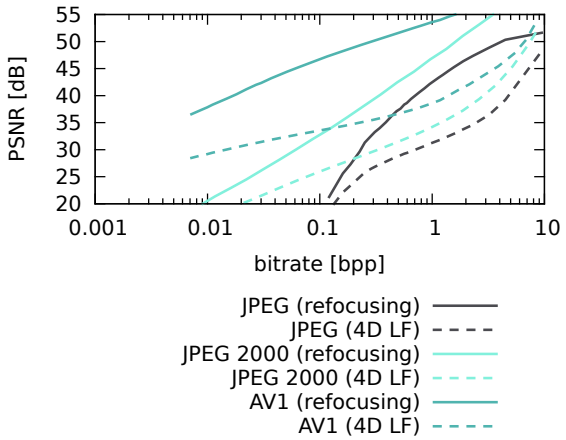


Figure 5: Experiment 0. The difference in the quality assessment using the 4D light field directly vs. using images rendered at virtual focal planes. Illustratively shown on the Black Fence light field.

Experiment 0

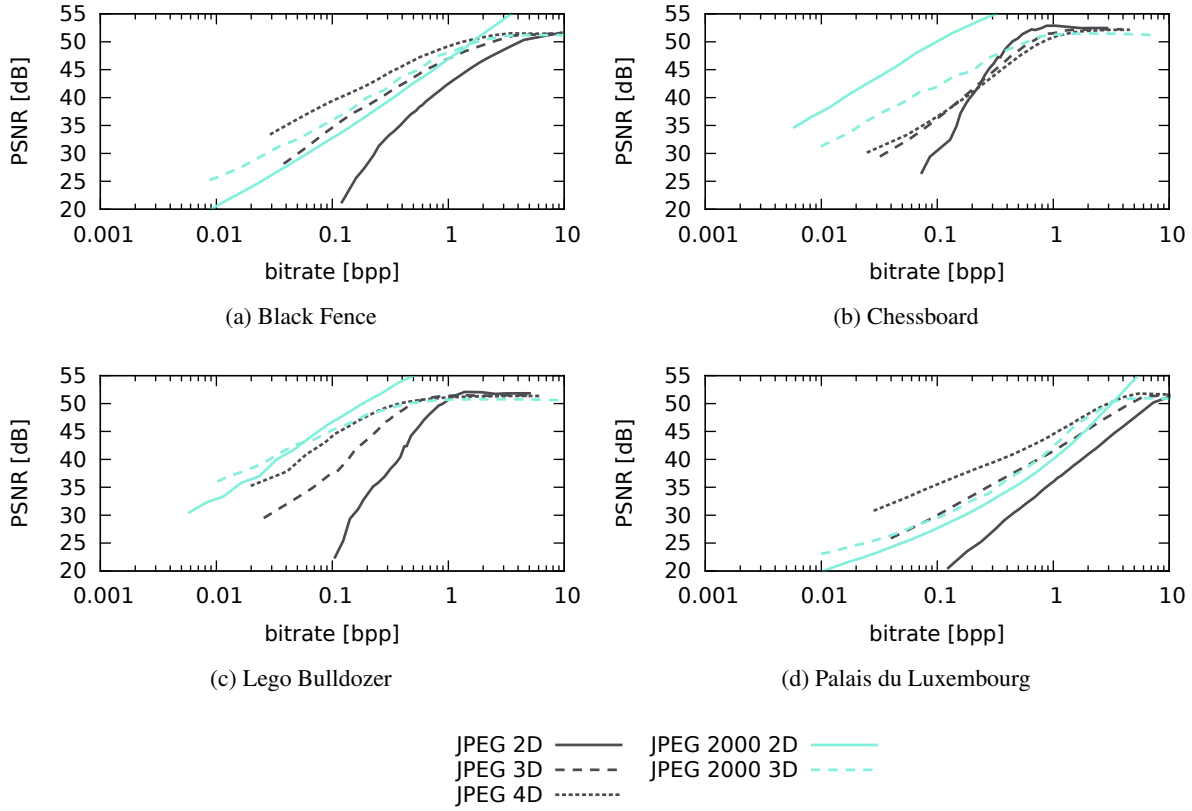
Before we start, the reader might wonder whether it is really necessary to assess the image quality on views rendered for multiple focal points rather than the original views (i.e. compare the original and decompressed LF directly). A quick experiment reveals that a big difference exists between the former and the latter (see Figure 5). This difference is about 10 decibels in the PSNR, depending on the bitrate and compression method. This can be explained by the fact that any pixel in the rendered view is a sum of pixels from the 4D LF so that this sum all together suppresses compression artifacts. In other words, we can afford to compress the 4D light fields much more than independent images, while maintaining the same visual quality of a screened picture.

Experiment 1

As seen from the previous section, most current LF compression approaches handle either 2D data or their sequence (video compression). Compression of 4D LF images is still a relatively unexplored area of research. Since 4D LF are sequences of 2D images (views), the 2D compression methods may be used to code the views independently. However, such methods fail to exploit pixel correlations in all four dimensions. Similar reasoning can be used for 3D methods. In our first experiment, we were interested in examining the effects of LF compression in three and four dimensions. To evaluate the compression performance fairly, identical compression method must be used for the 2D, 3D, and 4D case. Thus, we have created a custom implementation of the JPEG compression method with the ability to process either the 2D, 3D, or 4D data. Additionally, we are aware of the existence of the JPEG 2000 standard, with the ability to compress the 2D and 3D data in the same manner. Unfortunately, the JPEG 2000 does not deal with the 4D images. Since the similarity of adjacent pixels in the third and four dimensions strongly depends on the camera baseline, different results can be expected depending on the baseline distance. The result of this experiment is shown in Figure 6. In each panel, the horizontal axis shows the bitrate (bits per pixel), whereas the vertical axis shows the mean of the PSNR for multiple rendered focal plane views.

On light fields with a small baseline (Black Fence and Palais du Luxembourg), both 3D compression methods clearly outperform their 2D counterparts over a

ISBN 978-80-86943-37-4



(a) Black Fence

(b) Chessboard

(c) Lego Bulldozer

(d) Palais du Luxembourg

Figure 6: Experiment 1. Comparison of image compression methods against their extensions into three and four dimensions.

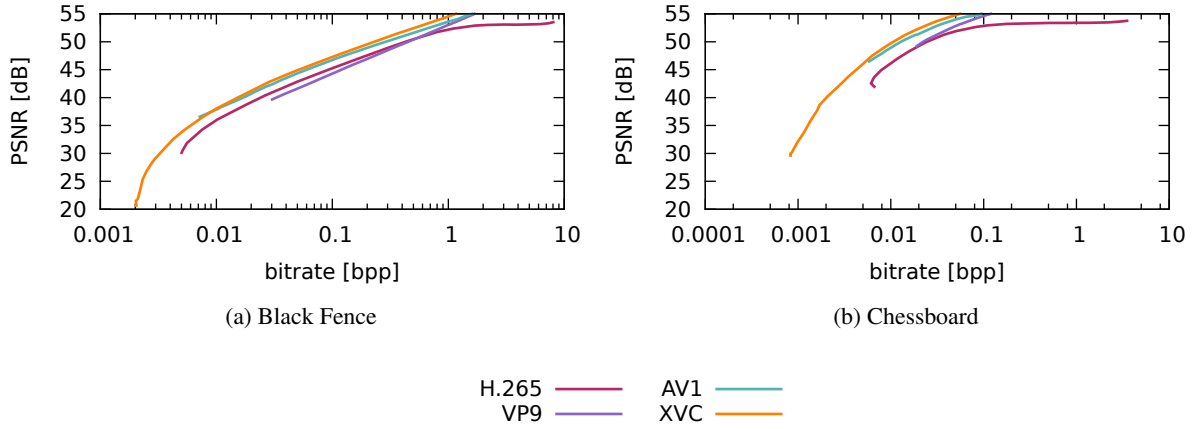
whole range of bitrates. Similarly, the 4D JPEG method clearly outperforms its 3D counterpart. This is not so surprising because pixels at the same spatial position in adjacent views are strongly correlated. However, the situation changes with increasing baseline. With increasing baseline (Lego Bulldozer and Chessboard), adjacent views are less and less similar, which results in higher amplitudes of the underlying transform coefficients. Consequently, the tide is turning in favor of the less-dimensional compression methods. Considering the JPEG method, the Lego Bulldozer is a special case because it contains large areas of blackness (black pixels). It turns out that it is more efficient to compress these solid areas at once using a single 4D block than using multiple 3D blocks. Similarly, it is more efficient to use a single 3D block than multiple 2D blocks.

Experiment 2

The second thing to notice in the previous section is the employment of the video compression standards. Upon this, a question arises: whether it be better to compress the 4D light fields as a sequence of 2D frames, or as multi-dimensional body. We, therefore, measured the performance of all the above-mentioned video compression standards. The results can be seen in Figure 7. This time results only for two light fields are shown for brevity. We have, however, got similar results for the other two. Interestingly, the XVC codec has really shown better compression performance than HEVC and AV1, as claimed by the official website.

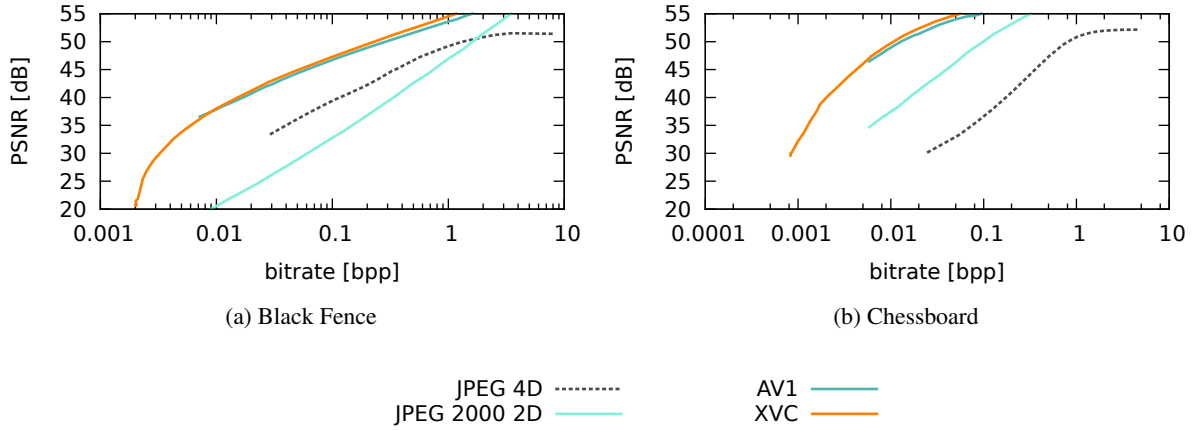
To answer the question, "What is the best compression method for LF data?", we have further compared these results with the best-performing methods from Experiment 1. The overall comparison is shown in Figure 8. Interestingly, video compression methods perform better than all image compression methods, even better than their 3D and 4D extensions.

ISBN 978-80-86943-37-4



(a) Black Fence (b) Chessboard

Figure 7: Experiment 2. Performance of video compression methods. The XVC and AV1 clearly overcome the older standards.



(a) Black Fence (b) Chessboard

Figure 8: Overall performance of the best compression methods. Video compression methods perform better than all image compression methods.

4 CONCLUSIONS

The purpose of our work was to evaluate the current methods suitable for lossy compression of 4D light fields. Since the light field is basically a collection of images (views), image compression methods are often the first choice, when it comes to the need for compression. It turns out that the methods handling the 4D light fields directly in four (or three) dimensions are able to achieve better compression results than common image compression algorithms. This is, however, dependent on a baseline between neighboring views. For large baselines (e.g., camera arrays), the common image compression methods come handy.

We have also evaluated the performance of video compression methods. The underrated XVC compression format demonstrated superior performance, closely followed by the AV1. This confirms that the latest video compression standards offer better performance than their predecessors. Eventually, it turns out that these video compression methods perform better than the image compression methods (including their 3D and 4D extensions).

Acknowledgements

This work has been supported by the Technology Agency of the Czech Republic (TA CR) Competence Centres project V3C – Visual Computing Competence Center (no. TE01020415), the Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II) project IT4Innovations excellence in science (LQ1602), and the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 780470.

REFERENCES

[1] Aggoun, A. A 3D DCT compression algorithm for omnidirectional integral images. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, 2006. doi: 10.1109/ICASSP.2006.1660393.

[2] Alves, G., Pereira, F., and da Silva, E. A. Light field imaging coding: Performance assessment methodology and standards benchmarking. In *IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6, 2016. doi: 10.1109/ICMEW.2016.7574774.

[3] Bakir, N., Hamidouche, W., Deforges, O., Samrouth, K., and Khalil, M. Light field image compression based on convolutional neural networks and linear approximation. In *IEEE International Conference on Image Processing (ICIP)*, pages 1128–1132, 2018. doi: 10.1109/ICIP.2018.8451597.

[4] Chen, J., Hou, J., and Chau, L.-P. Light field compression with disparity-guided sparse coding based on structural key views. *IEEE Transactions on Image Processing*, 27(1):314–324, 2018. ISSN 1057-7149. doi: 10.1109/TIP.2017.2750413.

[5] Han, H., Xin, J., and Dai, Q. Plenoptic image compression via simplified subaperture projection. In *Advances in Multimedia Information Processing (PCM)*, pages 274–284, 2018. ISBN 978-3-030-00767-6.

[6] Higa, R. S., Chavez, R. F. L., Leite, R. B., Arthur, R., and Iano, Y. Plenoptic image compression comparison between JPEG, JPEG2000 and SPIHT. *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT)*, 3(6), 2013.

[7] Hou, J., Chen, J., and Chau, L.-P. Light field image compression based on bi-level view compensation with rate-distortion optimization. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(2):517–530, 2019. ISSN 1051-8215. doi: 10.1109/TCSVT.2018.2802943.

[8] Jia, C., Zhang, X., Wang, S., Wang, S., Pu, S., and Ma, S. Light field image compression using generative adversarial network based view synthesis. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2018. ISSN 2156-3357. doi: 10.1109/JETCAS.2018.2886642.

[9] Jiang, X., Pendu, M. L., Farrugia, R. A., and Guillemot, C. Light field compression with homography-based low-rank approximation. *IEEE Journal of Selected Topics in Signal Processing*, 11(7):1132–1145, 2017. ISSN 1932-4553. doi: 10.1109/JSTSP.2017.2747078.

[10] Li, L. and Li, Z. *Light Field Image Compression*, pages 547–570. Springer, 2018. ISBN 978-3-319-95504-9. doi: 10.1007/978-3-319-95504-9_24.

[11] Li, L., Li, Z., Li, B., Liu, D., and Li, H. Pseudo sequence based 2-D hierarchical coding structure for light-field image compression. In *Data Compression Conference (DCC)*, pages 131–140, 2017. doi: 10.1109/DCC.2017.10.

[12] Li, L., Li, Z., Li, B., Liu, D., and Li, H. Pseudo-sequence-based 2-D hierarchical coding structure for light-field image compression. *IEEE Journal of Selected Topics in Signal Processing*, 11(7):1107–1119, 2017. ISSN 1932-4553. doi: 10.1109/JSTSP.2017.2725198.

[13] Li, Y., Sjostrom, M., Olsson, R., and Jennehag, U. Efficient intra prediction scheme for light field image compression. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 539–543, 2014. doi: 10.1109/ICASSP.2014.6853654.

[14] Li, Y., Sjostrom, M., Olsson, R., and Jennehag, U. Coding of focused plenoptic contents by displacement intra prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(7):1308–1319, 2016. ISSN 1051-8215. doi: 10.1109/TCSVT.2015.2450333.

[15] Ng, R., Levoy, M., Bredif, M., Duval, G., Horowitz, M., and Hanrahan, P. Light field photography with a hand-held plenoptic camera. Technical report, 2005. Stanford Tech Report CTSR 2005-02.

[16] Perra, C. and Giusto, D. JPEG 2000 compression of unfocused light field images based on lenslet array slicing. In *IEEE International Conference on Consumer Electronics (ICCE)*, pages 27–28, 2017. doi: 10.1109/ICCE.2017.7889217.

[17] Viola, I., Rerabek, M., and Ebrahimi, T. Comparison and evaluation of light field image coding approaches. *IEEE Journal of Selected Topics in Signal Processing*, 11(7):1092–1106, 2017. ISSN 1932-4553. doi: 10.1109/JSTSP.2017.2740167.

[18] Zabrovskiy, A., Feldmann, C., and Timmerer, C. Multi-codec DASH dataset. In *ACM Multimedia Systems Conference*, MMSys '18, pages 438–443. ACM, 2018. ISBN 978-1-4503-5192-8. doi: 10.1145/3204949.3208140.

ISBN 978-80-86943-37-4

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

Analyzing pixel spread correlation with lenticular lens efficiency on multi user VR displays

Juan Sebastian Munoz-Arango
Emerging Analytics Center
University of Arkansas Little Rock
72204, Little Rock, AR, USA
jsmunozaran@ualr.edu

Dirk Reiners
Emerging Analytics Center
University of Arkansas Little Rock
72204, Little Rock, AR, USA
dpreiners@ualr.edu

Carolina Cruz-Neira
Emerging Analytics Center
University of Arkansas Little Rock
72204, Little Rock, AR, USA
cxcruz@ualr.edu

ABSTRACT

One of the all time issues with Virtual Reality systems regardless if they are head-mounted or projection based is that they can only provide perspective correctness to one user. Such limitation affects collaborative work which is nowadays the standard anywhere. One of the approaches for generating different perspective correct images to several users is through optical routing. This approach relies on bending the light to generate perspective correct images to the engaged users. Lenticular lenses can bend the light to be able to generate perspective correct images for several users depending on their positions. On this paper we present an analysis that lets us understand the pixel spread correlation with lenticular lens efficiency on multi user VR displays.

Keywords

Multi user displays, Virtual Reality, Simulation, lenticular lens

1 INTRODUCTION

Different approaches have been proposed across the years for interacting through Virtual Reality (VR), head mounted displays (HMDs), projection-based virtual reality systems, volumetric screens and several different devices aid on these tasks.

When the person who is interacting in VR is by himself, HMDs work well to some extent. Unfortunately, working alone is rarely the case nowadays in any field. Several attempts have been proposed to work cooperatively with HMDs [1] but unfortunately having weight on one's head not only exhausts users but also hinder their eyesight when used on prolonged sessions. Although some sessions can be short, another issue that raises is that HMDs occlude one's face, hence making facial expressions less visible to other participants removing important nonverbal communication channels.

Normally, when a group of experts get together, they discuss and gesture around a common dataset hoping to achieve a consensus. This engagement is persistent across disciplines and begs for a VR system that accomodates small groups of people working together with correct perspective point of views. Different studies [2] [3] back up the perspective correctness requirement demonstrating that collaboration times get significantly longer when participants dont have a correct perspective point of view compared to a correct one.

An interesting classification for different approaches when multiplexing images from a single display was presented by Mark bolas et al in [4]. Here, they introduce a "solution framework" of four categories for multiplexing images: Spatial Barriers, Optical Filtering, Optical Routing and Time Multiplexing.

Lenticular lenses fall in the optical routing approach and are widely used for lenticular printing, 3D TV and lenticular screens among other uses [5][6][7]. These lenses can also used for multiplexing views from a single screen. It has been shown that the lens specs that maximizes the minimum visible unique pixels and minimizes ghosting is a lenticular lens that has a 50% radius increase from the minimum possible radius, covers 18 to 21 pixels per lenticule and has the smallest possible substrate thickness in [8].

Pixel spread is also an important factor that also contributes to ghosting between users. On this paper we are going to present and discuss the correlation factor of pixel spread with lenticular lens performance for multi user displays.

2 BACKGROUND

2.1 Lenticular Lens Overview

A lenticular lens can be thought as a set of cylinders (lenticules) placed next to each other overlapping with one side flat and the other side with the protubing lenses (fig 1), these lenses are designed so that when viewed from different angles different slices of an image are displayed (fig 2).

Lenticular printing is one of the most prominent examples of lenticular lens usage. Here, lenses are employed to give illusion of depth or to make images that change depending on the angle the print is looked from.

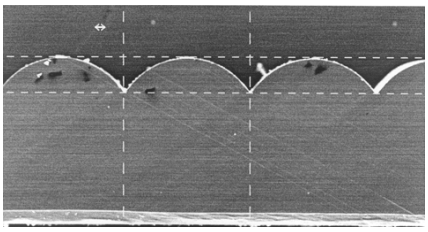


Figure 1: Micro profile of a lenticular lens.

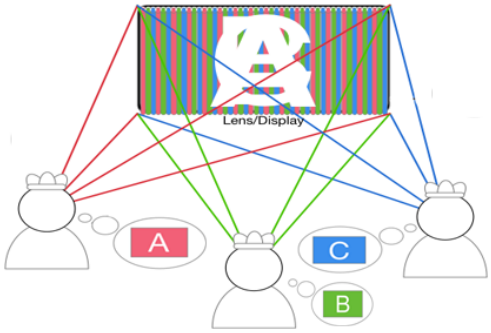


Figure 2: Image looked from different angles in a lenticular lens

Depending on how pronounced the lenticules are engraved in the sheet, different types of effects can be achieved (fig 3). These effects can be just about anything one can do with video; some of these effects include morphing effects, animations, flipping and 3D effects [9] [10].

Figure 3: Flip to 3D lenticular lenses.

- **Morph effect:** This is commonly used to create the illusion of transformation (fig. 4 A).
- **Animation effect:** Generates illusion of motion from a set of sequential images (fig. 4 B).
- **Flip effect:** A dramatic swap of two images occurs vanishing and then reappearing from one to another (fig. 4 C).
- **3D effect:** Provides an illusion of depth and perspective by layering objects within images (fig. 4D).

Lenticular lenses are classified by the number of lenticules that can be fit in an inch (Lenses per inch (LPI)), the higher the LPI the smaller each lenticule is. Manufacturers normally offer lenses in 10, 15, 20, 30, 50, 60, 75, 100LPI lenses; however, with enough funds one can get a custom lens manufactured with specific characteristics.

Following the same principle of how lenticular printing works, lenticular lenses can also be used for multiplexing different images to different users in VR under an optical routing approach.

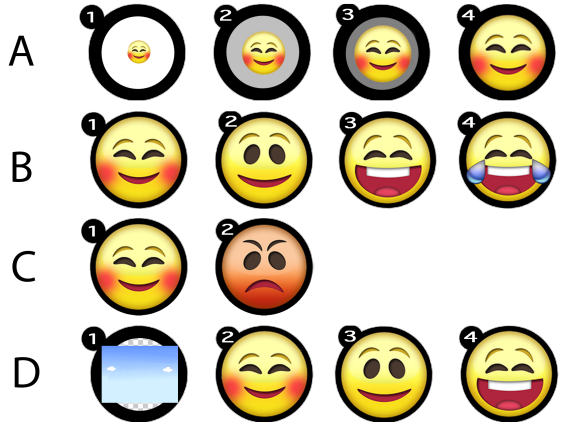


Figure 4: Effects. A:Morph B:Animation C:Flip D:3D

2.2 Pixel overview

Pixels are the building blocks of screens; pixels are composed of subpixels which are generally colored red green and blue; When all the three subpixels are at full intensity a white pixel light is generated; when all the subpixels are completely dark; black is the resulting color; combination of intensities of the different subpixels generate colors like pink, yellow, etc.

An important factor that occurs on screens is pixel spread. Pixel spread is referred in this paper as the angle from where the light can be seen from each pixel. Naturally, for TVs, monitors and screens in general, an increased pixel spread is the desired effect so people can see the same image from different angles without any variation in contrast / brightness.

Pixel spread for the purpose of this paper varies from 0 (collimated light) to 89 degrees (fan like setting). The more pixel spread the wider the angle an image on a screen can be seen from (fig. 5).

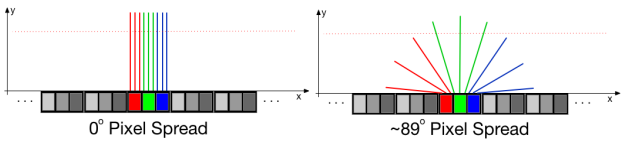


Figure 5: Pixel spread range

3 PREVIOUS WORK

Optical routing uses the angle-sensitive optical characteristics of certain materials to direct or occlude images based on the user's positionm [4].

In 1994, Little et al talk about a design for an autostereoscopic, multiperspective raster-filled display [11]. Here, they propose a time multiplexed approach and an optical routing approach. The optical routing approach features video cameras and LCTV projectors. Here, they use an array of video cameras to capture multiple perspective views of the scene and then they fed these to an array of LCTVs and simultaneously

ISBN 978-80-86943-37-4

project the images to a special pupil-forming viewing screen. The viewing screen is fabricated by either a holographic optical element or a Fresnel lens and a pair of crossed lenticular arrays.

Van Berkel et al in [12] [13] built a prototype display using a LCD and a lenticular lens from Philips Optics to display 3D images; they slanted the lenticular lens with respect to the LCD panel in order to reduce the "picket fence" effect.

Later in the same year, Matsumoto et al in [14] proposes a system that consists of combination of cylindrical lenses with different focal lengths, a diffuser screen and several projectors to create a 3D image. They had issues with one of the lenses causing a dark stripe in the 3D image affecting the stereoscopic vision by reducing the sense of depth.

Omura, presents a system that uses double lenticular lenses with moving projectors that move according to the tracked user's position to extend the viewable area [15], their system needs a pair of projectors per person and their projectors move to adjust each user's position. Their system suffers from latency due to the mechanical movement.

Lipton proposed the Synthagram [16], a system that consists of an LCD Screen with a lenticular screen that overlays the LCD display. They angled the lenticular screen in order to reduce moire patterns and their system uses nine progressive perspective views from a single image. They sample these views into a program called the Interzig where they process the images and assign each pixel to a specific position in the screen.

Matusik proposes a system that consists of an array of cameras, clusters of network connected PCs and a multi-projector 3D display with the purpose to transmit autostereoscopic realistic 3D TV [17]. They record the imagery with a small cluster of cameras that are connected to PCs. The PCs broadcast the recorded video which later on is decoded by another cluster of consumer PCs and projectors. Their 3D Display consists of 16 NEC LT-170 projectors that are used for front or rear projection. The rear projection approach consists for two lenticular sheets mounted back to back with an optical diffuser material in the center and the front projection system uses one lenticular sheet with a retro reflective front projection screen material.

Another way of optical routing approach use is the display proposed by Nguyen et al [18] [19]. Here, they propose a special display which consists of a screen with 3 layers that has directional reflections for projectors so each participant sees a customized image from their perspective; their system supports up to 5 viewing zones but doesn't support tracking and it needs a projector per participant.

Takaki et al introduces a system that can produce 72 views[20]. Their system consists of a light source array, a micro lens and a vertical diffuser (a lenticular sheet). They mention that as the horizontal positions of all light sources are different, rays from different light sources proceed to different horizontal directions after passing through the micro lenses thus generating different views. They also mention that it's difficult to fabricate a large micro lens array and also say that unused pixels remain at the corners of the LCD panel.

Later on. In [21] [22] followed by [23], Takaki discusses a multiple projection system that is modified to work as a super multiview display. Here, they attach a lens to the display screen of a HDD projector and by combining the screen lens and the common lens, they project an aperture array. This aperture array is placed on the focal plane of the common lens, and the display screen (a vertical diffuser) is placed on the other focal plane. Hence, the image of the aperture array is produced on the focal plane of the screen lens. With this, the image of an aperture array gets enlarged generating enlarged images that become viewpoints. The authors comment that there is some discontinuity between the different generated views when the observation distance is different from the distance to the viewpoints.

In 2009. Takaki and his team introduce a prototype panel that can produce 16 views [24]. They do this by building a LCD with slanted subpixels and a lenticular screen. They place a diffusion material between the lenticular sheet and the LCD screen in order to defocus the moire pattern but increase the crosstalk among viewpoints. They mention that by slanting the subpixel arrangement instead of the lenticular sheet, they can increase the number of views but the optical transmittance of the display decreases. They conclude that by slanting the subpixels in the screen instead in the lenticular sheet, they can reduce significantly the crosstalk and moire compared to the normal approaches. Their approach requires to build a LCD display which is a quite a complex task.

Finally, in 2010 Takaki and his team combine several 16-view flat-panels that have slanted subpixels[24] and creates a system with 256 views[25]. They superimpose the different projected output of the panels to a single vertical diffuser. The multiple viewing zones for each flat panel are generated on an incident pupil plane of its corresponding projection lens. Each projection lens projects to the display surface of its corresponding flat panel system on the common screen and finally a screen lens is located on the common screen so the lens generates viewing zones for observers. They mention that their prototype display has the possibility of producing 3D images on which the human eye can focus but also they report that there is considerable crosstalk between the viewing zones and the resolution of the prototype is not very high.

 ISBN 978-80-86943-37-4

Another system that takes advantage of the optical routing approach is the Free2C display, a project proposed by Surman in [26]. Here, they created a single viewer autostereoscopic display using a head tracker. The display accommodates the head movement of the viewer by continually re-adjusting the position of the lenticular lens in relation to the LCD to steer the stereoscopic views onto the eyes of the viewer. Their display resolution is 1200x1600, the viewing distance goes from 40cm to 110cm and side to side movements range of approximately +-25 degrees from the center of the screen. They also attempted a multi-user display that steers the LCD instead of the lenses to produce image regions for the users but they mention the display performance was really poor.

Similarly to Free2C, Brar et al use image recognition to track users' heads to produce multiple steerable exit pupils for left and right eyes [27] [28]. Here, they describe the design and construction of a stereoscopic display that doesnt require wearing special eye wear. A stereo par is produced on a single LCD by simultaneously displaying left and right images on alternate rows of pixels. They propose steering optics controlled by the output the aforementioned head tracker to direct regions, referred as exit pupils to the appropriate viewers' eyes. Their prototype is not optimal due to insufficient brightness and instability in the holographic projector and their current research doesn't support multiple users.

Kooima et al [29], uses 24" and 42" 3DHD Alioscopy displays[30] which come with integrated lenticular lenses. They propose a system that consists of scalable tiled displays for large field of views and use a generalization of a GPU based autostereoscopic algorithm for rendering in lenticular barriers.They tried different methods for rendering but they had issues where they perceived repeated discontinuities, exaggerated perspectives and as the displays pixels cannot be moved smoothly but in discrete steps. The tracked viewer moves into transition between channels, the user begins to see the adjacent view before the channel perspective is updated to follow the user's head.

Zang et al proposes a frontal multi-projection autostereoscopic display [31]. Their approach consists of 8 staggered projectors and a 3D image guided screen. The 3D image screen is mainly composed of a single lenticular sheet, a retro-reflective diffusion screen and a transparent layer that is filled between them to control the pitch of the rearranged pixel stripe in interlaced images. Their system is space efficient compared to previous approaches that produce light from the back of the screen, but the loss of intensity and crosstalk are seriously increased out of the system boundaries and besides being complex it doesn't provide perspective perfect views for each user.

We have mentioned here some research that has been done throughout the years that use an optical routing approach; specifically, lenticular lenses to separate users. Still, none of these projects compare the pixel spread correlation with the performance of said lenticular lenses.

4 MATHEMATICAL MODEL

Depending on the lens / screen marriage used, different lens parameters need to be set for the best performance. Some factors like the minimum number of pixels perceived, ghosting, color banding among others can be improved / minimized depending on the values the lens have. Such values need to be taken into account when pursuing an optical routing approach for multiplexing images.

To know how good or bad a specific lens with specific attributes works with any given screen, a lenticular lens simulator was developed. Such simulator helped determine the best parameters for a lens given a specific screen. This simulator also can assess how the screen's pixel spread is correlated with the lens performance.

Lenticular lenses have different parameters that can be tweaked to produce different effects; manufacturers sell lenses based on their LPI but at the end, each of these lenses comes with a set of specifications like the refraction index of the material the lens is made of, substrate thickness, viewing angle, lenticule radius, etc.(fig. 6). In order to simulate how a lenticular lens works, we

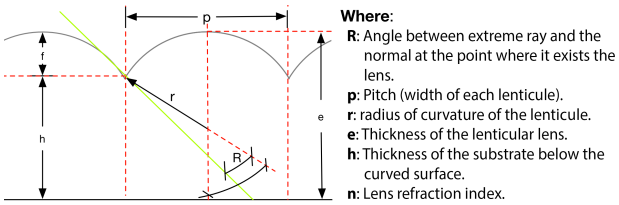


Figure 6: Detailed lenticular lens

discretize the light generated from each pixel and represent it as several rays that start from each subpixel and travel along the substrate of the lens and get refracted to the air from each lenticule. These rays are calculated in three steps: Substrate contact, lens contact and lens refraction.

4.1 Step 1: Substrate contact.

In this phase n number of rays are calculated for each pixel with a spread S (in deg) in order to get n contact points $P_0, P_1 \dots P_n$ from a horizontal line (parallel to the screen) that defines the substrate thickness of the lens (fig. 7). To find the points $P_0, P_1 \dots P_n$ where the rays intersect the end of the lens substrate we represent the rays from the pixel and the substrate with line equations and find their respective intersection points.

ISBN 978-80-86943-37-4

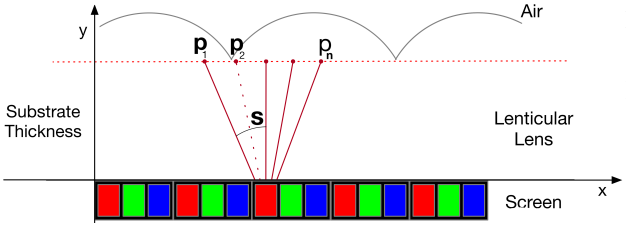


Figure 7: **Step 1:** Find where rays intersect substrate line.

Given the points R_1 and R_2 from the line L_1 that represents the ray that gets generated from the pixel and points S_1 and S_2 from the line L_2 that defines the substrate of the lens, we can generate two standard line equations with the form $y = mx + b$, make them equal on the y axis (as it is the substrate thickness that the manufacturer gives) and find the intersection point P_i on x (fig. 7).

$$L_1 \rightarrow y = m_1x + b_1 \qquad L_2 \rightarrow y = m_2x + b_2$$

$$\begin{aligned} m_1x + b_1 &= m_2x + b_2 \\ m_1x - m_2x &= b_2 - b_1 \\ x(m_1 - m_2) &= b_2 - b_1 \\ x &= \frac{b_2 - b_1}{m_1 - m_2} \end{aligned}$$ intersection X axis (P_x).

Again, finding the P_y becomes trivial as is given by the lens manufacturer and is the lens substrate thickness.

4.2 Step 2: Lens contact.

After finding where the rays of light intersect in the substrate thickness line, we proceed to find which lens the ray "belongs" to in order to apply the corresponding refraction in step three.

To do so, we find the center C_i of the lenticule l_i that is closest to the intersection P_j in order to know which lens refracts each ray from the pixel (fig. 8). To find these centers we just need to find C_x for each

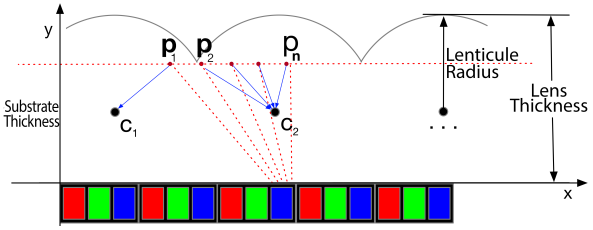


Figure 8: **Step 2:** Find points P_j closest to center C_i.

lens because C_y in all the lenticule centers remain the same and can be easily deduced from figure 8 as $C_y = lensThickness - lenticuleRadius$.

Getting C_x is also pretty straight forward as the pixel number ($PixNum$) the ray comes from is known, the physical pixel size (Pps) has already been precalculated from the screen density PPI and we can know each lenticule size (Ls) by just dividing the LPI of the lens by 1 inch ($Ls = \frac{1}{LPI}$).

First, we calculate the lenticule center that stays on top of the current pixel (Lct) with:

$$Lct = \left(\left\lfloor \frac{PixNum * Pps}{Ls} \right\rfloor * Ls\right) + \left(\frac{Ls}{2}\right) \quad (1)$$

After calculating Lct (eq. 1), we check if the distance with $P_{x\ i}$ is lesser than half of an individual lenticule size ($|Lct - P_x| < \frac{Ls}{2}$), if it is we have found the lenticule center C_x the ray belongs to, else we carry checking for neighboor lenticule centers with $C_x \pm \frac{Ls}{2}$ until the condition gets satisfied.

4.3 Step 3: Lens Refraction.

After figuring the closest lenticule center C_j from a given ray intersection P_i we can continue with the ray direction $\vec{r}$ and finally find the intersection point Q_i with the lenticule L_j where the ray refraction occurs (fig. 9). $\vec{rf}$.

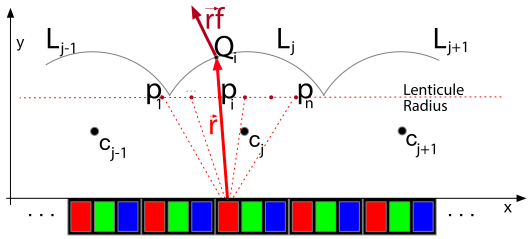


Figure 9: **Step 3:** Ray intersection with lens and refraction.

4.3.1 *Finding lens intersection point:*

To find Q_i (fig 9) we can treat each lenticule as a circle and the rays that come from each pixel as lines and then the lens-ray intersection point can be treated as a line-circle intersection as follows [32] [33]: Given a circle with center (c_x, c_y) with radius r representing the lenticule with center C_j and a line representing the ray of light that comes from a given pixel:

$$\textbf{\textit{Ray}} \rightarrow y = mx + b \qquad \textbf{\textit{Lens}} \rightarrow (x - c_x)^2 + (y - c_y)^2 = r^2$$

$$0 = (x - c_x)^2 + (mx + (b - c_y))^2 - r^2 \ \textit{Replace ray in lens equation}$$

$$0 = x^2(1 + m^2) + x(2mb - 2c_x - 2mc_y) + (c_x^2 + b^2 - 2bc_y + c_y^2 - r^2)$$

Solving the quadratic form of the resulting equation we end up with:

$$\begin{aligned} x_{1,2} &= \frac{-mb + c_x + mc_y \pm}{1 + m^2} \\ &= \frac{\sqrt{-2mbc_x + 2mc_xc_y - b^2 + 2bc_y - c_y^2 + r^2 + r^2m^2 - m^2c_x^2}}{1 + m^2} \end{aligned} \quad (2)$$

After finding the x component in eq. 2, there are three possible values that the quadratic equation gives us under the square root. Lets call this D.

$$D = -2mbc_x + 2mc_xc_y - b^2 + 2bc_y - c_y^2 + r^2 + r^2m^2 - m^2c_x^2$$

ISBN 978-80-86943-37-4

If $D<0$ there is no intersection point, when $D=0$ the line touches the lens tangentially and finally if $D>0$ there are two intersection points (as each lenticule is at the end a circle.) We are only interested in the positive value as the lenses point torward the positive Y axis so on this case, the x component of the intersection point Q_i where the ray touches the lens ends up being:

$$x = \frac{-mb + c_x + mc_y +}{1+m^2} = \frac{\sqrt{-2mbc_x + 2mc_xc_y - b^2 + 2bc_y - c_y^2 + r^2 + r^2m^2 - m^2c_x^2}}{1+m^2} \quad (3)$$

Finally, by replacing eq. 3 on the line equation from the ray we can get the y component of Q_i.

4.3.2 Generating refracted ray from the lens:

After finding the point of intersection where the ray (coming from the pixel) touches the lens (Q_i) we finally calculate the refracted pixel ray ($\vec{rf}$)(fig. 9) using Snell's law [34].

Snell's Law states that the products of the index of refraction and sines of the angles must be equal (eq. 4).

$$n_1 \sin(\theta_1) = n_2 \sin(\theta_2) \quad (4)$$

Snell's equation (eq. 4) can be re-written as:

$$\sin(\theta_2) = \frac{n_1}{n_2} \sin(\theta_1) \quad (5)$$

One can immediately see a problem here, and is that if $\sin(\theta_1) > \frac{n_2}{n_1}$ then $\sin(\theta_2)$ has to be bigger than 1 which is impossible. So when this happens, we have a TIR (*Total Internal Reflection*), TIR only happens if you go from a denser material (lens) to a less dense material (air). When TIR happens, we just ignore that ray and do nothing about it. So eq. 5 can be written like this:

$$\sin(\theta_2) = \frac{n_1}{n_2} \sin(\theta_1) \longleftrightarrow \sin(\theta_1) \leq \frac{n_2}{n_1} \quad (6)$$

To find $\vec{rf}$, lets begin by splitting it up in a tangent and a normal part:

$$\vec{rf} = \vec{rf_\parallel} + \vec{rf_\perp} \quad (7)$$

As all the vectors are normalized and any vector $\vec{v}$ can be decomposed in its tangent and parallel parts, and its parts are perpendicular to each other ($\vec{v_\parallel} \perp \vec{v_\perp}$), with basic trigonometry, the following rules apply:

$$\sin(\theta) = \frac{|\vec{v_\parallel}|}{|\vec{v}|} = |\vec{v_\parallel}| \qquad \cos(\theta) = \frac{|\vec{v_\perp}|}{|\vec{v}|} = |\vec{v_\perp}| \quad (8)$$

Since Snell's law talks about sines (eq. 6), we can use eq. 8 and rewrite eq. 6 as:

$$|\vec{rf_\parallel}| = \frac{n_1}{n_2} |\vec{r_\parallel}| \quad (9)$$

Since $\vec{rf_\parallel}$ and $\vec{r_\parallel}$ are parallel and point in the same direction, eq. 9 becomes:

$$\vec{rf_\parallel} = \frac{n_1}{n_2}\vec{r_\parallel} = \frac{n_1}{n_2}(1 - \cos(\theta_r)\vec{n}) \quad (10)$$

To find $\vec{rf_\perp}$ one can simply use pythagoras ($|\vec{v}|^2 = |\vec{v_\parallel}|^2 + |\vec{v_\perp}|^2$) and end up with:

$$\vec{rf_\perp} = -\sqrt{1 - |\vec{rf_\parallel}|^2}\vec{n} \quad (11)$$

Replacing eq. 8, eq. 9 and eq. 11 in eq. 7 we get:

$$\vec{rf} = \frac{n_1}{n_2}\vec{r} - \left(\frac{n_1}{n_2}\cos(\theta_r) + \sqrt{1 - \sin^2(\theta_{rf})}\right)\vec{n}$$

Finally, we need to find $\sin^2(\theta_{rf})$ in this last equation, but this can be easily deduced it using Snell's law in equation 9.

$$\sin^2(\theta_{rf}) = \left(\frac{n_1}{n_2}\right)^2 \sin^2(\theta_r) = \left(\frac{n1}{n2}\right)^2 (1 - \cos^2(\theta_r))$$

With these two equations we can finally calculate the refracted vector $\vec{rf}$.

Each step is built with information from the previous step starting by disctretizing the light that comes from each pixel in rays, each ray is generated from the pixel until it intersects the lens substrate (step 1). Afterwards, we figure out which lenticule the ray "belongs to" and calculate the center of the aforementioned lenticule for the ray and get the position where the ray intersects with the lens (step 2). Finally, we calculate the new refracted ray for the ray that comes from the pixel (step 3). These three steps are performed for each pixel n ray times for all the pixels in the screen.

5 EXPERIMENT

To assess how pixel spread correlates to the performance of lenticular lenses when multiplexing user views, two main factors need to be taken into account. The maximum ghosting perceived for each user and the minimum amount of unique pixels users perceive.

A simulation was performed to assess these factors. This experiment simulate separate lenses that can fit in 18 to 21 pixels per lenticule, lenses have a 50% radius increase from the minimum possible lenticule radius, lens substrate thickness of 0.1mm with a refraction index of 1.56 (PETG)[8].

The aforementioned lens is designed to fit a Samsung QN65Q900R TV. This 8k 65" TV has a resolution of 7680x4320 pixels, a refresh rate of 120Hz with physical display dimensions of 1428.51x803.538mm, pixel density of 137ppi and a pixel pitch of 0.186mm.

Tests where performed with three users evenly separated by 50 centimeters between them and 1 meter away from the screen as seen in figure 10.

ISBN 978-80-86943-37-4

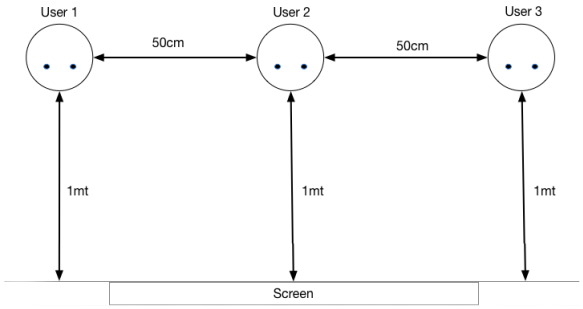


Figure 10: User separation diagram.

In this experiment we measure the minimum unique pixels perceived for each user and the maximum amount of image ghosting users get in said layout by varying the pixel spread, starting with a collimated light (0 deg spread) up to 89 degrees of pixel spread (fan like setting).

One thing to note is that in this paper we treat the spread angle as the angle from the center of the pixel to where the last ray ends (see figure 5). The total amount of "vision" from any given pixel would be twice the values of the reported spread here.

6 DATA ANALYSIS

Different spread angles will result in different metrics for ghosting and the amount of unique pixels perceived per user. As the reader can notice in table 1. As expected, the more angle spread pixel rays have the more ghosting appears between users. Also, in said table, the reader can see that regardless of the number of pixels per lenticule, the change between ghosting values is not representative.

The first thought that one could have after checking table 1 is that collimated light would perform best (as in there is no ghosting), but if the reader looks closely; with 10 degrees of pixel spread, ghosting just starts to appear. One can also see that when the pixel spread is above 20 degrees (measured from the center of the pixels), pretty much the ghosting occludes half of the imagery each user perceives.

Another thing to notice in table 1 is that ghosting increases drastically from 10 to 20 degrees and even more from 20 to 40. For sure pixel spread needs to be below 20 degrees to have an usable optical routing approach when multiplexing images between users.

Spread (deg)	Max Ghosting 18px/lent(%)	Max Ghosting 19px/lent(%)	Max Ghosting 20px/lent(%)	Max Ghosting 21px/lent(%)
0	0.000%	0.000%	0.000%	0.000%
10	2.467%	1.345%	1.139%	1.0566%
20	56.841%	56.255%	55.359%	54.684%
30	85.607%	85.580%	85.070%	84.974%
40	94.534%	94.528%	94.721%	94.587%
50	94.630%	94.603%	94.567%	94.466%
60	94.374%	93.825%	93.978%	94.068%

Table 1: Maximum ghosting perceived between 18-21 pixels per lenticule.

In table 2 the reader can also appreaciate the minimum unique number of pixels perceived between either of the three users. As one can notice from 0 to 10 degrees of pixel spread, the minimum unique perceived pixels double up to $\sim$20% without generating pretty much any ghosting (as seen in table 1).

From 10 to 20 degrees (table 2) the minimum unique pixels also increase, but not as much as from 0 to 10 and unfortunately the ghosting (table 1) increases radically when reaching 20 degrees.

It is worth mentioning that with as little as $\sim$20% of unique pixels perceived from an 8K TV, the resulting image will contain at least around 1500 pixels. Not bad for assembling an image.

Spread (deg)	Min Unique seen pixels 18px/lent(%)	Min Unique seen pixels 19px/lent(%)	Min Unique seen Pixels 20px/lent(%)	Min Unique seen Pixels 21px/lent(%)
0	12.291%	12.072%	11.875%	11.718%
10	24.140%	23.945%	23.763%	23.671%
20	32.526%	32.356%	32.291%	32.760%
30	40.065%	39.856%	39.218%	39.570%
40	46.093%	45.820%	45.625%	45.338%
50	46.992%	46.601%	46.432%	46.132%
60	46.458%	46.054%	45.898%	45.820%

Table 2: Minimum unique perceived between 18-21 pixels per lenticule.

Analyzing the initial values from tables 1 and 2 begs to ask questions like: "How much can we push the pixel spread before introducing ghosting between users?" and also "How much more unique pixels are we gaining by increasing the spread angle before generating ghosting?".

In figure 11, the reader can see three aligned graphs, the first one shows the minimum percentage of the whole screen either of the users see, the second one presents the minimum amount of unique pixels users perceive and finally the third graph displays the maximum ghosting perceived by either of the aforementioned users.

As one can see, with a collimated light, the experiment generates zero ghosting but at the same time collimated

 ISBN 978-80-86943-37-4

light doesnt maximize the amount of perceived pixels per user. If one looks carefully on the second graph, the reader can see that as the pixel spread increases, still there is zero ghosting present in the third graph until reaching 9 degrees of pixel spread with around 1700 minimum unique perceived pixels.

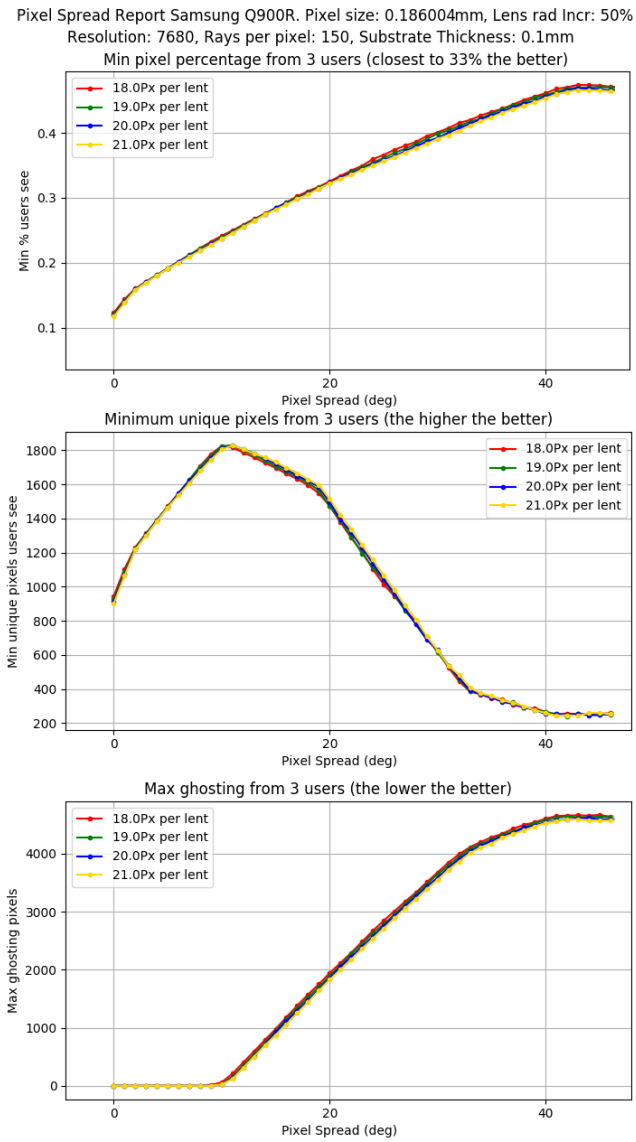


Figure 11: Pixel spread graph for Samsung Q900R 8k 65" TV.

Looking more closely, unique perceived pixels get maximized when the pixel spread is 11 degrees but, at the same time ghosting starts to appear when the spread reaches 10 degrees. When the pixel spread is 11 degrees, the maximum amount of ghosting perceived by either of the users is ∼9%.

Finally, in Figure 12 one can appreciate how ghosting also increases the thicker the substrate gets regardless the number of pixels per lenticule. With a substrate thickness of 0mm (pretty much the lencitules directly attached to the TV), the ghosting is reduced the most. Unfortunately no lenticular lens comes with a substrate thickness of 0mm. Hence the calculations done with a substrate thickness of 0.1mm.

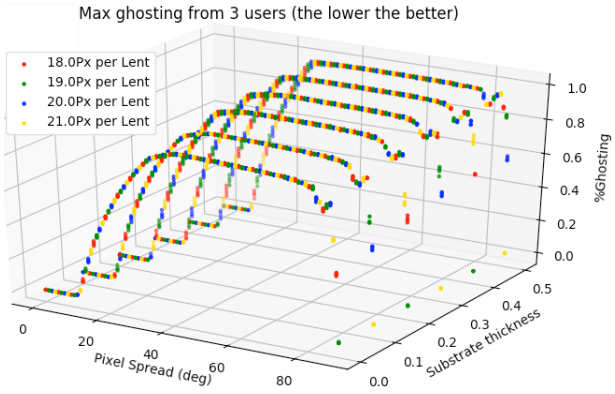


Figure 12: Ghosting vs Substrate thickness Samsung Q900R.

7 CONCLUSIONS

An overview of how lenticular lenses work and how are they used was presented in this paper; a lenticular lens simulator was also introduced and its internal mathematics on how it works where also exposed. An experiment with three users evenly separated to assess pixel spread on this paper was also mentioned and an analysis of the experiment was covered.

Ghosting is one of the biggest issues that deter the most user experience when multiplexing images through one screen. Pixel spread as discussed here is a contributing factor on ghosting when employing lenticular lenses. As presented in this paper the more straight the pixel rays the better in regards to reducing ghosting.

Lenticular lenses that have 18 to 21 pixels per lenticule, 50% of minimal lenticule radius increase and a substrate thickness of 0.1mm behave similarly in regards of maximum amount of ghosting perceived and minimum amount of unique pixels displayed for either of the users.

The percentage of perceived ghosting radically increases from 10 to 20 degrees of pixel spread as seen in table 1. It is also clear that collimated pixel rays will generate zero ghosting, but will not maximize the minimum number of unique pixels perceived by either of the users as seen in figure 11.

11 degrees of pixel spread maximizes the minimum amount of unique pixels perceived by the users but unfortunately introduces a perceived ghosting of ∼ 9%. Nine degrees of pixel spread on the other hand almost maximizes the minimum unique perceived pixels by either of the users without introducing any ghosting to the final interlaced image in the proposed setup.

With a pixel spread of nine degrees, the minimum amount of unique pixels perceived is ∼23%; more or

ISBN 978-80-86943-37-4

less 1700 pixels. Enough to assemble an image that contains zero ghosting.

Given the case that it's not physically possible to manufacture a lens with said requirements of substrate thickness of 0.1mm; its feasible to increase the thickness to some extent without inducing ghosting with 9 degrees of pixel spread.

8 FUTURE WORK

Subpixel layout is something that changes from screen to screen; in this paper we covered a subpixel layout that follows the typical red/green/blue layout with even subpixel sizes. It would be interesting to see if the same pixel spread behavior repeats with different physical subpixel layouts.

The lenticular lens simulator presented in this paper helped assessing maximum ghosting and minimum amount of unique pixels perceived by either of the users; color banding is a phenomenom that occurs when lenticules magnify enough a pixel hence producing separate red/green/blue bands instead of an even white color. Color banding wasnt addressed in this paper and is worth doing more research on this topic to assess if the proposed lens with the current pixel sizes of the tested screen generates color banding.

The experiment proposed in this paper had static users; user movement should be considererd for the data in order to assess how much ghosting and pixel spread vary troughout the physical space. A mapping of the space would be valuable to manipulate where users should be located depending on the generated virtual content.

An interesting way to reduce pixel spread would be with privacy screens; said screens are cheap and easy to attach to a screen. It would be interesting to know if with the new material between the screen and the lens; color banding is produced or if ghosting and unique pixels change.

9 REFERENCES

[1] Zsolt Szalavári, Dieter Schmalstieg, Anton Fuhrmann, and Michael Gervautz. 'studierstube': An environment for collaboration in augmented reality. *Virtual Reality*, 3(1):37–48, 1998.

[2] Brice Pollock, Melissa Burton, Jonathan W Kelly, Stephen Gilbert, and Eliot Winer. The right view from the wrong location: Depth perception in stereoscopic multi-user virtual environments. *IEEE Transactions on Visualization & Computer Graphics*, (4):581–588, 2012.

[3] Karen B Chen, Ryan A Kimmel, Aaron Bartholomew, Kevin Ponto, Michael L Gleicher, and Robert G Radwin. Manually locating physical and virtual reality objects. *Human factors*, 56(6):1163–1176, 2014.

[4] Mark Bolas, Ian McDowall, and Dan Corr. New research and explorations into multiuser immersive display systems. *IEEE computer graphics and applications*, 24(1):18–21, 2004.

[5] 3D Adworks. 3d lenticular printing. `https://3dadworks.com/en/3d-lenticular-printing/`, Mar 2018. Accessed: 2019-Apr-15.

[6] Zauberklang. Timeline of historical film colors. `https://zauberklang.ch/filmcolors/cat/lenticular-screen/#/`. Accessed: 2019-Apr-15.

[7] Rachel Rosmarin. Give me 3d tv, without the glasses. `https://www.tomsguide.com/us/3DTV-autostereoscopic-CES,review-1490.html`, Jan 2010. Accessed: 2019-Apr-15.

[8] Juan Sebastian Munoz-Arango, Dirk Reiners, and Carolina Cruz-Neira. Maximizing lenticular lens performance for multi user vr displays. In *Laval Virtual 2019*, page 12. ACM, 2019.

[9] Microlens. Choosing the correct lenticular lens sheet. `http://www.microlens.com/pages/choosing_right_lens.htm`, Nov 2018. Accessed: 2018-Nov-12.

[10] LenstarLenticular. Possible lenticular effects, Nov 2018. https://www.lenstarlenticular.com/lenticular-effects/ Accessed: 2018-Nov-12.

[11] Gordon R Little, Steven C Gustafson, and Vasiliki E Nikolaou. Multiperspective autostereoscopic display. In *Cockpit Displays*, volume 2219, pages 388–395. International Society for Optics and Photonics, 1994.

[12] Cees Van Berkel and John A Clarke. Characterization and optimization of 3d-lcd module design. In *Stereoscopic Displays and Virtual Reality Systems IV*, volume 3012, pages 179–187. International Society for Optics and Photonics, 1997.

[13] Cees Van Berkel. Image preparation for 3d lcd. In *Stereoscopic Displays and Virtual Reality Systems VI*, volume 3639, pages 84–92. International Society for Optics and Photonics, 1999.

[14] Kenji Matsumoto and Toshio Honda. Research of 3d display using anamorphic optics. In *Stereoscopic Displays and Virtual Reality Systems IV*, volume 3012, pages 199–208. International Society for Optics and Photonics, 1997.

[15] Katsuyuki Omura, Shinichi Shiwa, and Tsutomu Miyasato. Lenticular autostereoscopic display system: multiple images for multiple viewers. *Journal of the Society for Information Display*, 6(4):313–324, 1998.

 ISBN 978-80-86943-37-4

[16] Lenny Lipton and Mark H Feldman. New autostereoscopic display technology: the synthagram. In *Stereoscopic Displays and Virtual Reality Systems IX*, volume 4660, pages 229–236. International Society for Optics and Photonics, 2002.

[17] Wojciech Matusik and Hanspeter Pfister. 3d tv: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 814–824. ACM, 2004.

[18] David Nguyen and John Canny. Multiview: spatially faithful group video conferencing. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 799–808. ACM, 2005.

[19] David T Nguyen and John Canny. Multiview: improving trust in group video conferencing through spatial faithfulness. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1465–1474. ACM, 2007.

[20] Yasuhiro Takaki. Thin-type natural three-dimensional display with 72 directional images. In *Stereoscopic Displays and Virtual Reality Systems XII*, volume 5664, page 56. International Society for Optics and Photonics, 2005.

[21] Hiroshi Nakanuma, Hiroyuki Kamei, and Yasuhiro Takaki. Natural 3d display with 128 directional images used for human-engineering evaluation. In *Stereoscopic Displays and Virtual Reality Systems XII*, volume 5664, pages 28–36. International Society for Optics and Photonics, 2005.

[22] Kengo Kikuta and Yasuhiro Takaki. Development of svga resolution 128-directional display. In *Stereoscopic Displays and Virtual Reality Systems XIV*, volume 6490, page 64900U. International Society for Optics and Photonics, 2007.

[23] Yasuhiro Takaki. Super multi-view display with 128 viewpoints and viewpoint formation. In *Stereoscopic Displays and Applications XX*, volume 7237, page 72371T. International Society for Optics and Photonics, 2009.

[24] Yasuhiro Takaki, Osamu Yokoyama, and Goro Hamagishi. Flat panel display with slanted pixel arrangement for 16-view display. In *Stereoscopic Displays and Applications XX*, volume 7237, page 723708. International Society for Optics and Photonics, 2009.

[25] Yasuhiro Takaki and Nichiyo Nago. Multi-projection of lenticular displays to construct a 256-view super multi-view display. *Optics express*, 18(9):8824–8835, 2010.

[26] Phil Surman, Ian Sexton, Klaus Hopf, Wing Kai Lee, Richard Bates, Wijnand IJsselsteijn, and Edward Buckley. Head tracked single and multi-user autostereoscopic displays. 2006.

[27] Rajwinder Singh Brar, Phil Surman, Ian Sexton, Richard Bates, Wing Kai Lee, Klaus Hopf, Frank Neumann, Sally E Day, and Eero Willman. Laser-based head-tracked 3d display research. *Journal of Display Technology*, 6(10):531–543, 2010.

[28] Rajwinder Singh Brar, Phil Surman, Ian Sexton, and Klaus Hopf. Multi-user glasses free 3d display using an optical array. In *3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2010*, pages 1–4. IEEE, 2010.

[29] Robert Kooima, Andrew Prudhomme, Jurgen Schulze, Daniel Sandin, and Thomas DeFanti. A multi-viewer tiled autostereoscopic virtual reality display. In *proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*, pages 171–174. ACM, 2010.

[30] Alioscopy about us. url-http://www.alioscopy.com/en/about_us.php. Accessed: 2019-Apr-15.

[31] Shang-Fei Zang, Qiong-Hua Wang, Wu-Xiang Zhao, Jie Zhang, and Jing-Long Liang. A frontal multi-projection autostereoscopic 3d display based on a 3d-image-guided screen. *Journal of display technology*, 10(10):882–886, 2014.

[32] AmBrSoft. Intersection of a circle and a line. `http://www.ambrsoft.com/TrigoCalc/Circles2/circlrLine_.htm`, Feb 2018. Accessed: 2018-Nov-14.

[33] Attila's Projects. Circle and line intersection. `http://apetrilla.blogspot.com/2011/12/circle-and-line-intersection.html`, Dec 2011. Accessed: 2018-Nov-14.

[34] Bram De Greve. Reflections and refractions in ray tracing. *Retrived Oct*, 16:2014, 2006.

 ISBN 978-80-86943-37-4

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

Anatomy-Focused Volume Line Integral Convolution for Brain White Matter Visualization

T. Schult
Institute for Applied Computer Science
Stralsund University of Applied Sciences
Zur Schwedenschanze 15
Germany, 18435 Stralsund, MV
thomas.c.schult@hochschule-stralsund.de

U. Klose
Department of Diagnostic and Interventional Neuroradiology
University Hospital Tübingen
Hoppe-Seyler-Straße 3
Germany, 72076 Tübingen, BW
uwe.klose@med.uni-tuebingen.de

T.-K. Hauser
Department of Diagnostic and Interventional Neuroradiology
University Hospital Tübingen
Hoppe-Seyler-Straße 3
Germany, 72076 Tübingen, BW
till-karsten.hauser@med.uni-tuebingen.de

H.-H. Ehricke
Institute for Applied Computer Science
Stralsund University of Applied Sciences
Zur Schwedenschanze 15
Germany, 18435 Stralsund, MV
hans.ehricke@hochschule-stralsund.de

ABSTRACT

3D visualization of volumetric line integral convolution (LIC) datasets has been a field of constant research. So far, most approaches have focused on finding suitable transfer functions and defining appropriate clipping strategies in order to solve the problem of occlusion. In medicine, extensions of the LIC algorithm to diffusion weighted magnetic resonance imaging (dwMRI) have been proposed, allowing highly resolved LIC volumes to be generated. These are used for brain white matter visualization by LIC slice images, depicting fiber structures with good contrast. However, 3D visualization of fiber pathways by volume rendering faces the problem of occlusion of anatomic regions of interest by the dense brain white matter pattern. In this paper, we introduce an anatomy-focused LIC algorithm, which allows specific fiber architectures to be visualized by volume rendering. It uses an anatomical atlas, matched to the dwMRI dataset, during the generation of the LIC noise input pattern. Thus, anatomic fiber structures of interest are emphasized, while surrounding fiber tissue is thinned out and its opacity is modulated. Additionally, we present an adaptation of the orientation-dependent transparency rendering algorithm, which recently has been proposed for fiber streamline visualization, to LIC data. The novel methods are evaluated by application to dwMRI datasets from glioma patients, visualizing fiber structures of interest in the vicinity of the lesion.

Keywords

Volume LIC, Line integral convolution, Anatomical atlas, Visualization, Volume rendering, Diffusion MRI

1. INTRODUCTION

Three-dimensional visualization of data from line integral convolution (LIC) is a challenging field in scientific visualization. The major challenges in Volume LIC are occlusion and finding suitable transfer functions. Although streamline tractography has been established as a standard technology, LIC is a promising approach in the visualization of fiber pathways from diffusion weighted magnetic resonance imaging (dwMRI) datasets. However, three-dimensional depiction of LIC data suffers from occlusion and cluttered visualization. Figure 1 presents an example of rendering a dwMRI LIC volume with dense white matter structures. In previous solutions, clipping approaches and sophisticated transfer functions have been proposed to visualize vector fields. In particular, when visualizing dense LIC volumes from dwMRI data, such strategies do not lead to visualizations depicting the three-dimensional course of single fiber bundles and lack anatomical context. Furthermore, most of these recent approaches focus on depicting a whole LIC volume after it has been generated.

In this paper, we propose a new approach by emphasizing volumes of interest (VOI) while generating the noise input pattern to the LIC algorithm. This is achieved by the incorporation of an anatomical atlas which is matched to the individual dwMRI dataset, allowing anatomical VOIs to be specified by the user without tedious interactions. Our

approach is based on the A-Glyph LIC algorithm proposed by Höller et al. [HESK17, HOKG14].
The paper is organized as follows. First, we give an overview of related methods and solutions to the problem of visualizing Volume LIC datasets (section 2). In section 3, we give a detailed description of our approach of anatomy-focused Volume LIC by atlas-based sample placement. In section 4, experimental results from the application of the methods to clinical datasets from glioma patients are presented. We conclude with section 5 by discussing our results and giving an outlook to further algorithmic amendments.

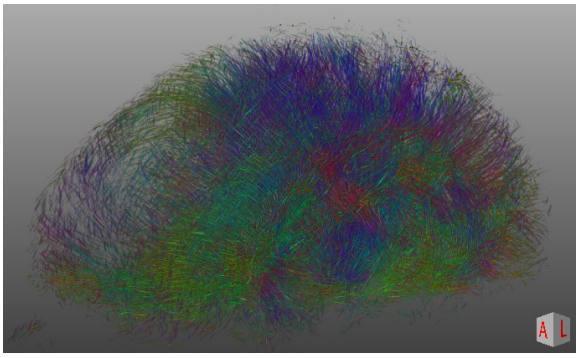


Figure 1. Volume rendering of a Volume LIC dataset. An insight into the inner structures of the brain is occluded. In the front part of the brain, lowered fiber density can be seen.

2. RECENT SOLUTIONS

State of the art – Volume LIC

LIC, originally introduced by Cabral and Leedom [CaLe93], is a texture-based algorithm to visualize vector fields. In its original two-dimensional version, it takes a random noise texture as input and blurs it locally along the given vector field. There are several extensions to the LIC algorithm [LHDV04] that include Volume LIC as well. In scientific literature there are different approaches to visualize Volume LIC data with the aim to give insight into the volumes and to highlight certain characteristics of the underlying data. One area of research focuses on various forms of clipping planes. Schurade et al. [SHSK10] use a Virtual Klinger dissection method with which curved dissection surfaces are generated. These extend locally parallel to user specified fiber bundles and thereby shape a free-form clipping surface.
Peeters et al. [PeVR09] pursue a different approach. They do not calculate a whole LIC volume and depict it, but define a 2D cross section of a heart and carry out a 3D vector field visualization for this “Plane of Interest”. In the specified intersection of the plane with the vector field, the 3D vectors are rendered as 3D line segments by using a local raycasting approach. In addition, line lightning and shadow computation is performed.
The question of how four-dimensional blood flow patterns can be visualized interactively in vivo is answered by Kainz et al. [KRRS09]. Here, an overview of experiments of how to visualize and measure flow fields in the human vascular system is given and sparse and dense flow visualization algorithms are presented. To visualize the results, a volume raycasting approach is introduced that reintegrates 3D information into 2D cutting planes by using velocity directions as plane normal vectors.
Another field of research focuses on the use of transfer functions in generating volume renderings. Tax et al. [TCSV15] do not describe a solution for visualizing Volume LIC, but propose an adaptation of the transparency rendering of fiber trajectories as a function of fiber orientation. This can lead to an enhanced visibility, because pathways that are orientated along a user-specified axis, can be made more transparent. Other approaches render LIC volumes based on 3D Significance Maps [ChFS03, SFCN03]. Significance values are derived from intrinsic properties of the vector field and control the transfer functions for LIC volume rendering by manipulating the opacity of each texel, whereby different regions are highlighted or neglected. Besides the single use of clipping planes and transfer functions, there are several approaches that combine both techniques. Burns et al. [BHWV07] propose feature emphasis and contextual cutaways for multimodal medical visualization. This importance driven approach aims to present embedded data in a way that it is clearly visible along with its spatial relation to the surrounding volumetric material. The specification of importance and ranking of materials within the volume is achieved by augmentation of conventional transfer functions. Additionally, view-dependent cutaway structures are used to cut away occluding materials based on its importance. Similar to the importance driven approach is the employment of cost functions introduced by Auzinger et al. [AMBS13]. A Curved Surface Reformation is proposed to visualize vessel lumen. This method generates a cut surface along a vessel centerline fully in 3D and handles occlusion by employing a cost function on the cut surface that incorporates the location of individual points to the viewer and to the vessel centerline. Correspondingly necessary cutaways are then generated automatically.
Rezk-Salama et al. extend the use of clipping planes and transfer functions to animated clipping objects that are used for an interactive exploration of Volume LIC based on 3D-Texture Mapping [RHTE99]. Here, LIC calculation is based on sparse noise textures. Through animated clipping objects, an insight into the volume can be given. Additionally, interactive assignment of color and opacity values is used to

 ISBN 978-80-86943-37-4

enhance or suppress portions of the data. Liu et al. propose with animated and arbitrarily-oriented cutting planes a similar approach [LiMo05]. Here, time-varying flow volumes are visualized, using volume rendering with magnitude-based transfer functions that show flow structures in individual frames. The rendering method takes two volume datasets as an input. One is the flow texture and the other includes flow magnitude. During visualization only the flow texture is rendered while the flow magnitude is used to define color and opacity mapping in the transfer function.

Highlighting salient features inside volumetrically data is also possible by using short animations during user interactions [SRBG10]. The purpose is to explore the data more efficiently and to aid the user in the detection of unknown features. The method tries to anticipate the region of interest and changes the rendering style to improve the visualization.

The methods listed so far mainly focused on improving the three-dimensional visualization after the volume datasets were generated. However, Interrante et al. present strategies for effectively visualizing 3D flow with Volume LIC [InGr97]. A definition of an appropriate input texture is described, that clarifies the distinct identities and relative depths of the advected texture elements. Regions of interest (ROI) are highlighted in the input and output volumes using "halos". The input texture of the LIC algorithm is sparsely opaque with evenly distributed points that are premultiplied by a function of the velocity of the flow. The halo effect is then achieved by using a second, slightly different input texture which is incorporated in the rendering process. A method that also uses a sparse input texture and produces a halo effect is introduced by Helgeland et al. [HeAn04]. Here, shading is done by limb darkening using a transfer function and performing a two-field visualization. Unlike before, seeding is done by the usage of ROIs, so that seeds are only placed inside these ROIs and the focus of the visualization lies on these regions.

Although the aforementioned methods work well in their respective application areas, they are not directly applicable to Volume LIC data of white matter. Particularly, usage of sophisticated transfer functions to highlight individual fiber bundles is of limited effect, because they do not differ enough in their intensity and change their color in their course. Furthermore, with clipping techniques it is hardly possible to emphasize single fiber bundles and to depict their anatomical context.

3. METHODS

A-Glyph LIC

We base our methods on the A-Glyph LIC approach by Höller et al. [HESK17, HOKG14], which produces high contrast color-coded LIC maps that visualize local anisotropy information and regional fiber architecture of the human brain from dwMRI datasets. Typically, these are acquired by the high-angular resolution diffusion imaging (HARDI) technique. Since the approach has been published before, we give only a short summary here. A-Glyph LIC is an extension of the LIC approach proposed by Cabral and Leedom [CaLe93]. Different to the procedure suggested there, no white noise is used as input texture, because this leads to low-contrast fiber visualizations with poor delineation of fiber pathways. Instead, an anisotropic glyph sample pattern is generated and processed with a multiple kernel LIC algorithm. Since it is the groundwork for the methods described in this paper, we will give a brief explanation in the following. The different steps of the A-Glyph LIC method are illustrated by Figure 2. The necessary directional vectors to carry out LIC, originate from fiber orientation distributions (FOD) computed by spherical deconvolution from the dwMRI data [TCGC04, ToCC07]. From the FODs the direction vectors of the fibers can be derived by determination of global and local maxima of anisotropic diffusion. The first step of A-Glyph LIC is the generation of a high-resolution input pattern with an isotropic voxel size of 0.1 mm. We use multi-cylindrical glyph samples derived from the FODs and place them along very short streamlines. The streamlines originate from seeds, which are randomly distributed over the whole volume. Then, a multiple kernel LIC algorithm smoothes the generated high-resolution anisotropic glyph input pattern. This is done

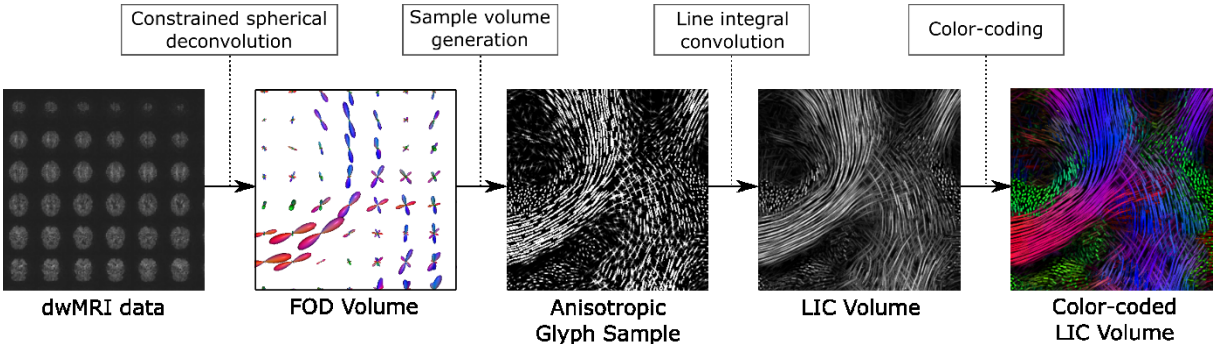


Figure 2. Processing steps of the A-Glyph LIC algorithm.

ISBN 978-80-86943-37-4

not only along a single anisotropy direction, but also along a second local FOD maximum. Thus, crossing and branching fiber pathways can be depicted. In the last step, planar 2D slices are generated from the resulting 3D LIC volume. By directional color-coding fiber orientation in 3D space is depicted [PaPi99].

Proposed Solution

This paper focusses on Volume LIC data calculated with the A-Glyph LIC algorithm. Originally, the input texture of the convolution process consists of anisotropic glyph samples, placed randomly with the same probability throughout the whole volume. This produces good results for 2D orthogonal slice images, but leads to occlusion in volume renderings (see Figure 1). To overcome this, we introduce an anatomy-driven approach which varies the sample placement probability for different anatomic regions. These regions are defined by usage of an anatomical atlas. The ICBM-DTI-81 atlas is utilized, created from diffusion tensor imaging of 81 healthy volunteers, that segments the white matter of the human brain into fifty areas [MOJJ08]. This atlas is individually registered to the respective dwMRI datasets, using the atlas based segmentation [ChRM94, MCAG06, RoMa05] in the Computational Morphometry Toolkit (CMTK) [Comp18].

During computation of the noise input texture for the LIC algorithm, a VOI is defined by selecting segmented white matter structures of interest. Inside the VOI the probability of placing anisotropic glyph samples is increased from 0.5 percent, what would be the standard probability of the A-Glyph LIC algorithm, to 1.0 percent, whereas in an adjustable environment around the VOI the glyph placement probability is decreased linearly from 1.0 percent to 0.025 percent. This procedure allows the anatomical VOI to be emphasized within the input texture. After application of the LIC algorithm, occlusion by fiber structures outside the VOI is diminished because of a sparser placement of glyph samples outside the specified VOI.

For volume rendering of the anatomy-focused LIC volume we used MeVisLab (MeVis Medical Solutions AG, Bremen, Germany) [Mevi18]. To allow analysis of fiber affection near lesions, we used the integrated surface renderer for 3D display of the tumor surface. In a pre-processing step the tumor is delineated through segmentation with a region growing algorithm. All of the algorithms for tumor delineation and reconstruction are provided by MeVisLab modules.

To enhance 3D visualization, the orientation-dependent transparency rendering approach proposed by Tax et al. [TCSV15] was adopted to Volume LIC datasets. The original algorithm of making fibers more transparent along a user-defined axis is based on streamlines, produced by whole-brain tractography. For a single streamline its path through 3D space is known and its overall orientation can easily be computed. Since LIC generated a fiber texture only, the course and overall orientation of a single fiber cannot be determined. However, based on the orientational color-coding scheme, the local orientation of the fiber can be determined for every voxel. The RGB color vector represents a fiber's spatial orientation in x-, y- and z-direction. After definition of the viewing direction, all voxels in a conic region around the viewing axis are made more transparent. Thus, fiber structures running orthogonal or nearly orthogonal to the viewing plane are eliminated and no longer occlude fibers of interest. Figure 3 summarizes the processing steps of the anatomy-focused Volume LIC algorithm.

As the computing environment we used a Lenovo ThinkPad S540 laptop computer with an Intel Core i7 processor, 16 GB RAM, and an AMD Radeon HD 8670M GPU with 2 GB VRAM. Computationally expensive routines of the A-GLYPH LIC algorithm were transferred to the graphics hardware using the OpenCL programming language. Thus, high-resolution volume LIC datasets could be computed within timeframes of typically approx. 25-30 minutes. Realtime volume rendering by MeVisLab was carried out on the GPU by exploiting the texture mapping hardware.

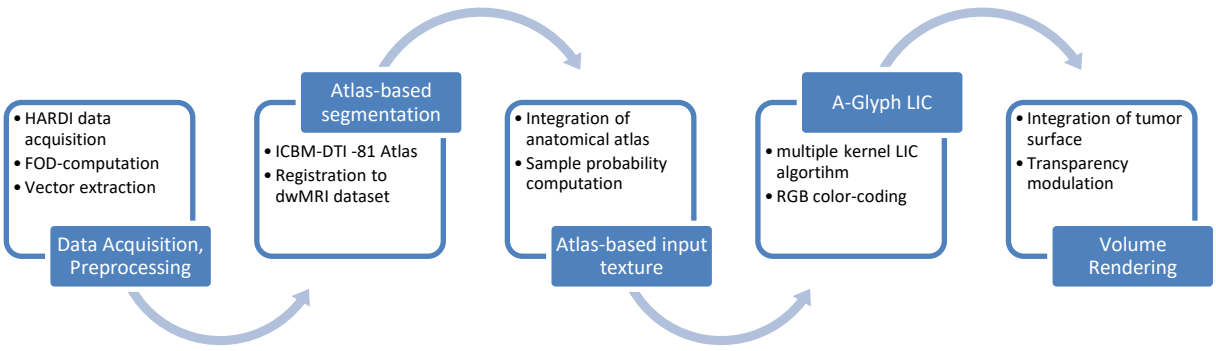


Figure 3. Processing steps of the anatomy-focused Volume LIC algorithm.

ISBN 978-80-86943-37-4

Patient Datasets

For a preliminary evaluation, two clinical patient datasets with different pathologies were used. The datasets were acquired as part of ongoing research studies approved by the ethics committee of the medical faculty of the Eberhard-Karls-University of Tübingen. Informed-written consent was obtained from the patients or their parents. Patient A is a 54-year old woman with a glioblastoma in the right frontal lobe. Patient B is a 15-year old boy with a left-sided brain tumor in the central region close to the primary motor cortex. Both datasets were acquired at the University Hospital Tübingen. Table 1 gives an overview of the acquisition protocols.

	Patient A	Patient B
Manufacturer	Siemens	Siemens
Scanner	Skyra	Sonata
Magnetic field strength [T]	3.0	1.5
Number DWI directions	64	60
B-value [s/mm²]	1000	3000
TR/TE [ms]	6100/85	11500/122
Voxel length [mm]	2.0	2.5

Table 1. Diffusion weighted magnetic resonance imaging (dwMRI) acquisition protocols

4. EXPERIMENTAL RESULTS

The methods described above were applied to the two clinical datasets. By atlas-based generation of the anisotropic glyph samples the focus was set to clinically relevant regions in the vicinity of the tumor. Figure 4 shows data from Patient A, opposing the original input texture of the A-Glyph LIC algorithm (left) to the atlas-based input texture (right). In this axial view it is clearly noticeable that there are less glyph samples outside the region around the VOI, defined by the corpus callosum (CC). This is because the probability of placing glyph samples outside the VOI was only 2.5 percent of that for placing them inside the VOI. However, there is no smooth transition from inside to outside of the atlas-based VOI. This problem can be solved by smooth adaptation of the placement probability from inside to outside in a continuous manner. The result is demonstrated in Figure 5, showing data from Patient A. Here, the effect of increasing the environment around the VOI is depicted. Figure 5a shows the result of the A-Glyph LIC method with the atlas-based glyph sample shown in Figure 4b. To incorporate the closer neighborhood around the CC, Figure 5b expands the VOI by an environment of 5 voxels. Thus, additional anatomical context is displayed. In Figure 5c the environment is expanded even more to 10 voxels. However, this leads to an undesirable occlusion. For this reason, we decided to use an environment of 5 voxels as a default value.

To avoid occlusion by fiber bundles that run along the viewing axis, voxels that represent such fibers are rendered transparent. Figure 6 demonstrates the effect of this occlusion culling technique on an axial view of Patient A. In Figure 6a, fibers that run along the viewing axis are depicted in blue color. They are not rendered transparent. Occlusion is diminished by making fibers transparent that run along the viewing axis within a cone of 45 degrees. As a result, a damaged area of the CC with lower fiber density becomes visible (marked by yellow arrows).

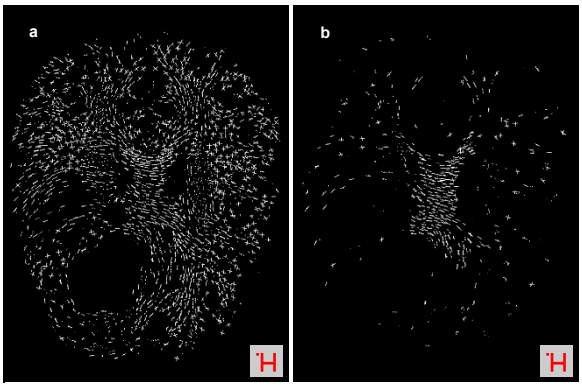


Figure 4. Anisotropic glyph samples for the LIC algorithm in an axial view, generated from the dataset of Patient A. (a) Original input texture, as proposed by Höller et al. (b) Atlas-based input texture with focus on the corpus callosum.

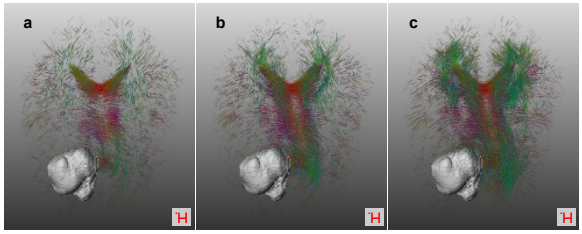


Figure 5. Effect of adjusting the environment around the atlas-based VOI, in which the probability of placing glyph samples is lowered, demonstrated by data from Patient A. (a) No environment, (b) environment of 5 voxels, and (c) environment of 10 voxels.

Through the aforementioned steps a better insight into the inner structure of volume LIC datasets can be given. Figure 7 shows a comparison of volume rendering and anatomy-focused Volume LIC with data from Patient A. Both visualization approaches are combined with surface rendering of the tumor surface. For the atlas-based generation of the anisotropic glyph sample, the corpus callosum was used as VOI, which is a fiber bundle that links both hemispheres of the

ISBN 978-80-86943-37-4

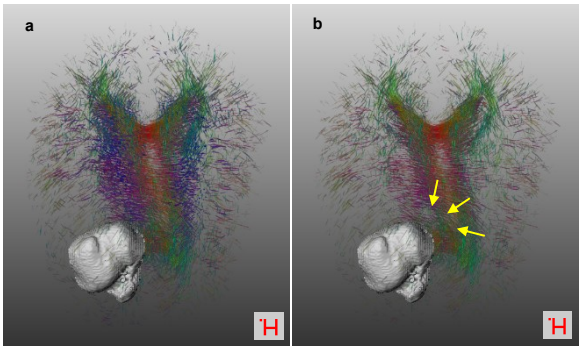


Figure 6. Adaptation of orientation-dependent transparency rendering to anatomy-focused Volume LIC data of Patient A. (a)Without orientation-dependent transparency rendering. (b) Transparent rendering of fibers that run along the viewing axis. The yellow arrows mark a region of CC with reduced fiber density.

brain. It is a central lead structure for neuroradiologists and neurosurgeons for diagnosis and therapy planning. In the example, the CC is damaged by a tumor and its surrounding edema inside the right frontal lobe. By the anatomical focus inside the input texture of the LIC algorithm and through rendering fibers running from anterior to posterior transparent, the whole CC and its damaged region become visible (marked by yellow arrowsFigure 8 shows the issue from another perspective. In the volume rendering result of Figure 8a, an insight into inner structures of the brain is prevented by occlusion. The CC is depicted only in part in the fissura longitudinalis. However, by applying the anatomy-focused Volume LIC approach the whole CC becomes visible and its disruption can be seen in Figure 8b (marked by the yellow arrows). The choice of the anatomical focus to be selected as VOI depends on the pathology of the individual patient dataset. In Patient A, there is a tumor inside the right frontal lobe that damages parts of the CC. Therefore, the atlas-based generation of the anisotropic glyph sample focuses on this region. In the dataset of Patient B, there is a tumor in the left side of the brain that might interfere with the cortico-spinal tract (CST). Neuroradiologists and neurosurgeons often use a left-to-right comparison to analyze pathologies. Therefore, in this case the anatomical focus was on the left and right CST. Figure 9 shows a comparison, similar to that presented by the two previous figures. In Figure 9a the LIC volume is visualized by volume rendering together with the reconstructed tumor surface, located in the left hemisphere. In the image an impairment of a CST is not recognizable. However, Figure 9b indicates a reduced fiber density and disruption of the left branch of the CST in the region, marked by a dotted yellow ellipse, in comparison to the right branch of the CST, marked by a continuous yellow ellipse. To get a better view on the left CST, fibers running from anterior to posterior were rendered transparent.

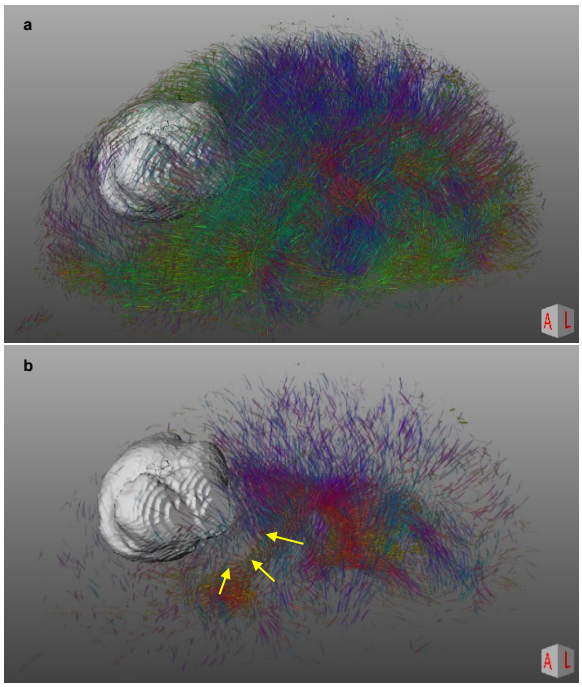


Figure 7. Comparison of (a) volume rendering and (b) anatomy-focused Volume LIC with transparent fibers running from anterior to posterior (Patient A). On both visualizations the same look-up table has been used and both are combined with the reconstructed tumor surface (grey). The arrows point to a pathological disruption of the corpus callosum.

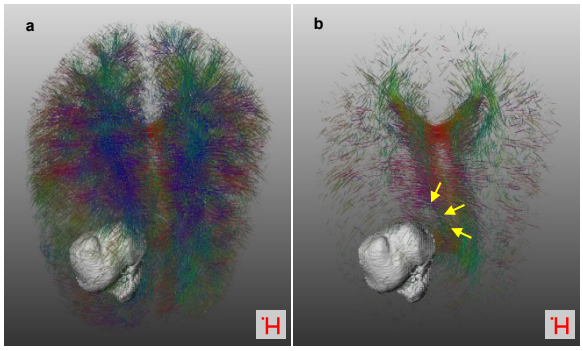


Figure 8. Same comparison as in Figure 7, but with an axial view. (a) Volume rendering with occlusion of inner structures of the brain. (b) Anatomy-focused visualization of the CC, revealing a disrupted region, marked by yellow arrows.

ISBN 978-80-86943-37-4

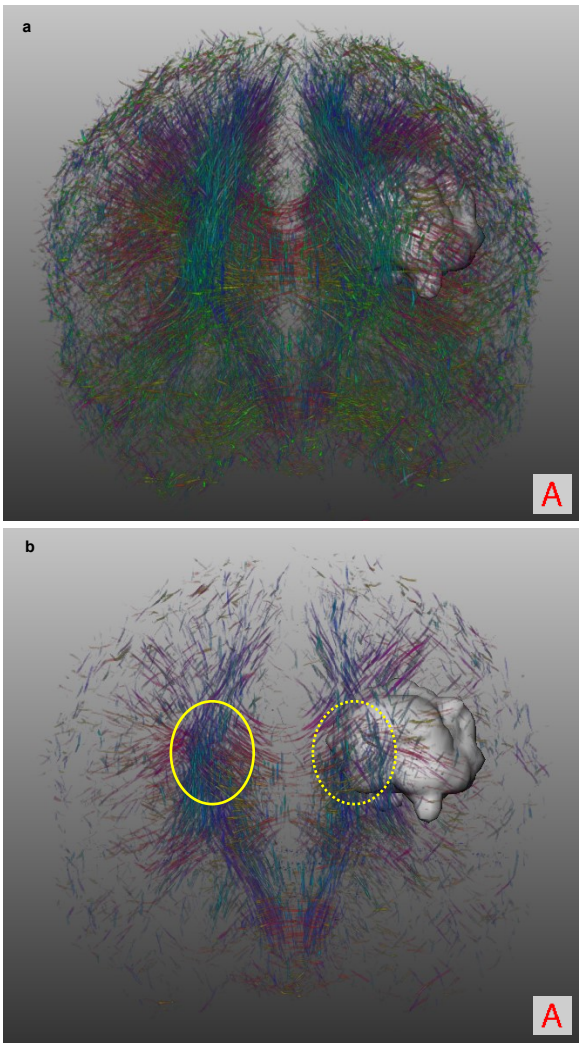


Figure 9. Coronary view of volume rendering and anatomy-driven visualization with focus on left and right CST from Patient B. (a) Volume rendering with integrated tumor surface in the left hemisphere. (b) Anatomy-focused Volume LIC that reveals a reduced fiber density of the left branch of the CST (dotted yellow ellipse) in comparison to the right branch (yellow ellipse).

Figure 10 shows a final comparison of four different methods to visualize the CC of Patient A in an axial view. In contrast to previous axial representations, a view from bottom to top is given. In Figure 10a, showing a volume rendering of the LIC volume, the CC is occluded by fibers running from anterior to posterior (depicted in green) and from superior to inferior (depicted in blue). A disruption of the CC as shown in Figure 7b cannot be seen. Figure 10b shows a volume rendering with an optimally adjusted look-up table and transparency to present the CC in more detail. Since in LIC data single fibers do not differ enough in their intensity the application of transfer functions is of limited effect. Especially fibers from the CC and the CST are of similar intensity in this example, which leads to the CC still being occluded and the disrupted area of the CC can just be seen to some extent (marked by yellow arrows). Furthermore, the adjustment in the look-up table and the transparency causes skips in the CC (marked by dotted yellow ellipse). Figure 10c shows a volume rendering with applied clipping to select the CC. Since the course of the CC is selectable by planar clipping planes the disruption becomes visible to some extent (marked yellow arrows) but is still partially occluded. With more curved fiber bundles, planar clipping planes would not be sufficient and curved planes would have to be used. Furthermore, the anatomical context of the clipped region might be missing. Figure 10d shows the anatomy-focused Volume LIC with emphasized CC and transparent rendering of fibers running along the viewing axis. Here, the disrupted area of the CC with reduced fiber density becomes visible (marked by yellow arrows). This method is applicable to more curved fiber bundles as well, but its accuracy depends on the used anatomical atlas and on the precision of the matching between atlas and individual patient dataset.

5. DISCUSSION and CONCLUSION

With anatomy-focused Volume LIC for brain white matter visualization, we have presented a novel approach to utilize Volume LIC for dwMRI data. To minimize occlusion effects, an integration of an anatomical atlas is proposed, that is registered to the individual dwMRI dataset. While generating the anisotropic glyph sample input texture for the LIC algorithm, the atlas can be used to vary the sample placement probability in different anatomic regions. Anatomy-focused Volume LIC makes it possible to get an insight into the inner structures of the brain and to highlight single fiber bundles. With an adaptation of the orientation-dependent transparency rendering approach, fibers that run along the viewing axis can be rendered transparent to clarify the display even more. The presented results focus on two patient datasets with different types and locations of brain tumors. Next steps will be a broader clinical evaluation with more patient datasets with different pathologies. Furthermore, it has to be examined how well a registered anatomical atlas fits an individual patient dataset, especially when brain structures are considerably displaced by pathologies. Here, novel techniques of elastic atlas matching have to be evaluated in order to avoid manual adjustments. Additionally, other anatomical atlases with different segmentations of anatomic structures will be analyzed as to their potential use for generating atlas-based input structures to the LIC algorithm.

ISBN 978-80-86943-37-4

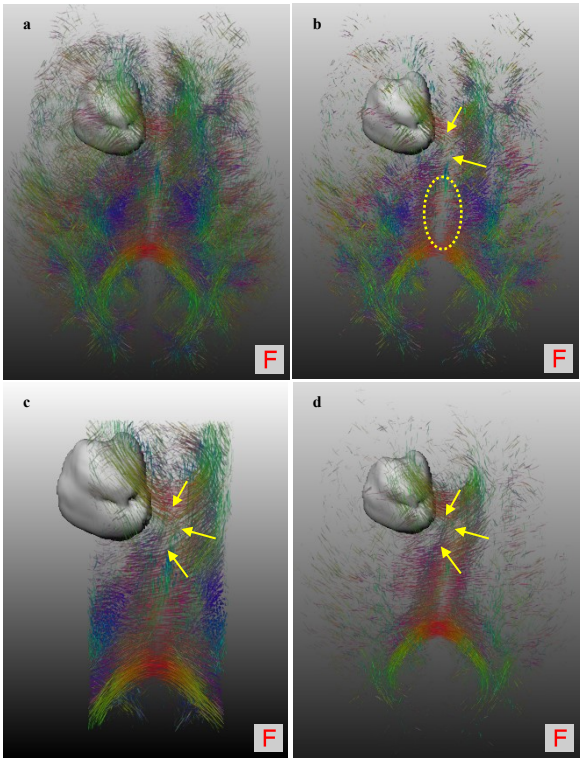


Figure 10. Comparison of the depiction of the CC from Patient A on (a) volume rendering, (b) volume rendering with adjusted look-up table and transparency, (c) volume rendering with applied clipping and (d) anatomy-focused Volume LIC with emphasized CC and transparent fibers running from superior to inferior. The tumor surface in the left hemisphere is integrated in all representations. Yellow arrows mark the region where the CC is disrupted. The dotted yellow ellipse marks a region with skips in the CC due to look-up table and transparency adjustments.

REFERENCES

[AMBS13] AUZINGER, THOMAS; MISTELBAUER, GABRIEL; BACLIJA, IVAN; SCHERNTHANER, RUDIGER; KOCHL, ARNOLD; WIMMER, MICHAEL; GROLLER, M. EDUARD; BRUCKNER, STEFAN: Vessel visualization using curved surface reformation. In: *IEEE Transactions on Visualization and Computer Graphics* Bd. 19 (2013), S. 2858–2867

[BHWV07] BURNS, MICHAEL; HAIDACHER, MARTIN; WEIN, WOLFGANG; VIOLA, IVAN; GRÖLLER, M EDUARD: Feature Emphasis and Contextual Cutaways for Multimodal Medical Visualization. In: *Eurographics / IEEE VGTC Symposium on Visualization 2007* : Eurographics Association, 2007, S. 275–282

[CaLe93] CABRAL, BRIAN; LEEDOM, LEITH CASEY: Imaging vector fields using line integral convolution. In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques - SIGGRAPH '93*. Bd. 27. New York, New York, USA : ACM Press, 1993 — ISBN 0897916018, S. 263–270

[ChFS03] CHEN, LI; FUJISHIRO, ISSEI; SUZUKI, YASUKO: Comprehensible Volume LIC Rendering based on 3D Significance Map. In: *Visualization and Data Analysis 2002*. Bd. 4665, 2003, S. 142–153

[ChRM94] CHRISTENSEN, G. E.; RABBITT, R. D.; MILLER, M. I.: 3D brain mapping using a deformable neuroanatomy. In: *Physics in Medicine and Biology* Bd. 39 (1994), Nr. 3, S. 609–618

[Comp18] *Computational Morphometry Toolkit (CMTK)*. URL https://www.nitrc.org/projects/cmtk/. - abgerufen am 2018-12-17

[HeAn04] HELGELAND, ANDERS; ANDREASSEN, OYVIND: Visualization of vector fields using seed LIC and volume rendering. In: *IEEE Transactions on Visualization and Computer Graphics* Bd. 10 (2004), S. 673–682

[HESK17] HÖLLER, M; EHRICKE, H.-H.; SYNOFZIK, M; KLOSE, U; GROESCHEL, S: Clinical Application of Fiber Visualization with LIC Maps Using Multidirectional Anisotropic Glyph Samples (A-Glyph LIC). In: *Clinical Neuroradiology* Bd. 27 (2017), Nr. 3, S. 263–273

[HOKG14] HÖLLER, MARK; OTTO, KAY-M.; KLOSE, UWE; GROESCHEL, SAMUEL; EHRICKE, HANS-H.: Fiber Visualization with LIC Maps Using Multidirectional Anisotropic Glyph Samples. In: *International Journal of Biomedical Imaging* Bd. 2014 (2014), S. 1–14 — ISBN 1687-4188 (Print) 1687-4188 (Linking)

[InGr97] INTERRANTE, V.; GROSCH, C.: Strategies for effectively visualizing 3D flow with volume LIC. In: *roceedings of the 8th Conference on Visualization '97* : IEEE Computer Society Press, 1997, S. 421ff.

[KRRS09] KAINZ, BERNHARD; REITER, URSULA; REITER, GERT; SCHMALSTIEG, DIETER: In vivo interactive visualization of four-dimensional blood flow patterns. In: *The Visual Computer* Bd. 25 (2009), Nr. 9, S. 853–862

[LHDV04] LARAMEE, ROBERT S.; HAUSER, HELWIG; DOLEISCH, HELMUT; VROLIJK, BENJAMIN; POST, FRITS H.; WEISKOPF, DANIEL: The State of the Art in Flow Visualization: Dense and Texture-Based Techniques. In: *Computer Graphics Forum* Bd. 23 (2004), Nr. 2, S. 203–221 — ISBN 1467-8659

[LiMo05] LIU, ZHANPING; MOORHEAD, ROBERT J.: A texture-based hardware-independent technique for time-varying volume flow visualization. In: *Journal of Visualization* (2005), S. 235–244

[MCAG06] MILLER, M. I.; CHRISTENSEN, G. E.; AMIT, Y.; GRENANDER, U.: Mathematical textbook of deformable neuroanatomies. In: *Proceedings of the National Academy of Sciences* Bd. 90 (2006), Nr. 24, S. 11944–11948

[Mevi18] MEVIS MEDICAL SOLUTIONS AG: *MeVisLab*. URL https://www.mevislab.de. - abgerufen am 2018-10-04

[MOJJ08] MORI, SUSUMU; OISHI, KENICHI; JIANG, HANGYI; JIANG, LI; LI, XIN; AKHTER, KAZI; HUA, KEGANG; FARIA, ANDREIA V.; U. A.: Stereotaxic white matter atlas based on diffusion tensor imaging in an ICBM template. In: *NeuroImage* Bd. 40 (2008), S. 570–582 — ISBN 1053-8119 (Print) 1053-8119 (Linking)

[PaPi99] PAJEVIC, SINISA; PIERPAOLI, CARLO: Color schemes to represent the orientation of anisotropic tissues from diffusion tensor data: Application to white matter fiber tract mapping in the human brain. In: *Magnetic Resonance in Medicine* Bd. 42 (1999), Nr. 3, S. 526–540

[PeVR09] PEETERS, T. H. J. M.; VILANOVA, A.; ROMENY, B. M. TER HAAR: Interactive Fibre Structure Visualization of the Heart. In: *Computer Graphics Forum* Bd. 28 (2009), Nr. 8, S. 2140–2150

[RHTE99] REZK-SALAMA, C.; HASTREITER, P.; TEITZEL, C.; ERTL, T.: Interactive exploration of volume line integral convolution based on 3D-texture mapping. In: *Proceedings Visualization '99 (Cat. No.99CB37067)* : IEEE, 1999 — ISBN 0-7803-5897-X, S. 233–528

[RoMa05] ROHLFING, TORSTEN; MAURER, CALVIN R.: Multi-classifier framework for atlas-based image segmentation. In: *Pattern Recognition Letters* Bd. 26 (2005), Nr. 13, S. 2070–2079

[SFCN03] SUZUKI, Y.; FUJISHIRO, I.; CHEN, L.; NAKAMURA, H.: Case study: hardware-accelerated selective LIC volume rendering. In: , 2003

[SHSK10] SCHURADE, R; HLAWITSCHKA, M; SCHEUERMANN, B HAMANN G; KNÖSCHE, T R; ANWANDER, A: Visualizing white matter fiber tracts with optimally fitted curved dissection surfaces. In: *Proceedings of Eurographics Workshop on Visual Computing for Biology and Medicine*, 2010, S. 41–48

[SRBG10] SIKACHEV, P.; RAUTEK, P.; BRUCKNER, S.; GRÖLLER, M.E.: Dynamic Focus+Context for Volume Rendering. In: *Proceedings of Vision, Modeling, and Visualization 2010*, 2010 — ISBN 9783905673791, S. 331–338

[TCGC04] TOURNIER, J D; CALAMANTE, F; GADIAN, D G; CONNELLY, A: Direct estimation of the fiber orientation density function from diffusion-weighted MRI data using spherical deconvolution. In: *Neuroimage* Bd. 23 (2004), Nr. 3, S. 1176–1185 — ISBN 1053-8119 (Print)1053-8119 (Linking)

[TCSV15] TAX, CHANTAL M. W.; CHAMBERLAND, MAXIME; VAN STRALEN, MARIJN; VIERGEVER, MAX A.; WHITTINGSTALL, KEVIN; FORTIN, DAVID; DESCOTEAUX, MAXIME; LEEMANS, ALEXANDER: Seeing More by Showing Less: Orientation-Dependent Transparency Rendering for Fiber Tractography Visualization. In: HODAIE, M. (Hrsg.) *PLOS ONE* Bd. 10 (2015), Nr. 10, S. e0139434

[ToCC07] TOURNIER, J D; CALAMANTE, F; CONNELLY, A: Robust determination of the fibre orientation distribution in diffusion MRI: non-negativity constrained super-resolved spherical deconvolution. In: *Neuroimage* Bd. 35 (2007), Nr. 4, S. 1459–1472 — ISBN 1053-8119 (Print)1053-8119 (Linking)

ISBN 978-80-86943-37-4

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

Dynamic Radiosity

Alexandr Shcherbakov
Lomonosov Moscow State University
GSP-1, Leninskie Gory
119991, Moscow, Russia
alex.shcherbakov@graphics.cs.msu.ru

Frolov Vladimir
Lomonosov Moscow State University
Keldysh Institute of Applied Mathematics
(Russian Academy of Sciences)
GSP-1, Leninskie Gory
119991, Moscow, Russia
vfrolov@graphics.cs.msu.ru

ABSTRACT

In this paper we propose a novel radiosity implementation which we called *dynamic radiosity*. By storing and updating local form-factor matrix of the patches closest to the observer we have solved 2 main problems of radiosity algorithm: (1) quadratic complexity of the algorithm and thus difficulty of applying it to a large-scale scenes, and (2) the possibility of changing geometry on the fly (dynamic geometry).

Keywords

Radiosity for large scenes, dynamic geometry.

1 INTRODUCTION

Global Illumination (GI) is a fundamentally difficult computational problem. Thus, real-time applications usually apply approximate algorithms to solve it. Such algorithms can be divided into two main classes: (1) fully dynamic and (2) precomputed radiance transfer. Dynamic GI algorithms allow changing scene (both geometry and materials) each frame. Precomputed Radiance Transfer (PRT) usually immobilizes geometry or both materials and geometry of scene and moves the most complicated computations to *precomputation stage*. PRT gets the best balance of speed and accuracy because of that. The proposed algorithm lies somewhere in between of these two classes.

2 RELATED WORK

2.1 Dynamic GI

Reflective Shadow Maps [1] is a popular solution for interactive global illumination. RSM is a variation of the Instant Radiosity [2] method — global illumination via virtual light sources. The main disadvantage of this approach is the low accuracy of the resulting solution; its advantage is high speed. Although similar methods (called *many lights*) have been developed [3], with the exception of RSM, they are not often used in real-time applications due to bad balance of speed/quality. At high speed they provide low accuracy, and at high quality, they are too far from real time applications [4, 5].

Voxel Cone Tracing [6] uses pre-integrated lighting and Final Gathering via tracing cones. The method is pretty accurate, however, it still suffers of artifacts and it is expensive in terms of computing due to several cones should be traced for each pixel of image.

Light Propagation Volumes [7] numerically solves a differential equation on a three-dimensional grid. The main disadvantage is low accuracy and high memory costs for a regular grid. Cascaded LPV [8] amortizes this cost but does not solve the problems of the method.

With the advent of Nvidia RTX technology [9], Path Tracing in combination with denoising has become a popular method [29, 30, 31, 32]. It is one of the most computationally intensive solutions, however.

Modern dynamic GI approaches compute lighting only around the observer [34]. It allows to apply these methods for a large scenes.

2.2 PRT

Among PRT methods, three main classes should be distinguished: methods based on spherical harmonics [10], radiosity [20], and neural network methods [22].

There are a lot of methods using spherical harmonics: [10, 11, 12, 13, 14, 15]. Their basic idea comes from the Radiance Caching [17]. The harmonics themselves are just a compact storage of the incident light via decomposition of the incident light in the basis of orthogonal functions. Next by analogy with relighting [18], these methods allow replacing the environment and evaluate the lighting dynamically by adding contribution from

the harmonics in the same way that relighting adds contribution from different sources, summing up individual images. These methods are well-suited for outdoor scenes, but fail for indoor ones, since harmonics believe that light comes to a point from infinitely distant objects. In the interiors, this rule is often broken, which leads to various types of artifacts: leakage, incorrect lighting and others [15]. With a proper effort, however, one can achieve a good result for interiors as well [19].

Precomputed Radiance Transfer via neural networks is a relatively new area of research. Authors of [22] show high accuracy in real-time and compact memory representation. The shortcomings of the method include a long pre-calculation and a complete static nature of the scene — neither geometry nor materials could be changed at all (in harmonic or radiosity based methods, for example, these limitations can be relaxed).

2.3 Radiosity

The radiosity method is a physically based approximation for diffuse global illumination [21]. Despite its venerable age, radiosity is widely used in real-time applications [23, 16] and lighting engineering [24, 25].

We chose radiosity because of its physical correctness, relatively high accuracy and speed when small number of patches are used. In addition, ones to be used on practice and well-suited for GPU and hardware implementations since it can be expressed as a single matrix-vector multiplication [26]. On the other hand, the disadvantages of the radiosity are: (1) the dependence on memory and computation as $O(N^2)$ where N is a number of patches and (2) the static geometry of the scene (in this case, unlike the method based on neural networks, materials can be easily replaced).

Hierarchical radiosity [27] subdivides polygons of the original scene into smaller patches until the form-factors for them can be computed with acceptable accuracy (the error is less than a specified threshold), or the maximum depth of the splitting is reached. Thus, a hierarchy of patches is obtained. In the calculation, part of the light is not transferred between the detailed patches, but between the top patches in the hierarchy, which reduces the amount of computation. For some scenes, a very detailed split may be required for good computational accuracy. This method can no longer be expressed in terms of lineal algebra operations and thus it is more complicated than original radiosity for specific GPU or hardware implementations.

Progressive radiosity [28] is a method for gradual calculating of the illumination. Initially, all patches are assigned the same illumination. Gradually, for individual patches, the correct illumination is calculated based on form-factors. The main disadvantage of this method is the difficulty of choosing of criterion for the next lit patch. In this method, each rotation or shift of the camera leads to a complete lighting recalculation. This is a serious disadvantage in comparison with original radiosity that is completely independent from camera parameters.

It is a well known that the radiosity problem can be expressed as a linear equation system problem. By using well known LU (low-upper) decomposition this problem can be solved in $O(N * N)$ operations where N is a number of patches. In [26] more straightforward and GPU friendly algorithm was presented. It was shown that single matrix-vector multiplication can be used to solve multi-bounce radiosity equation.

3 SUGGESTED APPROACH

We suggest a new method for calculating global illumination based on the radiosity method, which we called *dynamic radiosity*. The new method allows to effectively compute the lighting around the observer (Fig. 1), recalculating the local matrix of form-factors. Due to this, we can reduce $O(N^2)$ complexity to $O(M^2)$ (where $M < N$) and add dynamic objects to the scene. We further extend our method to local multibounce matrix [26] which is important contribution of our paper. Following to the approach in [26], we suggest to use the radiosity method for indirect illumination.

Our basic idea is as follows: at first, we restricted area of interest with a subset of patches close to observer (Fig. 1, left). We do not update patches that lie out of this area. Next, we have developed the method for reusing calculations made on previous frame if we know that observer moves only slightly (Fig. 1, right). Thus, we can perform only a small number of calculations every frame, effectively using the information that we have accumulated in time.

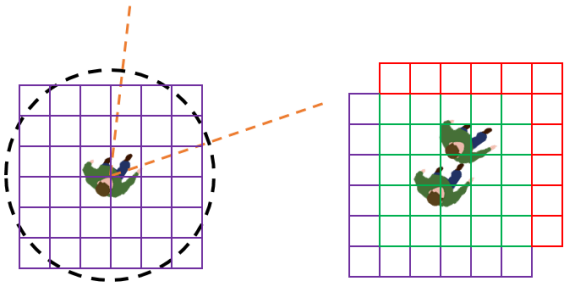


Figure 1: Concept of our method. Red patches (right part of the image) illustrate new patches that we should insert into local matrix. Green patches can be reused.

To start, assume using original radiosity in which we simply have $O(N^2)$ form-factor matrix. Select a subset of M patches and then make M^2 submatrix of form-factors (Fig. 2). We could directly evaluate several radiosity bounces using this M^2 form-factor submatrix,

ISBN 978-80-86943-37-4

but this approach has several disadvantages. First, it requires $B * M^2$ operations per each frame where B is the number of light bounces. Second, this idea has poor utilization of modern computing architectures (both CPU and GPU) because it does not combine computations with memory operations when making local submatrix and does not use the principle of data locality. Due to that we used different approach and worked with multibounce matrix instead of simple form-factor matrix.

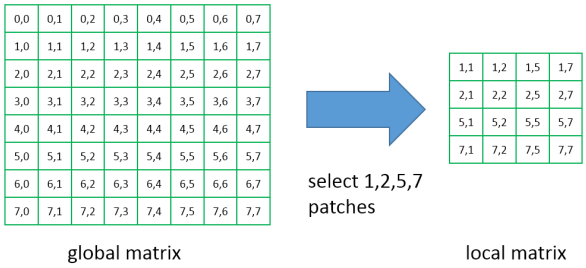


Figure 2: local matrix

The local matrix F_{local} is a multibounce matrix (by analogy with [26]), containing information only about the transfer of light for patches around the observer. The lighting is recalculated when the emission of the patches changes or the position of the observer changes. When the observer's position changes the area of interest also changes.

Light — vector of emission of patches.

Colors — vector of colors of patches.

Idx is a vector of length M, where M is a count of patches in F_{local}.

$Idx_i = k$ where k is index of patch in full form-factors matrix F and i is index of the same patch in local matrix.

$Colors^{local}$ — vector of colors for local patches. $Colors^{local}_i = Colors_{Idx[i]}$

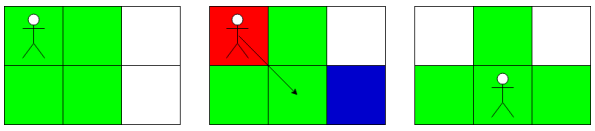


Figure 3: Green patches are included into local matrix (the nearest one to observer). Red patch is farthest out of patches in local matrix. Blue patch is the nearest of excluded patches from a local matrix.

To update the local matrix, the following algorithm is applied within each frame:

1. Select the farthest of the patches **involved** in the local matrix. Let it be indexed as i.

2. Select the patch which is the closest to the observer and **not included** into the local matrix. Let it be indexed as j.

3. If patch j is closer to the observer than patch i, then the matrix is updated, otherwise the update is not required (Fig. 3).

4. In the matrix, the column and row corresponding to patch i are cleared.

5. Calculate the row and column for the j-th patch.

6. The remaining data in the matrix is updated, taking into account the light reflected by patch j.

3.1 Form-factors composition

This method uses the idea of form-factors composition. For patches i, j, k, the value of $F_{ik} * Colors_k * F_{kj}$ is the amount of lighting that excidents from patch j to patch i through patch k.

Thus, the lighting transferred from patch j to patch i through other patches will be equal to:

$$\sum_{k=0}^{N} F_{ik} * Colors_k * F_{kj}.$$

Then, to transfer the lighting from patch i to patch j, taking into account one reflection, it is equal to:

$$\sum_{k=0}^{N} F_{ik} * Colors_k * F_{kj} + F_{ij}.$$

Similarly, it is possible to calculate form-factors that take into account the transfer of light between two patches, taking into account an arbitrary number of reflections from intermediate patches.

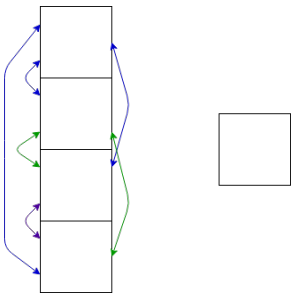


Figure 4: Patches in the local matrix with computed light transfer and the new patch.

3.2 Adding a new patch to local matrix

The process of adding a new patch i (Fig. 4) consists of two parts:

1. Calculation of form-factors for the new patch (just take them from global form factor matrix if geometry is static).

2. Update of the remaining values of the matrix.

ISBN 978-80-86943-37-4

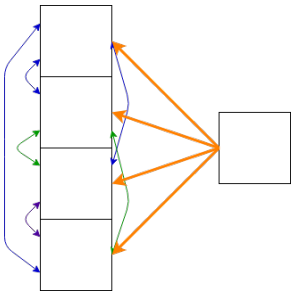


Figure 5: f_c – form-factors from i-th patch to other. Orange arrows are the directions corresponding to outcome form-factors for a new patch.

f_c is a form-factors *column* for the new patch. It contains information about the transfer of light from the i-th patch to the rest (Fig. 5).

f_r is a form-factors *line* for the new patch. It contains values showing which part of lighting is transferred from other patches to the i-th one. (Fig. 6)

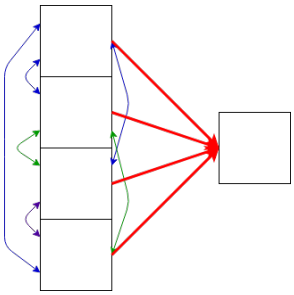


Figure 6: f_r – form-factors from all patches to i-th patch. Red arrows are the directions corresponding to income form-factors for new patch.

For the new patch, we compute the form-factor for lighting emitted by i-th patch to the rest and reflected by them to i-th (Fig. 7).

$$double\_reflection = \sum_{j=0}^{M} f_c[Idx[j]] * f_r[Idx[j]] * Colors_j^{local}. \quad (1)$$

M is the number of patches in the local matrix.

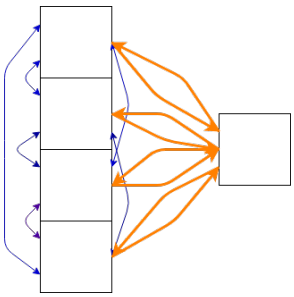


Figure 7: *double_reflection* form-factor for the new patch. Orange arrows shows direction of double light bounce.

double_reflection is taken into account in the vectors g_c and g_r. g_c is a column that takes into account the lighting, moving from the new patch to others, including three light reflections (Fig. 8). g_r is row, which takes into account the lighting that passes to the new patch, with three light reflections too (Fig. 9).

$$g_c = f_c + Colors_i * double\_reflection$$
$$g_r = f_r + Colors_i * double\_reflection$$

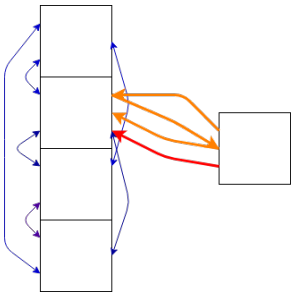


Figure 8: g_c vector of form-factors for excident lighting with information about three reflections. Orange arrows are the path of light in a triple bounce. Red arrow - light from single bounce (f_c).

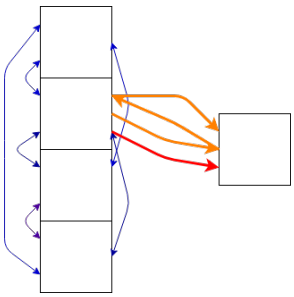


Figure 9: g_c vector of from-factors for incident lighting with information about three reflections. Orange arrows are the paths of light in triple bounce. Red arrow - light from single bounce (f_r).

Additionally, in g_c and g_r, we can add information about one reflection from the new patch and following re-reflection inside the local matrix (Fig. 10, 11):

$$g'_c = g_c + F_{local} \cdot (Colors^{local} \circ g_c)$$
$$g'_r = g_r + (Colors^{local} \circ g_r) \cdot F_{local}$$

g'_c and g'_r are the column and row in the local form-factor matrix for patch i, which take into account at least 3 reflections.

The local matrix of form-factors changes as follows:

$$F'_{local} = F_{local} + g'_r \cdot (Colors_i * g'_c).$$

This transformation adds reflections from the new patch to the matrix.

ISBN 978-80-86943-37-4

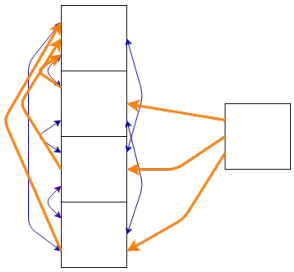


Figure 10: g'_c vector of from-factors for excident lighting from new patch through other patches in local matrix.

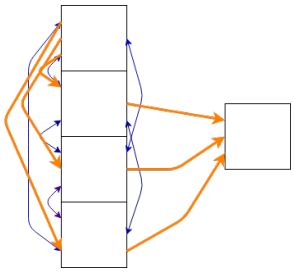


Figure 11: g'_r vector of from-factors for incident lighting from patches in local matrix through their inner reflections to new patch.

Supposed operations allow to recompute lighting in the area of interest, when the observer is moving. This approach can be applied for dynamic objects, but it needs to precalculate and store form-factors for each position of the object.

3.3 Lighting recalculation by local matrix

Lighting is recalculated only for the patches in a local matrix. The calculation itself is similar to the calculations for the multibounce matrix [26]: the local matrix is multiplied by the corresponding emission values of the patches in vector Light.

4 IMPLEMENTATION DETAILS

4.1 Eliminate duplicate reflections

If algorithm deletes a patch from the local matrix, and then adds it back, some patches in the local matrix will receive the reflection information from this patch again. To avoid such errors, we suggest storing a list of patches which lighting is already taken into account for each patch in local matrix. For this, the bitset container can be used. At the same time, it should be updated every time a new patch is added.

4.2 Form-factors streaming for large scenes

Traditional form-factors are used in new patch adding. For large scenes containing a huge amount of non-zero values it may be the problem to store all these numbers in memory.

Proposed method can be combined together with streaming technique for form-factors and load necessary values from HDD/SSD on demand.

5 EXPERIMENT RESULTS

The proposed method was compared with the naive radiosity method and path tracing. Comparisons were made for a scene with a ready-made patching (*rungholt house* from [33]), so the hierarchical radiosity is not applicable to it. The scene consists of 63125 patches. The multibounce matrix takes more than 44 GB for the scene. Such amount of data is not applicable in real-time applications. The local matrix needs less memory (96 MB for $M = 4096$). If fact the size of the memory required (and thus M) can be specified by the user.

5.1 Local multibounce matrix and form-factors sub-matrix

In the worst case our method needs the computation time equal to 3 bounces in naive radiosity for sub-matrix with dimensions $M \times M$. But multibounce can take into account more bounces. Moreover, multibounce matrix needs the same computation as usual form-factors matrix for a single reflection if observer doesn't move too far for matrix recompute.

5.2 Comparison and conclusion

For a local matrix, the lighting computation takes 400 times less time, and updating the local matrix takes 200 times less time than performing the naive radiosity algorithm (Fig. 15, 12). All tests were performed on Intel Core i7-7700HQ CPU in a single thread mode and gain real-time performance with same precision as original radiosity which is relatively close to the reference solution (path tracing, Fig. 12).

We have presented a concept of *Dynamic radiosity*. However, several problems were not considered in this paper and we believe this is our future research: (1) dynamic geometry will involve fast form-factor evaluation, which we didn't implement, (2) GPU implementation is our next area of interest and (3) effitient streaming of data from storages are not well studied yet. Finally (4), our algorithm does not account lighting from excluded patches at all. This can be fixed via cascades by analogy with Cascaded Light Propagation Volumes.

5.3 Acknowlegments

This work is sponsored by RFBR 18-31-20032.

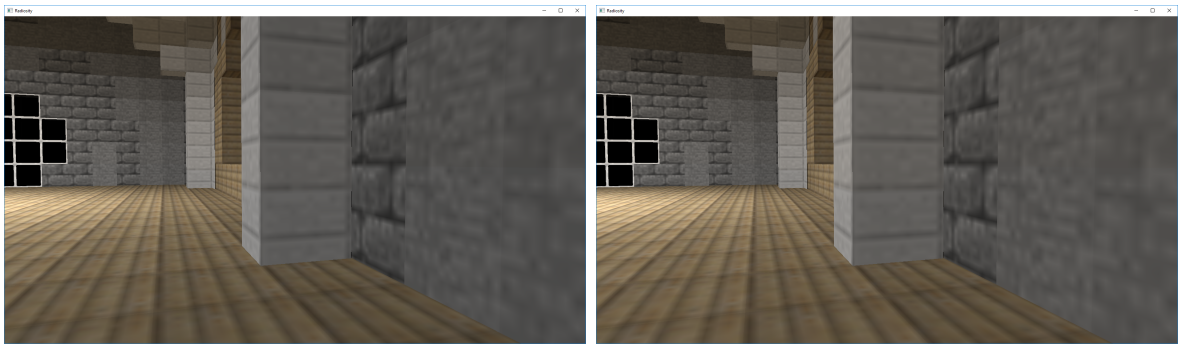
6 REFERENCES

[1] Dachsbacher C., Stamminger M. *Reflective shadow maps* // Proceedings of the 2005 symposium on Interactive 3D graphics and games. — ACM, 2005. — Ð¡. 203-231.

ISBN 978-80-86943-37-4

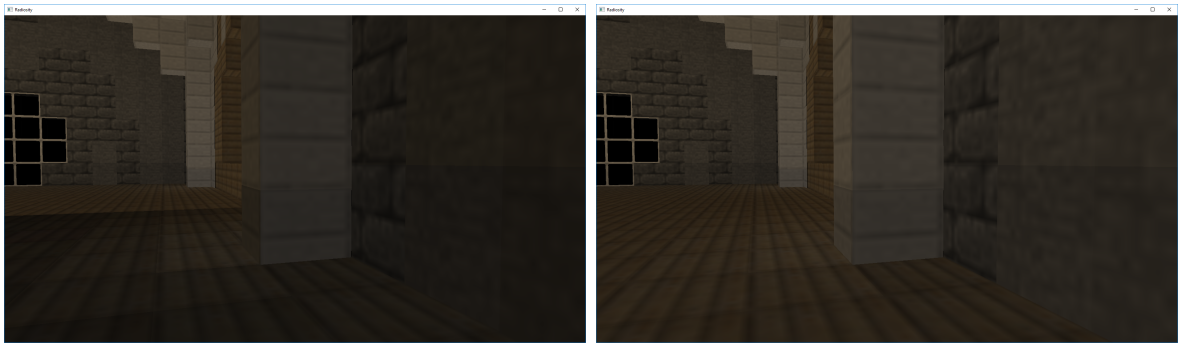
Our method 1024 patches (12 ms) Naive radiosity (1561 ms) Path Tracing (10 minutes)

Figure 12: Comparison of our method to naive radiosity and reference (path tracing).



Our method 1024 patches (12 ms) Naive radiosity (1561 ms)

Figure 13: Comparison of our method to naive radiosity. Patches around the observer.



Our method 1024 patches (12 ms) Naive radiosity (1561 ms)

Figure 14: Comparison of indirect lighting computed by our method and naive radiosity. Patches around the observer.

[2] Keller A. *Instant radiosity.* — 1997.

[3] Carsten Dachsbacher, Jaroslav Krivanek, Milos Hasan, Adam Arbree, Bruce Walter, and Jan Novak. 2014. *Scalable Realistic Rendering with Many-Light Methods.* Comput. Graph. Forum 33, 1 (February 2014), 88-104. DOI=http://dx.doi.org/10.1111/cgf.12256

[4] Walter, B., Fernandez, S., Arbree, A., Bala, K., Donikian, M., and Greenberg, D. P. (2005, July). *Lightcuts: a scalable approach to illumination.* In ACM Transactions on graphics (TOG) (Vol. 24, No. 3, pp. 1098–1107). ACM.

[5] Ou, J., and Pellacini, F. (2011). *LightSlice: matrix slice sampling for the many-lights problem.* ACM Trans. Graph., 30(6), 179-1.

[6] Crassin, C., Neyret, F., Sainz, M., Green, S., and Eisemann, E. (2011, September). *Interactive indirect illumination using voxel cone tracing.* In Computer Graphics Forum (Vol. 30, No. 7, pp. 1921–1930). Oxford, UK: Blackwell Publishing Ltd.

[7] Kaplanyan A. *Light propagation volumes in cryengine 3* // ACM SIGGRAPH Courses. 2009. Vol. 7. p. 2.

ISBN 978-80-86943-37-4

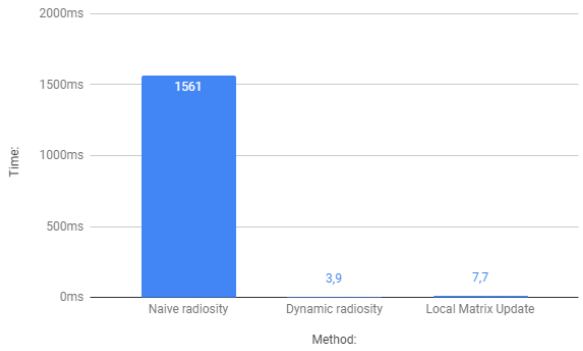


Figure 15: Detailing of computation time.

[8] Kaplanyan A., Dachsbacher C. *Cascaded light propagation volumes for real-time indirect illumination* // Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games. — ACM, 2010. — pp. 99–107.

[9] KELLER A., MCGUIRE M., MUNKBERG J., PHARR M., SHIRLEY P., WALD I., WYMAN C. *Ray Tracing Gems. HIGH-QUALITY AND REAL-TIME RENDERING WITH DXR AND OTHER APIS.* . 2019. ISBN-13 (pbk): 978-1-4842-4426-5 ISBN-13 (electronic): 978-1-4842-4427-2 https://doi.org/10.1007/978-1-4842-4427-2.

[10] Green R. *Spherical harmonic lighting: The gritty details* //Archives of the Game Developers Conference. 2003. Vol. 56. p. 4.

[11] Papaioannou G. *Real-time diffuse global illumination using radiance hints* //Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics. ACM, 2011. pp. 15-24.

[12] Peter-Pike Sloan, Jan Kautz, and John Snyder. *Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments.* ACM Trans. Graph. 21, 3 (July 2002), 527-536. DOI: https://doi.org/10.1145/566654.566612

[13] Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. *Clustered principal components for precomputed radiance transfer.* ACM Trans. Graph. 22, 3 (July 2003), 382-391. DOI: https://doi.org/10.1145/882262.882281

[14] Kautz, Jan, Jaakko Lehtinen, and Peter-Pike Sloan. *SIGGRAPH Precomputed Radiance Transfer: Theory and Practice course*. Aug. 2005.

[15] Akenine-Moller T., Haines E., Hoffman N. *Real-time rendering, 4th Edition*. AK Peters/CRC Press, 2018. pp 480 – 490.

[16] Akenine-Moller T., Haines E., Hoffman N. *Real-time rendering, 4th Edition*. AK Peters/CRC Press, 2018. pp 482.

[17] Krivanek J. et al. *Radiance caching for efficient global illumination computation* //IEEE Transactions on Visualization and Computer Graphics. 2005. Vol. 11. Number. 5. pp. 550–561.

[18] Dorsey, J., Arvo, J., Greenberg, D. *Interactive design of complex time dependent lighting. IEEE Computer Graphics and Applications.* 1995. 15(2), 26-36.

[19] McGuire, M., Mara, M., Nowrouzezahrai, D., Luebke, D. (2017, February). *Real-time global illumination using precomputed light field probes.* In Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (p. 2). ACM.

[20] Sillion F. X. et al. *Radiosity and global illumination.* San Francisco : Morgan Kaufmann, 1994. Vol. 1.

[21] Cohen M., GreenBurg D. *The Hemi-cube: A Radiosity solution for complex it is hard to achieve environments* // Proceedings of SIGGRAPH 85 in Computer Graphics, 1985. 19. number 3. pp. 31-40.

[22] Peiran Ren, Jiaping Wang, Minmin Gong, Stephen Lin, Xin Tong, and Baining Guo. 2013. *Global illumination with radiance regression functions.* ACM Trans. Graph. 32, 4, Article 130 (July 2013), 12 pages. DOI: https://doi.org/10.1145/2461912.246200

[23] Sam Martin. *Enlighten real-time radiosity*. SIGGRAPH 2011.

[24] Mangkuto, R. A. *Validation of DIALux 4.12 and DIALux evo 4.1 against the Analytical Test Cases of CIE 171*. 2006. Leukos, 12(3), 139-150.

[25] Yu X., Su Y., Chen X. *Application of RELUX simulation to investigate energy saving potential from daylighting in a new educational building in UK* // Energy and Buildings. 2014. Vol. 74. pp. 191–202.

[26] Shcherbakov, A., and Frolov, V.*Accelerating radiosity on gpus.* In WSCG'2017 Full papers proceedings (2017), 2701, Computer Science Research Notes 2701 Pilsen, Czech Republic, pp. 99–105.

[27] Pat Hanrahan, David Salzman, and Larry Aupperle. *A rapid hierarchical radiosity algorithm.* SIGGRAPH Comput. Graph. 25, 4 (July 1991), 197-206.

[28] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. *A progressive refinement approach to fast radiosity image generation.* In Proceedings of the 15th annual conference on Computer graphics and interactive techniques (SIGGRAPH '88), Richard J. Beach (Ed.). ACM, New York, NY, USA, 75-84. DOI:

ISBN 978-80-86943-37-4

https://doi.org/10.1145/54852.378487

[29] Martin Stich. *Real-time raytracing with NVIDIA RTX*. GAMESCOM, KOLN. 2018. URL = http://on-demand.gputechconf.com/gtc-eu/2018/pdf/e8527-real-time-ray-tracing-with-nvidia-rtx.pdf

[30] Michael Mara, Morgan McGuire, Benedikt Bitterli, and Wojciech Jarosz. *An efficient denoising algorithm for global illumination.* In Proceedings of High Performance Graphics (HPG '17). ACM, New York, NY, USA, Article 3, 7 pages. DOI: https://doi.org/10.1145/3105762.3105774

[31] A. M. Gruzdev, V. A. Frolov, and A. V. Ignatenko. 2015. *Practical approach to the fast Monte-Carlo ray-tracing.* Program. Comput. Softw. 41, 5 (September 2015), 253–257. DOI=http://dx.doi.org/10.1134/S0361768815050035

[32] D. D. Zhdanov, I. S. Potemin, V. A. Galaktionov, B. Kh. Barladyan, K. A. Vostryakov, and L. Z. Shapiro. *Spectral Ray Tracing in Problems of Photorealistic Imagery Construction* // Programming and Computer Software, Vol. 37, No. 5, 2011, pp. 236–244. DOI: 10.1134/S0361768811050069

[33] Morgan McGuire. *Computer Graphics Archive.* 2017. URL = https://casual-effects.com/data.

[34] A. Yudintsev. *Scalable Real-Time Global Illumination for Large Scenes.* 2019. URL = https://www.gdcvault.com/play/1026469/Scalable-Real-Time-Global-Illumination

 ISBN 978-80-86943-37-4

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

Corotated meshless implicit dynamics for deformable bodies

Jean-Nicolas Brunet
Inria Nancy Grand-Est
jnbrunet2000@gmail.com

Vincent Magnoux
École polytechnique de Montréal
vincent.magnoux@polymtl.ca

Benoît Ozell
École polytechnique de Montréal
benoit.ozell@polymtl.ca

Stéphane Cotin
Inria Nancy Grand-Est
stephane.cotin@inria.fr

ABSTRACT

This paper proposes a fast, stable and accurate meshless method to simulate geometrically non-linear elastic behaviors. To address the inherent limitations of finite element (FE) models, the discretization of the domain is simplified by removing the need to create polyhedral elements. The volumetric locking effect exhibited by incompressible materials in some linear FE models is also completely avoided. Our approach merely requires that the volume of the object be filled with a cloud of points. To minimize numerical errors, we construct a corotational formulation around the quadrature positions that is well suited for large displacements containing small deformations. The equations of motion are integrated in time following an implicit scheme. The convergence rate and accuracy are validated through both stretching and bending case studies. Finally, results are presented using a set of examples that show how we can easily build a realistic physical model of various deformable bodies with little effort spent on the discretization of the domain.

Keywords

Meshless methods; Physically-based modelling; Interactive simulation; Point-based models

1 INTRODUCTION

Simulating realistic deformations of a soft virtual object is a complex task that can quickly transform into a considerable challenge when the simulation needs to be performed in a close to real-time framerate. Starting from a 3D representation of a deformable object, the model must account for all external forces such as gravity, collision impacts and other elements that alter the surface of the object. This can be achieved by building a volumetric representation of the object and by deriving the displacement field according to the continuum mechanics laws in a discretized space. This approach is usually referred to as a *physics-based simulation* framework [NMK*06]. Hence, the object's deformation results from the natural equilibrium between external forces being captured by the simulation process and elastic properties (mainly the resistance to stretching and compression) of the object's material. The complexity of the method then greatly depends on the application for which the simulation process is designed and more importantly, the performance criteria to be achieved.

1.1 Four key performance criteria

Over the last decade, a large number of engineering and medical computer simulation applications have been developed. For this type of application, the *accuracy* of the solution is usually the main performance criterion imposed on simulation methods. Simply stated, the main concern here is *how close is the virtual representation to the object's real deformation*? Since this strongly depends on the physical laws governing the simulation model, this condition can also be translated as *how do the states of deformation compare to a theoretical solution*? Physics-based methods are usually best suited where high accuracy is required.

Other applications such as interactive modelers or virtual simulators require quick updates of the various deformation states. This brings two other performance criteria into play, namely the *speed* and the *stability* of the simulation. In this work, the speed constraint is expressed as either a real-time requirement (around 30 frames per second) or a close to real-time (around 1 frame per second) requirement. The stability constraint, on the other hand, relates to the robustness of the simulation process and its response to unexpected external forces.

The *simplicity* of the simulation framework is our fourth and last criterion. The objective here is to adequately correlate the accuracy of the solution produced by the simulation process to the amount of configuration work and technical knowledge required by the end-user.

To be effective, the simulation framework must provide an adequate balance between these four criteria. More specifically, the method must be accurate. It must also

be well suited for interactive modeling and must be flexible enough to handle the different properties of elastic materials. Finally, the model must be simple enough to allow inexperienced users to easily configure and run the simulation with minimum assistance.

1.2 Current methods

Methods based on finite elements (FE) currently form one of the most popular physics-based simulation frameworks. Using a linear FE approach and a small displacement assumption, [BC96] and [CDA99] have pioneered the field of real-time surgical simulators. To minimize the impact of rotations, [Fel00] and [NPF05] have proposed a method, called the corotational FEM, to extract a rotation matrix from the displacement of elements. Their work has opened the door to a whole new branch of FEM-based tools for interactive simulation. Unfortunately, these methods come with some drawbacks: the simulated object's volume must be pre-filled with well-formed and well-placed elements. In addition, for nearly incompressible materials, FE methods are affected by volume locking effects which lead to numerical errors that directly impact the accuracy criterion. To overcome this situation, the user must not only discretize the volume with elements, but must also make sure that these elements will not generate numerical errors. The complexity of this approach compromises another of our performance criteria, i.e. the simplicity of the framework.

To address the inherent limitations of FE models in deformable object animation, a well-established branch of solutions is gaining momentum: *meshless methods*. With these frameworks, the discretization of the domain is simplified by removing the need to create polyhedral elements. They also prevent the volumetric locking effect exhibited by incompressible materials [BLG94]. Under normal conditions, the meshless approach merely requires that the volume of the object be filled with a cloud of points, frequently called particles.

Meshless methods have already been proposed in both the Computer graphics & Animations and Computational mechanics communities. [HWJM10] has proposed a fully non-linear element-free Galerkin method [BLG94] for surgical simulation. Although no mention was made with respect to real-time compliance, all indications are that they were the first to propose a fast enough Galerkin-based solution without element-based approximations. Since then, their work has been carried over to various surgical applications [MHJW12; ZWJ*14; DRTZ16; DLLW16; WGJ*16]. While these meshless methods are accurate, their use of fully non-linear material is time-consuming and not well adapted for applications requiring a real time or close to real-time framerate. Since they use an explicit time integration scheme, they are also restricted to small and regular time step intervals which implies a less robust solution in an interactive environment.

[MKN*04] presented a completely mesh-free method based on a moving least squares (MLS) approximation and a particle density approach to integrate the elastic energy equations in space. Using the same approach, [SSP07] also presented a complete mesh-free method, but this time using a smoothed particle hydrodynamics (SPH) approximation. Unlike [MKN*04], their shape function does not rely on the inverse of a moment matrix and allows for co-linear and co-planar neighborhoods. [BIT09; PGBT18] extended this strategy by carrying the corotational principle to the SPH formulation. While these meshless methods present visually plausible results, their volumetric integration approach is based on an approximation of the particle densities. This reduces the precision of the simulation and induces instabilities. It is especially true given that their particles represent both the degrees of freedom and the integration points.

Beside physics-based simulation frameworks, geometry-based methods such as [SCL*04; MHTG05] are often proposed where only the surface representation of simulated objects matters. While these approaches produce visually pleasing results, they do not satisfy our accuracy criterion where material properties must be correctly simulated anywhere inside the simulated object.

1.3 Our proposed solution

In this paper, we propose to address the problem of 3D deformable object simulation using a meshless method whereby each object is defined only by their surface meshes. The solution borrows concepts from both the FE and meshless models. But unlike the other solutions mentioned earlier, our approach relies on two basic adjustments that are directly driven by our four performance criteria.

Firstly, unlike meshless density-based integration methods, our approach relies on background quadrature grids where the intersection between the surface mesh and grid cells are used to accurately estimate the interior domain's volume. The objective here is to both improve the integration process of the Galerkin approach and reduce numerical errors inherent to large non-linear displacements by applying the corotational principle.

Secondly, since the effects of nonlinear rotations are minimized by the corotational principle, we propose to exploit a linear approximation of the stress and strain measures. This then allows for close to real time framerates. We also propose an implicit time integration scheme for large time steps that are better suited for interactive simulations.

 ISBN 978-80-86943-37-4

1.4 Paper outline

In Section 2, we present a brief overview of the mathematical framework that was selected to model material elasticity and deformation. The basic features of our proposed method are then discussed in Section 3. Among other things, we show how, from the updated positions of a set of integration points, the rotational part of a displacement can be extracted. We also expose the system assembly stages for both a static solving scheme and an implicit time integration scheme. This is followed by the introduction of a method to correctly map the vertices of the object surface mesh to the particle positions. We then explain how collision forces gathered from the surface mesh can also be mapped onto the particles. In Section 4 we present a series of case studies to measure the convergence rate and precision of our proposed model. We conclude this paper in Section 5 with some suggestions for future work.

2 MODELING ELASTIC MATERIALS

To simulate the elastic behavior of an object, a set of rules governing the shape of an object over time must be used. For a body that is stretched or compressed by external forces, whether from a gravitational field or from a collision with another object, the rules ensure that an equilibrium state is always maintained between these forces and the body's deformation resistance dictated by the intrinsic material properties. The rules also involve two key parameters to describe the stiffness of the material: i) Young's modulus E (N/m^2) which defines the lengthwise resistance to stretch and compression; and ii) Poisson's ratio ν which quantifies the perpendicular expansion (respectively the transverse compression) of the body's volume during compression (respectively stretch). A material is said to be incompressible when its Poisson's ratio approaches the limit of 0.5.

The rules that we selected for our simulation framework are derived from Cauchy's first law of motion, which states the conservation of linear momentum in a continuum. Here, Cauchy's stress tensor $\boldsymbol{\sigma}$ conveys the amount of stress (N/m^2) sustained by the material undergoing a certain deformation. Under the small strain hypothesis, if $\boldsymbol{u} = [u\ v\ w]^T$ is a displacement vector from position $\boldsymbol{x}^0$ in the undeformed state of the body to its deformed position $\boldsymbol{x} = \boldsymbol{x}^0 + \boldsymbol{u}$, the deformation of a material can then be approximated by the linear strain tensor :

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) \tag{1}$$

where $\nabla \boldsymbol{u}$ is the displacement gradient.

Using a constitutive model that follows Hooke's law of elasticity, we can define the stress tensor $\boldsymbol{\sigma}$ explicitly as a linear function of the displacement $\boldsymbol{u}$:

$$\boldsymbol{\sigma}(\boldsymbol{u}) = 2\mu\boldsymbol{\varepsilon} + \lambda tr(\boldsymbol{\varepsilon})\boldsymbol{I} \tag{2}$$

where $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$ and $\mu = \frac{E}{2(1+\nu)}$ are the Lamé coefficients. It is important to note that equations 1 and 2 are linearized versions of their counterparts found in finite strain theory. This means that the stress and strain tensors can only represent relatively small and linear displacements of the body. However, they can be computed faster and, as we will see later, they can be precalculated at the beginning of the simulation. Also, we will later present a method to minimize the effects of this linearization when nonlinear transformations (rotations) are encountered.

Cauchy's first law of motion can finally be translated into a system of partial differential equations which pose our set of rules for the simulation process:

$$-\nabla \cdot \boldsymbol{\sigma} = \boldsymbol{f} \Rightarrow \begin{cases} -(\frac{\partial \sigma_{11}}{\partial x} + \frac{\partial \sigma_{12}}{\partial y} + \frac{\partial \sigma_{13}}{\partial z}) &= f_x \\ -(\frac{\partial \sigma_{21}}{\partial x} + \frac{\partial \sigma_{22}}{\partial y} + \frac{\partial \sigma_{23}}{\partial z}) &= f_y \\ -(\frac{\partial \sigma_{31}}{\partial x} + \frac{\partial \sigma_{32}}{\partial y} + \frac{\partial \sigma_{33}}{\partial z}) &= f_z \end{cases} \tag{3}$$

where $\boldsymbol{f}$ is the external force. Finding the deformed shape of an elastic object is then reduced to the problem of solving for the unknown displacements $\boldsymbol{u}$ of the partial differential equations 3. This can be viewed as a *static simulation* as it does not involve any time integration scheme.

For a *dynamic simulation* that involves time dependent terms such as the gravitational force, equation 3 becomes:

$$\rho \frac{d^2\boldsymbol{x}}{dt^2} + \boldsymbol{f}^{\text{elastic}}(\boldsymbol{x},t) = \boldsymbol{f}^{\text{ext}}(\boldsymbol{x},t) \tag{4}$$

where ρ is the material density, $\boldsymbol{f}^{\text{elastic}}(\boldsymbol{x},t) = -\nabla\boldsymbol{\sigma}(\boldsymbol{u})$ is the internal elastic force and $\boldsymbol{f}^{\text{ext}}(\boldsymbol{x},t)$ is the external force. In this case, finding the deformed shape of an elastic object requires numerically integrating equation 4 over time.

The next section presents the complete process of computing the elastic force $\boldsymbol{f}^{\text{elastic}}(\boldsymbol{x},t)$ and the description of an implicit method to integrate equation 4 over time.

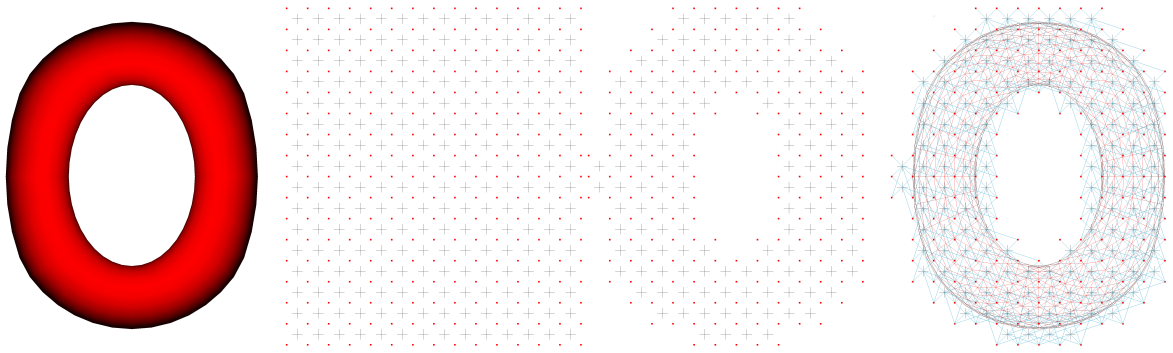
3 OUR PROPOSED METHOD

In the previous section, we described a linear relationship between stress and infinitesimal strain in a *continuous domain*. In order to solve the unknown displacement field $\boldsymbol{u}$ of our object, and given that we do not have an explicit definition of it, we propose using an approach that relies on the Galerkin method, which we now outline.

3.1 The Galerkin method

The Galerkin method uses a weak formulation of the discrete partial differential equations to be solved. To simplify the reading, we temporarily disregard the time dependent terms of our equations and refer only to the

 ISBN 978-80-86943-37-4



(a) (b) (c) (d)

Figure 1: Volumetric discretizations of a 3D surface. (a) Surface mesh provided by the user. (b) Background grid where the grid's cubes are used to place the DOFs and the integration points. (c) DOFs and integration points are cropped to fit the surface mesh. (d) A neighborhood of the closest particles is built around each integration point.

static case. By multiplying equation 3 by a test function $\boldsymbol{w}$ in the Sobolev solution subspace $H^1(\Omega_0)^3$, and by following Green's theorem, the set of equations becomes

$$\underbrace{-\int_{\Omega_0} \boldsymbol{\sigma}(\boldsymbol{u}) \cdot \delta\boldsymbol{\varepsilon} d\Omega_0}_{\Pi_{elastic}} = \underbrace{\int_{\Omega_0} \boldsymbol{b} \cdot \boldsymbol{w} d\Omega_0 + \int_{\Gamma_0} \boldsymbol{t} \cdot \boldsymbol{w} d\Gamma_0}_{\Pi_{ext}} \tag{5}$$

where Ω_0 is the initial (undeformed) domain, Γ_0 is the domain boundary, $\delta\boldsymbol{\varepsilon}$ denotes the variation of the strain tensor and $\boldsymbol{t} = \boldsymbol{\sigma} \cdot \boldsymbol{n}$ is the surface normal traction vector. Here the left term $\Pi_{elastic}$ can be viewed as the internal virtual work and Π_{ext} as the work related to external loads.

3.2 Volumetric discretization: the FE approach

In finite element methods, the initial domain Ω_0 is discretized into a set of polyhedral elements (usually tetrahedrons or hexahedrons). These elements serve two important purposes: i) to construct an interpolation function of the displacement anywhere in the domain, and ii) to numerically integrate the continuous equations. The FE approach begins by building an explicit geometrical representation of an element domain Ω_e. From this representation, an interpolation $\boldsymbol{u}_e(\boldsymbol{x})$ of the displacement inside every element e with respect to its nodes is assembled. The displacement $\boldsymbol{u}(\boldsymbol{x})$ of any position $\boldsymbol{x} \in \Omega_0$ then becomes the displacement $\boldsymbol{u}_e(\boldsymbol{x})$ where element e is the one surrounding $\boldsymbol{x}$. By gathering all element nodes into a set called *degrees of freedom* (DOFs), the problem of solving for the unknown displacement field $\boldsymbol{u}(\boldsymbol{x})$ is thus reduced to one of solving for a finite vector of n unknown displacements $[\boldsymbol{u}_0, \boldsymbol{u}_1, ..., \boldsymbol{u}_n]^T$. Furthermore, placing one or more Gauss integration point inside the elements provides a means of numerically estimating the integral terms of equation 5 since elements usually conform to the surface, hence producing an accurate volumetric representation (the sum of all integration point volumes should equal the total volume of the object).

3.3 Volumetric discretization: the meshless approach

Whereas FE methods use the nodes of a polyhedral element to interpolate the displacement at Gauss points, meshless methods instead create an approximation of the displacement by considering the values of nearby points. The idea is thus to fill the interior volume of the object with an evenly distributed cloud of points, *the particles*. These particles represent the DOFs of the system and, consequently, the unknown displacements to be solved. The approximation is built by using shape functions ϕ that are evaluated at every particle near a given position. The value of the displacement $\boldsymbol{u}(\boldsymbol{x})$ becomes:

$$\boldsymbol{u} \approx \tilde{\boldsymbol{u}} = \sum_{i \in V(\boldsymbol{x})} \phi_i(\boldsymbol{x})\, \boldsymbol{u}_i \tag{6}$$

where $V(\boldsymbol{x})$ is the set of particles in the vicinity of position $\boldsymbol{x}$. Here, $\tilde{\boldsymbol{u}}$ is an approximation since our shape function does not offer the Kronecker delta property at the nodes, and is therefore not a true interpolant [BKO*96].

For the volumetric integration of equation 5, we use a background grid of regular cubic elements that completely covers the domain. The Gauss points within these volume elements, *the integration points*, are used for numerical integration of the equation. The summation of the volume of every integration point must always be very close to the total interior volume of the simulated object. The integration over the continuous domain Ω_0 becomes:

$$\int_{\Omega_0} f(\boldsymbol{u}) d\Omega_0 \approx \sum_I v_I f(\tilde{\boldsymbol{u}}_I) \tag{7}$$

where v_I is the volume of integration point I and $\tilde{\boldsymbol{u}}_I$ is the approximation of the displacement at position of I.

ISBN 978-80-86943-37-4

Using equation 6, the displacement of any integration point is estimated by accumulating the weighted displacements of the particles surrounding it. The shape functions we selected for our proposed framework are similar to those described in [HWJM10] and are found by using a moving least squares approach to minimize weighted residuals from the polynomial approximation $u^h(\boldsymbol{x}) = \boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{a}(\boldsymbol{x})$. To conform with time constraints imposed by interactive simulations, we use linear basis functions $\boldsymbol{p} = [1\ x\ y\ z]$. Solving for the unknown coefficients $\boldsymbol{a}$, the shape function and its derivative become:

$$\begin{aligned}\phi_i &= \boldsymbol{P}_I \boldsymbol{A}^{-1} W_{Ii} \boldsymbol{P}_i \\ \phi_{i,x} &= [\boldsymbol{P}_{I,x}\boldsymbol{A}^{-1}W_{Ii} + \boldsymbol{P}_I(\boldsymbol{A}_{,x}^{-1}W_{Ii} + \boldsymbol{A}^{-1}W_{Ii,x})]\boldsymbol{P}_i \end{aligned} \quad (8)$$

where $\boldsymbol{P}_I = \boldsymbol{p}(\boldsymbol{x}_I)$, $\boldsymbol{A} = \sum_{j\in V(\boldsymbol{x}_I^0)} W_{Ii}\boldsymbol{P}_j\boldsymbol{P}_j^T$ and with $W_{Ii} = W(\left\|\boldsymbol{x}_I^0 - \boldsymbol{x}_j^0\right\|, h)$ is a monotonic and decreasing weight function on a distance threshold of h, beyond which it becomes null. In this work, we used the quartic spline weight function described in [HWJM10] where h is found by taking the mean distance of the k nearest neighbors of a position $\boldsymbol{x}$ and multiplying it by a small dilatation factor.

3.4 The corotational nodal elastic forces

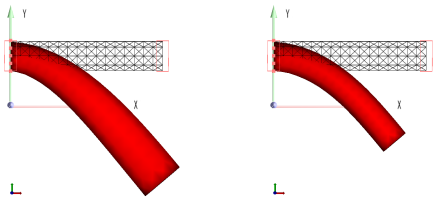
Similar to FE methods, and because the continuous integration terms in equation 5 are discretized into a sum over the integration points (equation 7), the continuous system is now reduced to a set of smaller systems of equations to be solved around each neighborhood. Before we describe the process of solving for the unknown displacements $\boldsymbol{u}$, let us start by assuming that they are already known. To derive the elastic forces of equation 4 applied to a particle i in the neighborhood of an integration point I, we look at the rate of change of internal elastic energy in the direction of $\boldsymbol{u}_i$, yielding:

$$\begin{aligned}\boldsymbol{f}_{I\to i}^{\text{elastic}} &= v_I \boldsymbol{\sigma}(\boldsymbol{u}_I)\cdot\delta\boldsymbol{\varepsilon} \\ &= v_I[\lambda(\nabla\cdot\boldsymbol{u}_I)\boldsymbol{I} + 2\mu\varepsilon(\boldsymbol{u}_I)]\cdot\delta\boldsymbol{\varepsilon} \\ &= v_I\boldsymbol{B}_i^T\boldsymbol{C}\sum_{j\in V(I)}[\boldsymbol{B}_j\ \boldsymbol{u}_j]\end{aligned} \quad (9)$$

where $\boldsymbol{C}_{ijkl} = \lambda\boldsymbol{I}_{ij}\boldsymbol{I}_{kl} + \mu(\boldsymbol{I}_{ik}\boldsymbol{I}_{jl} + \boldsymbol{I}_{il}\boldsymbol{I}_{jk})$ is the elasticity tensor, often reduced to a 6x6 matrix, and where $\boldsymbol{B}_i$ is the strain matrix and is defined as

$$\boldsymbol{B}_i^T = \begin{bmatrix}\phi_{i,x} & 0 & 0 & \phi_{i,y} & 0 & \phi_{i,z} \\ 0 & \phi_{i,y} & 0 & \phi_{i,x} & \phi_{i,z} & 0 \\ 0 & 0 & \phi_{i,z} & 0 & \phi_{i,y} & \phi_{i,x}\end{bmatrix} \quad (10)$$

Normally, a rigid transformation (translation or rotation) of the simulated body should not generate any elastic force since there is no deformation involved. Unfortunately, since both the strain tensor and constitutive model are linear approximations, rotations, which are non-linear transformations, will wrongly generate internal forces, often called "ghost forces" (see figure 2a).



(a) Ghost forces prevent the cylinder to rotate down.

(b) The corotational approach alleviates the effects of ghost forces by removing rotations before the computation of elastic forces.

Figure 2: Simulation of a cylinder fixed at the left and deformed by gravity (downward along the Y axis).

In order to minimize these ghost forces, we extend the work of [NPF05] and [BIT09] by introducing a corotational approach to our method. Whereas in FEM the rotation is extracted from an element, and in the SPH formulation of [BIT09] the forces are gathered around the particles, our method relies on the integration point neighborhood. Since these quadrature positions are not represented by unknown variables, their position does not get updated throughout the simulation. To solve this, we manually update their positions at the beginning of each time step by using the displacement of neighboring particles (equation 6). Using these updated positions, we construct a transformation matrix of each integration point subspace:

$$\boldsymbol{A}_I = \sum_{j\in V(\boldsymbol{x}_I^0)} v_j(\boldsymbol{x}_j - \boldsymbol{x}_I)(\boldsymbol{x}_j^0 - \boldsymbol{x}_I^0)^T \quad (11)$$

where v_j is the volume of a particle and is obtained by splitting the volume of the integration points evenly among its neighbors. We then use the stable SVD decomposition method of [PTVF07] to extract a rotation matrix $\boldsymbol{R}_I$ from the transformation matrix $\boldsymbol{A}_I$. Finally, we cancel this rotation from the displacement approximation in equation 9 to get the corotational forces:

$$\boldsymbol{f}_{I\to i}^{\text{elastic}} = v_I\boldsymbol{R}_I\boldsymbol{B}_i^T\boldsymbol{C}\sum_{j\in V(I)}[\boldsymbol{B}_j\ (\boldsymbol{R}_I^T\boldsymbol{x}_j - \boldsymbol{x}_j^0)] \quad (12)$$

The benefits of this approach are shown in figure 2b where ghost forces no longer appear.

3.5 Solving the static system

In the previous section, we described how to compute the localized elastic force $\boldsymbol{f}_{I\to i}^{\text{elastic}}$ applied to a given particle i in the vicinity of integration point I. The total elastic force on i is found by accumulating the contribution of every integration point. This elastic force definition is only useful when the current displacements $\boldsymbol{u}_j$

ISBN 978-80-86943-37-4

are known. Typically, in a static scheme (where there are no time dependent terms), the displacements are unknown, which implies that a linear system of equations must be solved in order to find them. When linear strain and stress tensors are used, displacements $\boldsymbol{u}$ can be factored to obtain the system $\boldsymbol{K}\ \boldsymbol{u} = \boldsymbol{f}^{\text{ext}}$ where $\boldsymbol{K}$ is the stiffness matrix and is constant in time. However, since our method relies on displacements relative to updated integration point positions due to our corotated approach, the stiffness matrix is no longer constant. Therefore, solving for the unknown displacements is done using an iterative Newton-Raphson method. Starting from an initial displacement, $\boldsymbol{u}^0$, we try through an iterative process to find a correction $\boldsymbol{\delta}_u$ that balances the linearized set of equations after n iterations:

$$\dot{\boldsymbol{K}}^{n-1}\ \boldsymbol{\delta}_u^n = \boldsymbol{f}^{\text{elastic}}(\boldsymbol{u}^0 + \boldsymbol{\delta}_u^{n-1}) + \boldsymbol{f}^{\text{ext}} \tag{13}$$

where $\dot{\boldsymbol{K}}$ is the tangent stiffness matrix obtained by deriving the force of equation 12 with respect to the displacement of the neighbors. The 3x3 sub-matrix $\dot{\boldsymbol{K}}_{ij}$ can be viewed as the linear action of the displacement $\boldsymbol{u}_j$ on the particle i:

$$\dot{\boldsymbol{K}}_{ij} = \sum_I v_I \boldsymbol{R}_I \boldsymbol{B}_i^T \boldsymbol{C} \boldsymbol{B}_j^T \boldsymbol{R}_I^T \tag{14}$$

where I represents an integration point influencing both particles i and j.

3.6 Solving the dynamic system

For dynamic schemes, time-dependent terms must be incorporated into the equations. For example, the gravity force involves the acceleration of particles, the damping force involves their velocity and collision forces involve the current state of the surfaces. To solve this time-dependent system, we decided to follow the Euler implicit time integration method [BW98]. Using the discrete formulation $\boldsymbol{M}\boldsymbol{a} - \boldsymbol{f}^{\text{elastic}} = \boldsymbol{f}^{\text{ext}}$, we derive the acceleration $\boldsymbol{a}$ and velocity $\boldsymbol{v}$ from the following equations:

$$\begin{aligned} \boldsymbol{M}\boldsymbol{a}^{t+\Delta t} &= \boldsymbol{f}^{\text{elastic}}(\boldsymbol{x}^{t+\Delta t}) + \boldsymbol{f}^{\text{ext}}(\boldsymbol{x}^{t+\Delta t}) + \boldsymbol{D}\boldsymbol{v}^{t+\Delta t} \\ \boldsymbol{v}^{t+\Delta t} &= \boldsymbol{v}^t + \Delta t \boldsymbol{a}^{t+\Delta t} \\ \boldsymbol{x}^{t+\Delta t} &= \boldsymbol{x}^t + \Delta t \boldsymbol{v}^{t+\Delta t} \end{aligned} \tag{15}$$

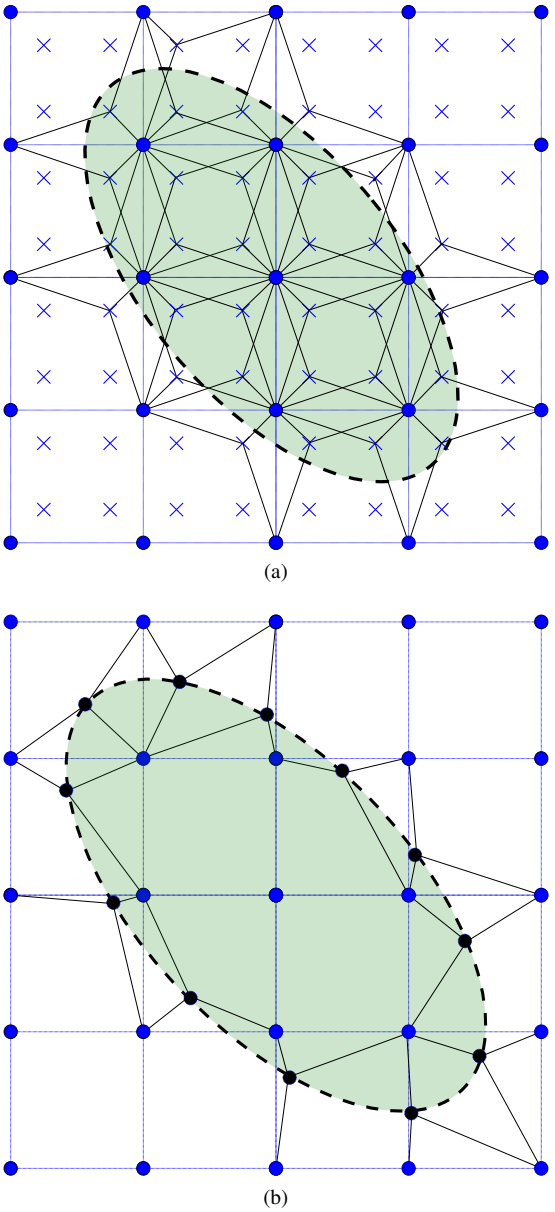
where $\boldsymbol{M}$ is a diagonal lumped mass matrix of the particles and $\boldsymbol{D}$ is a constant Rayleigh damping matrix. Since the forces at time $t+\Delta t$ are unknown, the following linear equation is obtained from a first order Taylor approximation :

$$(\boldsymbol{M} - \Delta t \boldsymbol{D} - \Delta t^2 \dot{\boldsymbol{K}})\boldsymbol{a}^{t+\Delta t} = \Delta t(\boldsymbol{f}^{\text{elastic}}(\boldsymbol{x}^t) + \boldsymbol{f}^{\text{ext}}(\boldsymbol{x}^t)) + \Delta t^2 \dot{\boldsymbol{K}} \boldsymbol{v}^t \tag{16}$$

This equation is finally solved using the iterative conjugate gradient method.

3.7 Surface displacement and external force mapping

So far our attention has been confined to the interior of the deformable body. The last and final step consists of extending the displacements derived in the previous steps to the object's surface using the shape functions. Since the surface of an object is usually represented by a mesh of triangles or quads, the displacement of these polygon vertices can be derived by evaluating the shape function of their neighboring particles (see figure 3b).



(a) (b)

Figure 3: The similarities between (a) the relation between particles and integration points, and (b) the mapping of particles and the surface tesselation. Here, the green oval shape is the simulated object, the blue nodes represent the unknown degrees of freedom, the black nodes are the surface vertices and the blue crosses are the integration points.

ISBN 978-80-86943-37-4

While this method is trivial and very fast, it should be noted that other methods based on an implicit representation (see [MKN*04]) can also be used to produce even more visually satisfying results. However, given the objectives set earlier in this paper, we consider the displacement approach to be sufficient.

For the external forces applied to the surface mesh (such as collision and external pressure), the surface nodal forces are applied to the neighboring particles following the same general idea. Thus, for an external force $\boldsymbol{f}_s$ applied to a surface vertex at the initial position $\boldsymbol{x}_s^0$, the mapped force $\boldsymbol{f}_i^{\text{ext}}$ at a neighboring particle i becomes

$$\boldsymbol{f}_i^{\text{ext}} = \phi_i(\boldsymbol{x}_s^0)\boldsymbol{f}_s \tag{17}$$

4 RESULTS

In this section, we seek to demonstrate how our method positions itself with respect to the four criteria pursued in this work. Beginning with bending and stretching scenarios, the solutions are validated through convergence and precision analyses. Next, we outline the stability and simplicity of the method using various experiments involving multiple objects and materials. The computation times are given and were measured on an Intel(R) Core(TM) i7-6700K CPU @ 4.00 GHz computer with 16 GB of memory. No multithreading maneuvers were used, hence leaving place for future speed improvements. Our method was implemented as a plugin for the multi-physics open source SOFA Framework [FDD*12].

4.1 Convergence analysis

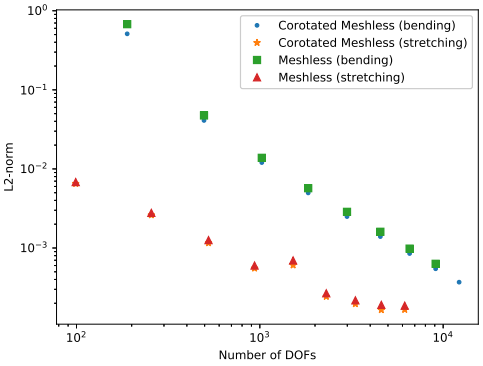
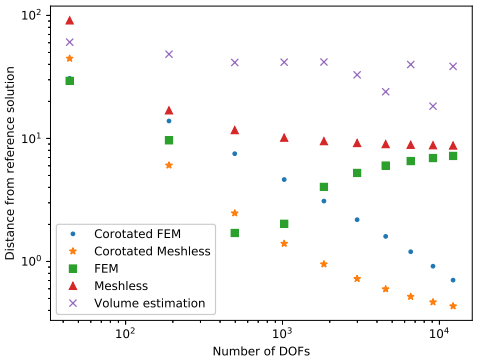
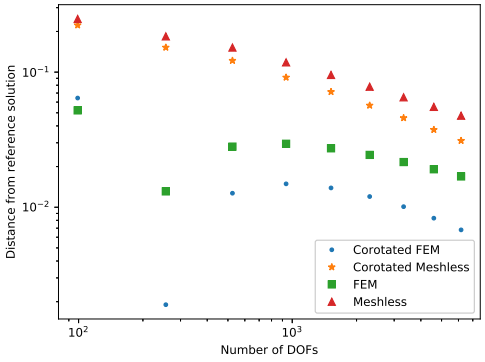


Figure 4: The convergence rate of our method in different scenarios. The error norm is obtained by comparing the n DOFs solution against the $(n-1)$ DOFs solution of the same variant of the method in the same scenario. The straight curve indicates a constant rate of convergence.

To verify that the implementation of our method works adequately from a numerical standpoint and that it results in accurate deformations, we performed



(a)



(b)

Figure 5: The accuracy of the solutions for the (a) bending and (b) stretching scenarios against a corotated FEM reference solution of 69k regular hexahedral elements.

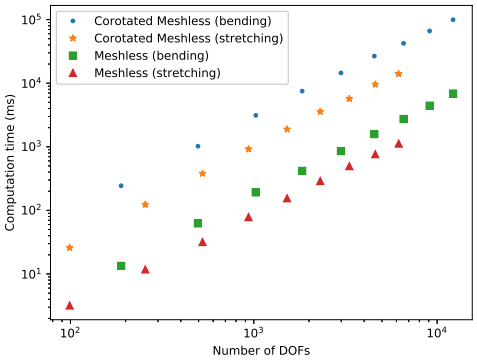


Figure 6: The computational times of the experiments

a convergence analysis using a FE solution as a reference. The first scenario was a regular beam of size $20x20x200\ mm^3$ placed horizontally, fixed at one end and subjected to a downward pressure force of $12\ N/mm^2$ at the other. The material used a Young modulus value of $50\ N/mm^2$ and a Poisson ratio of 0.45.

ISBN 978-80-86943-37-4

From this experiment, we gathered two measurements. The first one, presented in figure 4, shows the convergence rate of our method. It shows that, as the number of DOFs increases, the static solution tends towards a unique solution at a constant rate, whether that solution is accurate or not. The second one, presented in figure 5a, validates the accuracy of the converged solution against a reference solution. In this work, we used a corotated FEM solution of nearly 70k regular hexahedral elements as a reference.

To demonstrate issues with estimating the volume of an object without a background grid, we also simulated the beam with a nodal integration method where node mass and volume were estimated by sampling neighboring nodes as in [MKN*04]. While the estimated density is very accurate, the corresponding mass and volume are much higher than the actual ones and result in higher stiffness of the simulated object. Furthermore, the estimation is affected by particle distribution and will thus vary as the number of DOFs changes. This means that the simulation will not converge to a specific solution as we increase the number of DOFs.

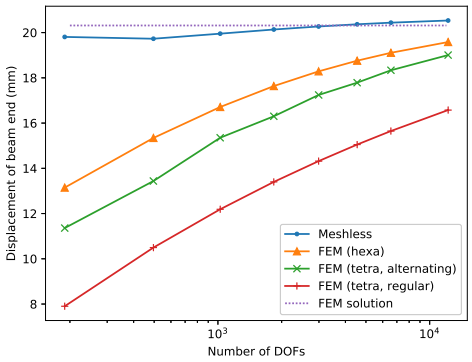


Figure 7: The "locking" artifact of linear tetrahedrons when used on an almost incompressible bending beam. The distances of displacement from the tip of the solutions are shown.

The bending of a beam is also a good way to illustrate one of the inherent meshing problems that come with FE methods. As those methods need to generate a mesh that conforms to the boundary surface, tetrahedron meshes are generally preferred over hexahedrons since they are well suited for automatic meshing methods. However, the solutions of linear tetrahedron-based FEM can vary a lot: the under-integrating aspect during the computation of the stiffness matrix can introduce large numerical errors, also known as the "locking" effect. By building two tetrahedron FE models from the regular hexahedron one, where both have 6 tetrahedrons per cube, but with different orientations, we can clearly see how this numerical error impacts the solution. Both have exactly the same number of elements, system unknowns and element shapes. However, as shown in figure 7, they converge to different solutions. In both cases, these tetrahedrons would have been accepted as "well-shaped elements" by most automatic meshing software. We can also see how our method and the hexahedron one does not suffer from this numerical constraint. Conversely, while the hexahedron FE method is trivial to implement for a squared cross-section beam, it remains very hard to apply to general complex objects. This demonstrates an attractive benefit of our methods over traditional FE methods.

In figure 4 and 5b, the experiment is repeated but this time through a stretching scenario. Here, using the same previously used material, a pressure force exerting 1500 N/mm^2 was used in a direction parallel to the beam. The computation times from both bending and stretching scenarios are outlined in figure 6. To solve the static systems, we used the Newton-Raphson method described in equation 13 with a maximum of 100 iterations and a residual threshold of 0.00001.

4.2 Multiple materials

To illustrate the process of discretizing an object with different material properties, figure 8 shows a cylinder fixed at its center and deformed by gravity. Here, two background grids of different material properties were placed side by side, illustrating the simplicity of setting the different material properties.

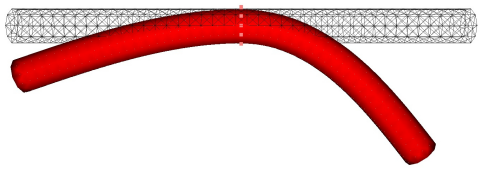


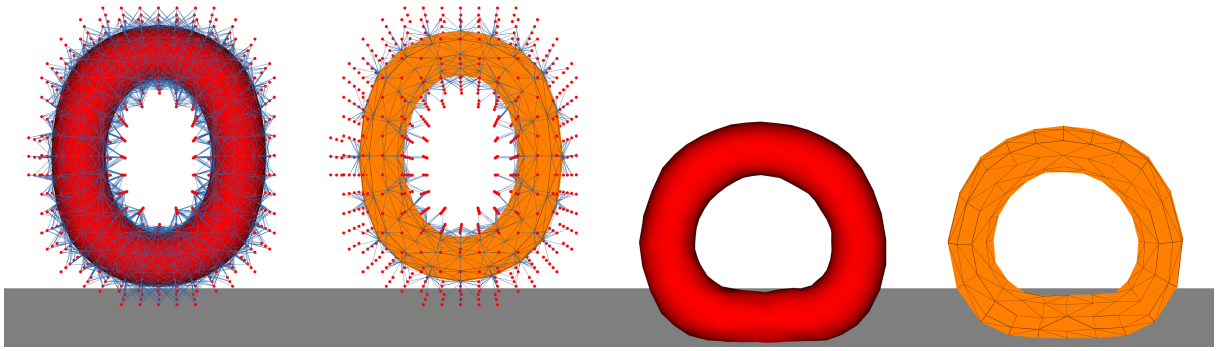
Figure 8: Material properties can vary inside a single object. The cylinder is fixed at its center and gravity is applied. The left half of the cylinder has a much higher Young's modulus than the right half.

We can imagine how this could be extended to complex objects where some parts must be stiffer or heavier than the others. A gradient of the material properties could also be added around the boundary regions of the different parts, smoothing out the change in material. The inherent simplicity of dealing with a cloud of particles then allows for a lot of flexibility to the user, either for determining how the objects should behave or to improve the accuracy of the solution in some specific regions of the object.

4.3 Surface mapping

The result of mapping the surface representation onto the degrees of freedom is best demonstrated visually, in figure 9. This figure shows the various object representations that are manipulated during a simulation. The master state, represented by the degrees of freedom (red

ISBN 978-80-86943-37-4



(a) (b) (c) (d)

Figure 9: Mapping between a surface and the deformation state of a torus. (a-b) Relation between the master state (red circles) and its slave surfaces: the visual (red) and contact (yellow) tesselations. (c-d) Resulting mapped surfaces after the contact with the floor.

circles in figures 9a and 9b) describes the deformation of the object, while visual and contact model representations completely depend on that state to get their final shapes (figures 9c and 9d). The contact forces can be obtained through different methods. In this work, a simple penalty based contact force was sufficient for our demonstration purposes, even when used with collisions between deformable objects as shown in figure 10.

5 CONCLUSION

We have presented a method for animating deformable objects at interative framerates that require no polyhedral meshing of the volume. Where finite element methods require well-formed and well-placed elements inside their domain, our method only needs a cloud of points to solve the system of unknown displacements. Unlike other meshless methods based on nodal density integration, we use a regular background grid which does not need to conform to the surface mesh to efficiently integrate the Galerkin formulation of elasticity PDEs to be solved, hence improving the accuracy of the solution. To minimize numerical errors and by using the integration points as a local reference frame, a rotation matrix was extracted from the neighborhood of those frames and used to reduce the effects of large nonlinear displacements. We have shown that this method is both stable and accurate by presenting convergence and precision analyses. We have also presented images directly extracted from various simulations involving collisions. These time dependent simulations followed an implicit time integration scheme that is well suited for interactive applications incorporating large and possibly inconsistent steps.

While this method is promising, there is however room for improvement. Since the computation involves neighborhoods containing more nodes than its FE method counterparts, the resulting computation time is a little bit heavier. Conversely, this can be greatly improved by using multi-core computers and exploiting the symmetry in elementary stiffness matrices by node numbering optimizations. The benefits of this method could also be explored in topological change scenarios, such as cutting and plastic deformations. Finally, we plan on doing extended convergence analyses to establish optimal neighborhood configuration based on the sparsity of the nodes and integration points to further improve convergence rates.

6 REFERENCES

[BC96] Bro-Nielsen, Morten and Cotin, Stéphane. "Real-time volumetric deformable models for surgery simulation using finite elements and condensation". *Computer Graphics Forum* 15 (1996), 57–66.

[BIT09] Becker, Markus, Ihmsen, Markus, and Teschner, Matthias. "Corotated sph for deformable solids". *Natural Phenomena*. 2009, 27–34.

[BKO*96] Belytschko, T, Krongauz, Y, Organ, D, et al. "Meshless Methods: An Overview and Recent Developments". *Computer Method in Applied Mechanics and Engineering* 139 (1996), 3–47.

[BLG94] Belytschko, Ted, Lu, Yun Yun, and Gu, Lei. "Element-free Galerkin methods". *International Journal for Numerical Methods in Engineering* 37.2 (1994), 229–256.

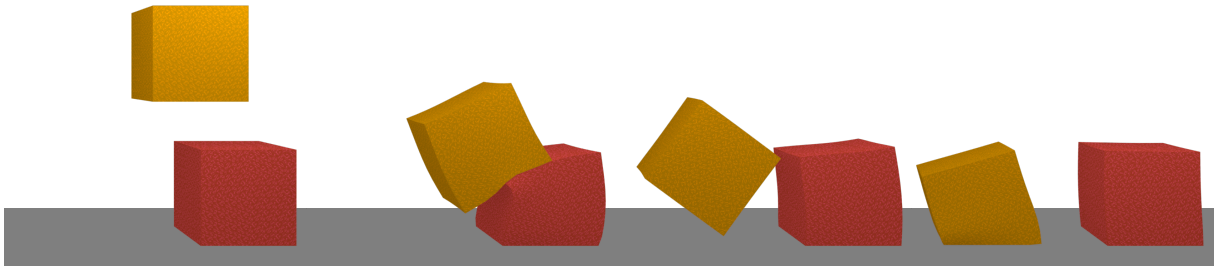
[BW98] Baraff, David and Witkin, Andrew. "Large steps in cloth simulation". *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*. 1998, 43–54.

[CDA99] Cotin, Stéphane, Delingette, Hervé, and Ayache, Nicholas. "Real-time elastic deformations of soft tissues for surgery simulation". *IEEE Transactions on Visualization and Computer Graphics* 5.1 (1999), 62–73.

[DLLW16] Dong, Yi, Liu, Xuemei, Li, Hairui, and Wang, Zhenkuan. "A Nonlinear Viscoelastic Meshless Model for Soft Tissue Deformation". *2016 International Conference on Virtual Reality and Visualization (ICVRV)* (2016), 204–211.

[DRTZ16] Dehghan, Mohammad Reza, Rahimi, Abdolreza, Talebi, Heidar Ali, and Zareinejad, Mohammad. "A three-dimensional large deformation model for soft tissue using meshless method". *International Journal of Medical Robotics and Computer Assisted Surgery* 12.2 (2016).

 ISBN 978-80-86943-37-4



(a) (b) (c) (d)

Figure 10: Collision between two deformable objects. Both cubes are deformed while colliding with each other.

[FDD*12] Faure, François, Duriez, Christian, Delingette, Hervé, et al. *SOFA; a Multi-Model Framework for Interactive Physical Simulation*. Vol. 11. 2012, 283–321.

[Fel00] Felippa, Carlos A. *A systematic approach to the element-independent corotational dynamics of finite elements*. Tech. rep. Technical Report CU-CAS-00-03, Center for Aerospace Structures, 2000.

[HWJM10] Horton, Ashley, Wittek, Adam, Joldes, Grand Roman, and Miller, Karol. "A meshless Total Lagrangian explicit dynamics algorithm for surgical simulation". *International Journal for Numerical Methods in Biomedical Engineering* 26.8 (2010), 977–998.

[MHJW12] Miller, K., Horton, A., Joldes, G. R., and Wittek, A. "Beyond finite elements: A comprehensive, patient-specific neurosurgical simulation utilizing a meshless method". *Journal of Biomechanics* 45.15 (2012), 2698–2701.

[MHTG05] Müller, Matthias, Heidelberger, Bruno, Teschner, Matthias, and Gross, Markus. "Meshless deformations based on shape matching". *ACM transactions on graphics (TOG)*. Vol. 24. 3. ACM. 2005, 471–478.

[MKN*04] Müller, Matthias, Keiser, Richard, Nealen, Andrew, et al. "Point based animation of elastic, plastic and melting objects". *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association. 2004, 141–151.

[NMK*06] Nealen, Andrew, Müller, Matthias, Keiser, Richard, et al. "Physically based deformable models in computer graphics". *Computer graphics forum*. Vol. 25. 4. Wiley Online Library. 2006, 809–836.

[NPF05] Nesme, Matthieu, Payan, Yohan, and Faure, François. "Efficient, Physically Plausible Finite Elements". *Eurographics* (2005), 77–80.

[PGBT18] Peer, Andreas, Gissler, Christoph, Band, Stefan, and Teschner, Matthias. "An implicit SPH formulation for incompressible linearly elastic solids". *Computer Graphics Forum*. Vol. 37. 6. Wiley Online Library. 2018, 135–148.

[PTVF07] Press, William H, Teukolsky, Saul a, Vetterling, William T, and Flannery, Brian P. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 2007, 1262.

[SCL*04] Sorkine, Olga, Cohen-Or, Daniel, Lipman, Yaron, et al. "Laplacian surface editing". *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. ACM. 2004, 175–184.

[SSP07] Solenthaler, Barbara, Schläfli, Jürg, and Pajarola, Renato. "A unified particle model for fluid-solid interactions". *Computer Animation and Virtual Worlds*. Vol. 18. 1. 2007, 69–82.

[WGJ*16] Wittek, Adam, Grosland, Nicole M., Joldes, Grand Roman, et al. "From Finite Element Meshes to Clouds of Points: A Review of Methods for Generation of Computational Biomechanics Models for Patient-Specific Applications". *Annals of Biomedical Engineering* 44.1 (2016), 3–15.

[ZWJ*14] Zhang, G. Y., Wittek, A., Joldes, G. R., et al. "A three-dimensional nonlinear meshfree algorithm for simulating mechanical responses of soft tissue". *Engineering Analysis with Boundary Elements* 42 (2014), 60–66.

 ISBN 978-80-86943-37-4

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

Straight Skeleton Computation Optimized for Roof Model Generation

Kenichi Sugihara
Gifu Keizai University
5-50 Kitagata-chou
Ogaki city, Gifu-Pref., 503-8550, Japan
sugihara@gifu-keizai.ac.jp

ABSTRACT

3D building models with roofs are important in several fields, such as urban planning and BIM (Building Information Model). However, enormous time and labor are required to create these 3D models. In order to automate laborious steps, a GIS and CG integrated system is proposed for the automatic generation of 3D building models, based on building polygons (building footprints) on digital maps. The generation is implemented through straight skeleton computation, in which three events ('*Edge*' and '*Split*', '*Vertex*' events) were proposed. In the computation process, usually three edges propagate into a node. Often it causes an acute angle shape that is not appropriate for roof boards. To avoid the inappropriate shape, in this paper, methodologies are proposed for adding '*Line segment*' events besides the conventional events, and monotone polygon nodes sorting.

Keywords

automatic generation, 3D building model, straight skeleton, building footprint, GIS.

1 INTRODUCTION

3D town models, such as the one shown in Fig.1 right, are important in urban planning and architectural design, e.g., BIM (Building Information Model). However, enormous time and labor are required to create these 3D models, using 3D modeling software such as 3ds Max or SketchUp. For example, when manually modeling a house with roofs by Constructive Solid Geometry (CSG), one must follow the following laborious steps:

(1) Generation of geometric primitives of proper size, such as boxes, prisms or polyhedra that will form parts of a house **(2)** Boolean operations are applied to these primitives to form the shapes of parts of a house such as making rectangular holes in a building body for doors and windows **(3)** Horizontal and vertical rotation of parts of a house **(4)** Placing the parts of a house to appropriate positions **(5)** Texture mapping onto these parts.

In order to save these laborious steps, a GIS and CG integrated system that automatically generates 3D building models is proposed, based on building polygons (building footprints) on a digital map as shown in Fig.1 left and Fig.7a, which show most building polygons' edges meet at right angles (orthogonal polygon). An orthogonal polygon can be divided or separated into a set of rectangles. The proposed integrated system divides orthogonal building polygons into a set of rectangles and places rectangular roofs and box-shaped building bodies on these rectangles. In the digital map, however, building polygons are not always orthogonal. In either orthogonal or non-orthogonal polygons, the methodology is proposed for automatically creating general shaped roofs by the straight skeleton computation defined by Aichholzer et al. [Aic95].

In their proposal, two events ('*Edge*' and '*Split*' events described in section 4) will occur during shrinking process. Besides two events, Eppstein et al. [Epp99] suggested a '*Vertex*' event in which two or more reflex vertices reach the same point simultaneously. A reflex vertex is a vertex whose internal angle is greater than 180 degrees. However, some roofs are not created by these three events proposed. In our paper, the methodology was proposed for constructing roof models by assuming 'the *Third* event' in which a reflex vertex runs into the edge, but the other split polygon is collapsed into a node (an *Edge* event happens in the *Split* event at the same time).

In this paper, a methodology is proposed for adding a '*Line segment*' event besides the conventional events.

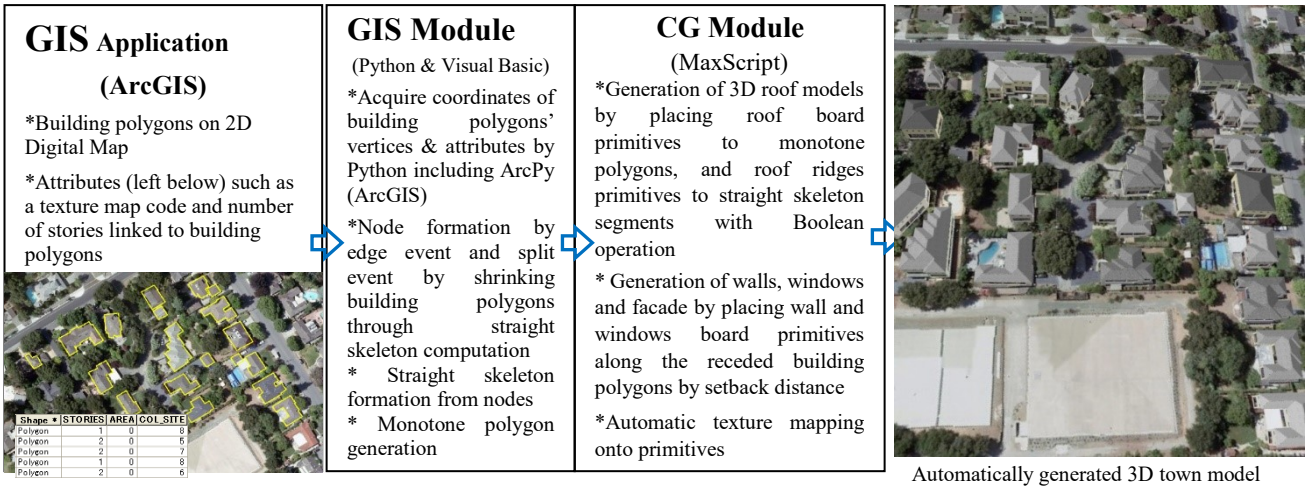


Figure 1: Pipeline of Automatic Generation for 3D Building Models by Straight Skeleton Computation

The shrinking process continues if polygons split have non-zero area. The shrinking process ends when polygons split fall into '*Vertex*' or '*Line segment*' since they have no area. Consequently, a '*Line segment*' can be a resulting shape of shrinking procedure, and we will classify a '*Line segment*' event in which two line segments collapse into one line segment.

Usually three edges propagate into a node. Often it causes an acute angle shape that is not appropriate for roof boards shown in Fig.3. To avoid the inappropriate shape, a '*Line segment*' event will be proposed for straight skeleton computation. We also propose 'monotone polygon nodes sorting' by which not self-intersecting monotone polygons are formed, where 'monotone polygons' are the areas divided by a straight skeleton, as shown in Fig.2c.

2 RELATED WORK

Since 3D building models are used for several different purposes, such as urban planning, archaeological reconstruction and game industries, various types of technologies, ranging from computer vision, computer graphics, photogrammetry, and remote sensing, have been proposed and developed for creating 3D building models. When 3D building models are used for urban planning or reconstruction of ancient cities, we will make non-existent building models which we cannot take a picture, and therefore these models are not created by CV, photogrammetry, and remote sensing. So, we focus on procedural modeling, especially roof creation by straight skeleton computation. Procedural modeling is an effective technique to create 3D models from sets of rules such as L-systems, fractals, and generative modeling language [Par01]. Mueller et al. [Mue06] have created an archaeological site of Pompeii and a suburbia model of Beverly Hills by using a shape grammar. They import building footprints from a GIS database and try to classify imported polygons as basic shapes in their shape vocabulary. In case of misclassification, they use a general extruded footprint together with a general roof obtained by the straight skeleton algorithm defined by a continuous shrinking process [Aic95].

The straight skeleton algorithm is useful in generating hipped roofs, since its resulting 'monotone polygons' correspond to the roof boards of a hipped roof. The downside is that the generated roof is only one of many possible roofs, and there is no way to get another roof [Ede14]. To overcome the downside, a new generalization of straight skeletons is proposed by Helda et al. [Hel17], introducing additively-weighted straight skeletons. An additively-weighted straight skeleton is the result of a wavefront-propagation process where wavefront edges do not necessarily start to move at the begin of the propagation, resulting in an automated generation of roofs in which the individual facets have different inclinations and start at different heights.

By using the straight skeleton, Kelly et al. [Kel11] present a user interface for the exterior of architectural models to interactively specify procedural extrusions. They use a profile editor which controls the sweeping of a plane from the base of the building footprint, and they finally construct a wide variety of roofs and a complex architecture.

By these interactive modeling, 3D building models with plausible detailed façade can be achieved. However, the limitation of these modeling is the large amount of user interaction involved [Jia09], and the models created are 'surface models' by sweeping or extruding dependent on edited profiles, or revolving 2D primitive geometries. However, when creating 3D building models for architectural design and BIM, 3D building models should be made up of solid geometries primitives which will be parts of the building, created through Boolean operation. Thus, the

ISBN 978-80-86943-37-4

GIS and CG integrated system that automatically generates 3D building models immediately by CSG (Constructive Solid Geometry) is proposed. Although surface models can have 'beautifully curved roof' by interactive procedural modeling, in reality these roofs are consisted of hundreds of narrow flat boards in most building design. These narrow boards will be properly placed along the roof curve.

3 PIPELINE of AUTOMATIC GENERATION

As the pipeline of automatic generation is shown in Fig.1, the source of 3D models is a digital map that contains building polygons linked with attributes data, such as the number of stories and the type of roof, shown in Fig.1 left below. The maps are then preprocessed at the GIS module, and the CG module finally generates the 3D building model.

The preprocessing at the GIS module includes the procedures as follows: **(1)** Calculate the minimum receding distance for an *Edge* event (including a *Third* and *Line segment* event). Until the *Edge* event occur, check if *Split* event happens by starting continuous shrinking process. **(2)** Start continuous shrinking process in which edges of the polygon move inward, parallel to themselves at a constant speed (Fig.2a&2b). **(3)** Detect any event such as a *Split*, *Edge* or *Line segment* event during shrinking process, and formation of nodes by these events. The position of the node is calculated by the intersection of angular bisectors. **(4)** Inherit and store three or more original edges' ID (e.g. *edgN* in Fig.2a) linked to the node during the shrinking process in which the topology of the polygon will change. In shrinking process, Fig.2b shows *edg2* firstly disappears into *Node1*, and two edges (*edg8 & 9*) secondly result in *Node2*. Since at least three original edges sweep into the node, *edg1,2 & 3* propagation result in *Node1,* and *edg4,5 & 10* propagation result in *Node3* (by *Split event*). **(5)** Every (original) edge will inquire 'each node' having three or more ID to find out which node has the same original edge ID. If so, then nodes of the same ID are collected and the set of nodes are sorted according to the edge vector to form 'monotone polygon' and the straight skeleton. **(6)** Calculate the length, width, center position and inclination of the bounding rectangle for 'monotone polygon'. **(7)** Export the coordinates of polygons' vertices, 'monotone polygons' information, and attributes of buildings.

In these procedures, the areas divided by a straight skeleton are called as 'monotone polygons' shown in Fig.2c, and to get 'monotone polygons', the set of the nodes belonging to the same original edge will be aligned depending on the coordinate value on the axis parallel to each original edge vector (the 'node vector projections' onto the original edge vector). These nodes are coplanar and will form roof boards for a 3D building model.

As shown in Fig.1, the CG module receives the pre-processed data that the GIS module exports, generating 3D building models. In GIS module, the system measures the length and inclination of the bounding rectangle for the monotone polygon that will be a roof board. The CG module generates a bounding box of the length and width, measured in GIS module. The monotone polygons will be converted into primitives, i.e., thin boxes by Boolean operation between the extrusion of the monotone polygon and the box primitive.

In case of modeling a building with roofs, the CG module follows these steps: **(1)** Generate primitives of appropriate size, such as boxes, prisms or polyhedra that will form the various parts of the house. **(2)** Boolean operations applied to these primitives to form the shapes of parts of the house, for examples, making holes in a building body for doors and windows,

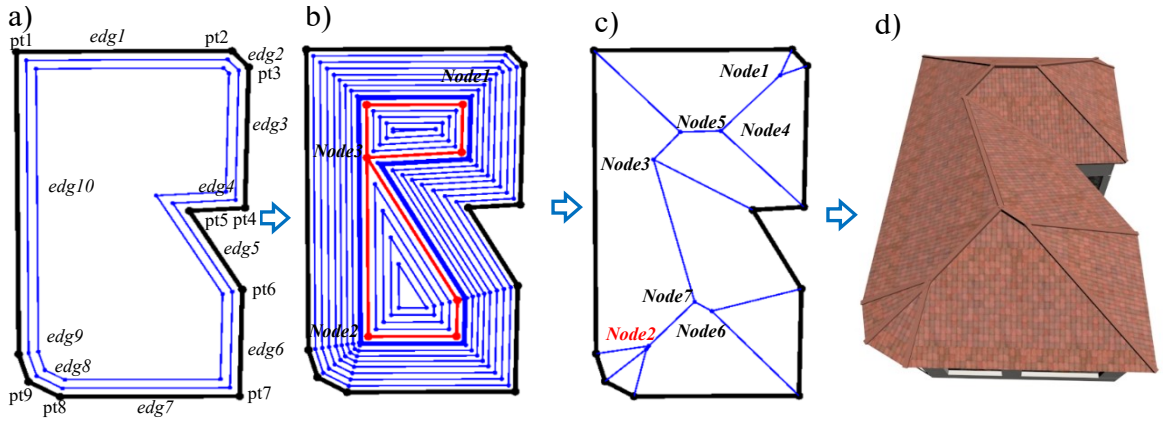


Figure 2: Shrinking process and a straight skeleton, a roof model generated. a) Input polygon (bold) start continuous shrinking process in which edges of the polygon move inward, parallel to themselves at a constant speed. **b)** Shrinking polygons (blue) by no event, and red one by a split event. **c)** The straight skeleton (blue) and monotone polygons. **d)** A roof model automatically generated: each roof board is based on an 'monotone polygon'.

ISBN 978-80-86943-37-4

making trapezoidal roof boards for a hipped roof and a temple roof. **(3)** Rotate parts of the house according to the inclination of the partitioned rectangle. **(4)** Place parts of the house. **(5)** Texture mapping onto these parts according to the attribute received. **(6)** Copy the 2nd floor to form the 3rd floor or more in case of building higher than 3 stories.

CG module has been developed using Maxscript that controls 3D CG software (3ds MAX, Autodesk Inc).

4 STRAIGHT SKELETON COMPUTATION

Aichholzer et al. [Aic95] introduced the straight skeleton defined as the union of the pieces of angular bisectors traced out by polygon vertices during a continuous shrinking process in which edges of the polygon move inward, parallel to themselves at a constant speed. The straight skeleton is applied to constructing general shaped roofs based on any simple building polygon, regardless of their being rectilinear or not.

As shrinking process shown in Fig.2, each vertex of the polygon moves along the angular bisector of its incident edges. This situation continues until the boundary change topologically. According to Aichholzer et al. [Aic95], there are two possible types of changes:

(1) ***Edge event***: *An edge shrinks to zero, making its neighboring edges adjacent now*.

(2) ***Split event***: *An edge is split, i.e., a reflex vertex runs into this edge, thus splitting the whole polygon. New adjacencies occur between the split edge and each of the two edges incident to the reflex vertex.*

The shrinking procedure is uniquely determined by the distance $\mathbf{d}_{shri}$ between the two edges of before & after shrinking procedure.

The distance e_$\mathbf{d}_{shri}$ is the $\mathbf{d}_{shri}$ when an *Edge* event happens in the shrinking process. e_$\mathbf{d}_{shri}$ for the edge (ed_i) is calculated as follows:

$$e\_\mathbf{d}_{shri} = {L_i}/{(\cot(0.5 * \theta_i) + \cot(0.5 * \theta_{i+1}))} \quad (1)$$

where L_i is the length of ed_i, and θ_i & θ_{i+1} are internal angles of vertices incident to ed_i.

When $0.5^* \theta_i + 0.5^* \theta_{i+1} < 180$ degrees, i.e. , the sum of the internal angles of two vertices incident to an edge is less than 360 degrees, an *Edge* event may happen unless the edge is intersected by an angular bisector from a reflex vertex and a *Split* event happens.

4.1 How Straight Skeleton is formed

How a straight skeleton and monotone polygons are formed is as follows.

(1) One simple polygon (**P**) is given such as shown in Fig.2a. If there is any reflex vertex in the **P**, then it can be divided into two or more polygons.

(2) The system calculates e_$\mathbf{d}_{shri}$ (receding distance for an *Edge* event, shown in above (1)) for all edges and finds the shortest of them. Then, the system checks if a *Split* event occurs by increasing $\mathbf{d}_{shri}$ by (e_$\mathbf{d}_{shri}$ /n_step). In this way, the shrinking process may proceed until $\mathbf{d}_{shri}$ reaches the shortest e_$\mathbf{d}_{shri}$ calculated.

(3) During shrinking until $\mathbf{d}_{shri}$ reaches the shortest e_$\mathbf{d}_{shri}$, the system checks if a 'checking angular bisector' from a reflex vertex intersects another edge of the polygon or not. If an edge is found intersected, then the system calculates the node position by the *Split* event. The position of the node is calculated by the intersection of two angular bisectors: one from the reflex vertex and the other between the intersected edge and one of two edges incident to the reflex vertex. However, edges may be intersected by several 'checking angular bisectors' from several reflex vertices. Among the several reflex vertices, the reflex vertex that gives the shortest $\mathbf{d}_{shri}$ will be selected for calculating the node position.

(4) In the process of (2), a *Split* event may happen and the polygon will be divided into some polygons: **P**s.

In this '*Split* event checking' process, all divided polygons are checked if they can be divided more. As long as there are some **P**s that can be divided, '*Split* event checking' routine will continue. After that, the system concentrates on the *Edge* event procedure.

(5) In this stage, since the number of polygons divided does not increase by the *Split* event, the system can concentrate on the *Edge* event including *Third* and *Line segment* event procedures. If the polygon divided has only three vertices, then the polygon (triangle) collapses to a node; this is the final stage for the polygon divided.

(6) While the *Edge* events are being executed, the topology of the polygon will change. If the change happens, then the system re-implement the process from (2) to (5) for the polygon whose topology has changed. At that moment, the system recalculates the length of each edge and the internal angle of each vertex in order to find the shortest $\mathbf{d}_{shri}$ for next events. This re-implementation process continues until all polygons changed collapse to a node or a line segment.

4.2 Node Structure

The generated node will be associated with the edges of original **P** (original edge: o-edge) which are identified by original edges' ID (e.g. *edg1* & *edg2* in Fig.2a), since at least three original edges sweep to form a node. Therefore, at each event when the node is generated, at least three o-edges will be linked to the

 ISBN 978-80-86943-37-4

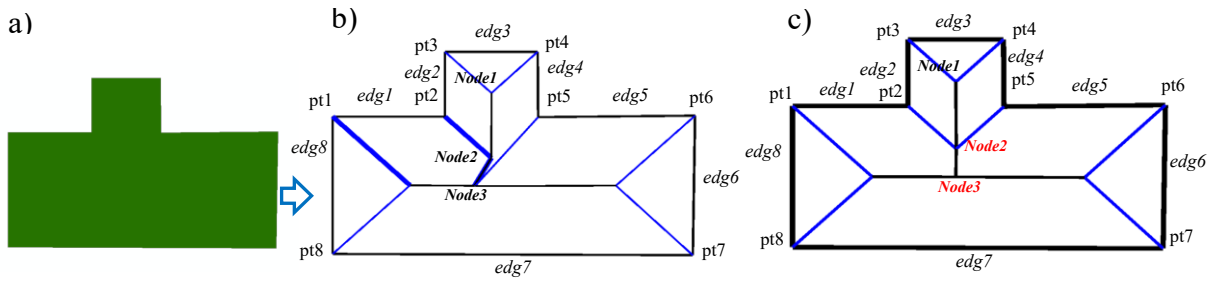


Figure 3: a) An orthogonal building polygon b) a monotone polygon with an acute angle c) rectified monotone polygons by '*Line segment*' event

node. This means more than three o-edges ID will be stored in the node with a suitable structure.

In our system, a node has the following properties; **(a)** '*Node Type*' (how the node is risen; by *Edge* event or *Split* event, *Vertex* event, *Multiple Edge* event and so on) **(b)** 'Number of forming edges' (usually three edges sweep to form a node, but more than three edges sweep in case of *Multiple Edge* event) **(c)** 'o-edge ID preceding the vanishing edge' (by *Edge* event) or 'o-edge ID of one of the edge incident to the reflex vertex' (by *Split* event) **(d)** 'o-edge ID following the vanishing edge' (by *Edge* event) or 'o-edge ID of the other edge incident to the reflex vertex' (by *Split* event) **(e)** 'o-edge ID of at least one vanishing edge' (by *Edge* event) or 'o-edge ID of a split edge' (by *Split* event)

Since three edges usually sweep into the node, three 'o-edge IDs' are stored in the property of a node. These IDs are used for forming a monotone polygon. The system is looking for the node which has the same 'o-edge ID' as each original edge of **P** to form monotone polygons.

In special cases, four or more edges collapse into nodes, such as *Node2* in Fig.2 and *Node1,2,3* in Fig.4c. In extreme cases, such as a hexagon or a regular polygon, a star-shaped polygon collapses to a node, four or more o-edges will sweep into a node, and more than three 'o-edge IDs' are stored in the property of the node. Therefore, a node needs 'Number of forming edges' property.

This is the case of a multiple *Edge* event or the case Eppstein et al. [Epp99] defined as a 'degenerate case' in which the straight skeleton can have vertices of degree higher than three, introduced by simultaneous events at the same location. However, in single or double precision floating point calculation for the position of the node, it is quite rare for four or more vertices to reach the same point simultaneously.

To rectify monotone polygons to be appropriate shape for roof boards, in our system, if multiple edges collapse into a certain area considered as a point for a node, then they are considered to converge into the same point and the node is formed.

4.3 Line Segment Event

Since three edges usually sweep into a node, very often this causes a quite acute angle shape that is not appropriate for roof board shape shown in Fig.3. In Fig.3c, pt5 propagates to join pt2 and four edges (*edg1,2,4,5*) propagate into *Node2*, whereas, in Fig.3b, pt5 does not join pt2 and goes off *Node2*, and three edges (*edg1,4,7*) result in *Node3* with acute angle shape. This acute angle shape is also found at the

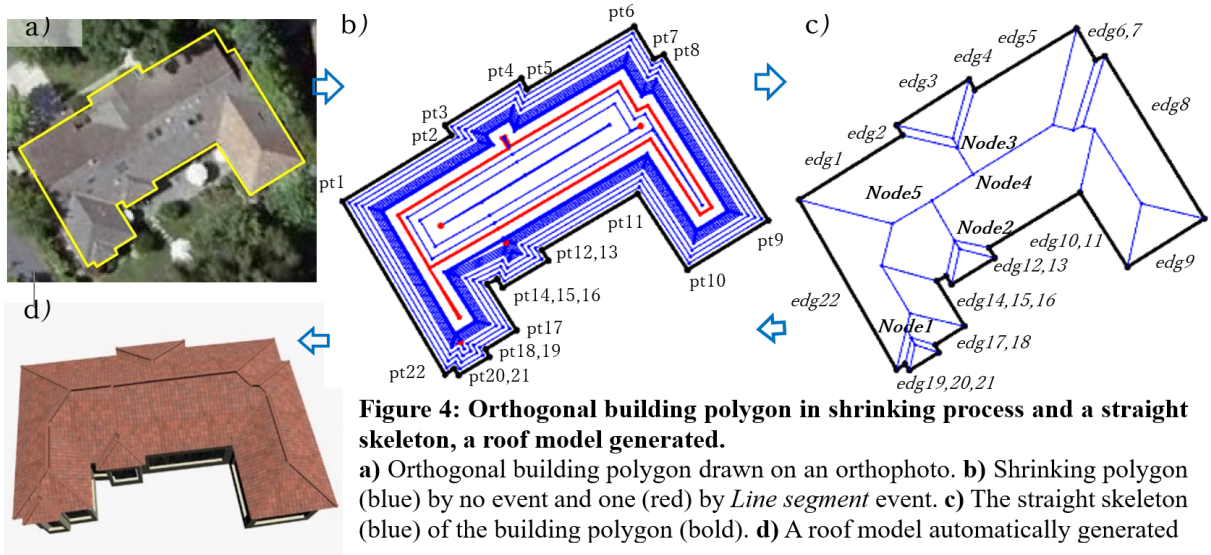


Figure 4: Orthogonal building polygon in shrinking process and a straight skeleton, a roof model generated.
a) Orthogonal building polygon drawn on an orthophoto. **b)** Shrinking polygon (blue) by no event and one (red) by *Line segment* event. **c)** The straight skeleton (blue) of the building polygon (bold). **d)** A roof model automatically generated

ISBN 978-80-86943-37-4

figure of Eppstein et al. [Epp99], which uses perturbation techniques, replacing the high-degree node with several nodes of degree three, connected by zero-length edge. In our system, using the technique completely opposite to Eppstein's perturbation, a '*Line segment*' event is proposed where edges are overlapped and collapse into a line segment instead of a node to avoid the acute angle shape. This so-called snapping function is done by setting up a certain range for possible '*Line segment*' events, in which edges converge into a certain area considered as a line segment, then they are supposed to converge into the same line segment.

By a '*Line segment*' event, two parallel edges converge into one edge (line segment), and the convergent line segment will be detached from a next shrinking body polygon. But if the detached line segment leaves no vertex for next shrinking process, then the line segment is disconnected from a body skeleton. Therefore, the detached line segment leaves at least one vertex for next shrinking process. Examples are shown in the line segment between *Node2* and *Node5* in Fig.4c and Fig.5b; one node whose interior angle is flat will remain for the next shrinking process so as to create the border of monotone polygons. For example, in Fig.4c & Fig.5b, four edges (*edg11,12,14,15*) propagate into *Node2*, and two overlapping edges (*edg12,14*) turn into the line segment incident to *Node2* & nearby *Node* after edg13 disappeared.

If a configurable range is quite narrow, then *edge* propagation will be extended, ending in *Node5* as shown in Fig.5a; three edges (*edg12,14,15*) result in *Node2*, and three edges (*edg11,12,15*) result in *Node5* whose inner angle is quite acute, which is improper for roof board shape.

4.4 Monotone Polygon Nodes Sorting

According to Aichholzer et al. [Aic95], the area divided by a straight skeleton will be a 'monotone polygon'. To get the monotone polygons, the set of the nodes belonging to each original edge will be sorted according to the 'coordinate value of node vector projections' onto the original edge vector parallel to each original edge. These nodes are coplanar and will form roof boards for a 3D building model. However, for some polygon, this methodology does not work, resulting in self-intersecting polygons. Fig.5c shows monotone polygons for *edg13* are self-intersecting. This is because the edge (connecting *Node3 & Node4*) of the monotone polygon is perpendicular to the original edge (*edg13*) of the polygon, and the nodes are connected in the order of 'node vector projection'. The self-intersection is found at *edg29* in Fig.5c and *edg21* in Fig.5b.

To avoid self-intersection, the azimuth angle of the nodes belonging to the same monotone polygon is proposed, where the azimuth is the angle between each original edge vector and a node vector. The first node in the monotone polygon vertices numbering is selected from the node with least azimuth, and the last node is the node with greatest azimuth, since the nodes near the both ends on an original edge may wrap around both ends for some monotone polygons, and wrapping around nodes may not have simply increasing 'coordinate value'. For example, in Fig.5c, the edge (connecting *Node1 & Node2*) of the monotone polygon is perpendicular to the original edge (*edg29*), and *Node1 & Node2* have the same 'coordinate value', resulting in self-intersection at nodes sorting. Thus, the nodes at ends are sorted by the azimuth angles. Then, the sorting of the nodes is found successful in a complicated shape polygon such as the one in Fig.4c and Fig.6c.

5 APPLICATION

Here are the examples of 3D building models automatically generated by the integrated system. Fig.6 & Fig.7 show the examples of 3D building models automatically generated. In generating these models, we classify the case of '*Line segment* event'. In *Split* event as mentioned in section 4.1, it is assumed to calculate the intersection of two non-parallel line

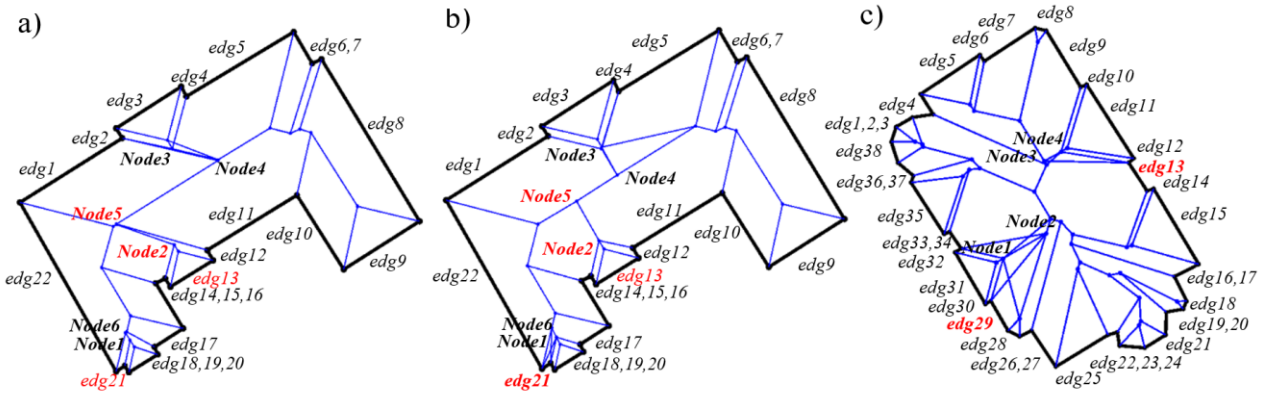


Figure 5: Monotone polygons with acute angle and self-intersecting monotone polygons. a) Some monotone polygons have acute angle if a *Line segment* event does not occur. **b)** & **c)** Some monotone polygons are self-intersecting if the set of the nodes are sorted according to the 'node vector projections' onto the original edge vector.

ISBN 978-80-86943-37-4

segments, i.e., two non-parallel angular bisectors. However, for some orthogonal polygons, two parallel edges will be overlapped when shrunk by e_$\mathbf{d}_{shri}$, and two parallel angular bisectors will be overlapped. If we do not classify the case of *Line segment* event, then we end up with numerical error by trying to calculate the intersection of two parallel line segments.

Once 3D models with roofs are created, a top view of these models can be a roof report as shown in Fig.7i & 7j, which can be used for the rapid assessment of roof damages by insurance companies. Automated generation of simple and complex roof geometries will be utilized for rapid roof area damage reporting by the length measurements and area calculations of all roof surfaces. The roof board area will be easily calculated, since a roof board is a monotone polygon, and can be partitioned into a set of trapezoids or triangles. The roof board area will be calculated by adding these trapezoids, and subtracted or added by two triangles, depending on the shape of the monotone polygon.

The advantage of our generation system is that our 3D building models created are utilized for architectural design, i.e., BIM (Building Information Model), while 3D models created by procedural modeling are not solid models but surface models which are to be converted into geometric primitives (CSG) when they are used for construction design.

Now, architectural design world is experiencing a shift from 2D CAD drawings to BIM 3D modeling. BIM revolution is happening in the construction industry that is producing a step change in efficiency and accuracy. In our research, 3D building models automatically created can be used for BIM 3D modeling. There is no automatic generation system for 3D building solid models with complicated roofs as far as we know. Automatic generation will be compared with manual creation which are a series of manual operations mentioned in section 1 by 3ds Max, and broken down into functions of the program (Maxscript) described as CG module's process in section 3. It will take about more than one hour to create one hipped roof house including making intricately shaped ridges, while several seconds to automatically generate one by the personal computer. If given digital maps with attributes being inputted, as shown in Fig.1&7, the system automatically generates one hundred 3D building models within less than 10 minutes by the personal computer with Intel(R) Core(TM) i7-7820HK CPU 2.90GHz.

6 CONCLUSION

In this paper, the new and extended methodology is proposed for adding '*Line segment*' event besides the conventional events, and 'monotone polygon nodes sorting' by which self-intersecting monotone polygons are not formed. Thus, the proposed integrated system succeeds in automatically generating 3D building models.

The roofs created by the straight skeleton are limited to hipped roofs with their roof ridges parallel to nearby long edges of the building contour. However, there are many roofs whose ridges are perpendicular to long edges. In the residential area all over the world, there are many roofs the straight skeleton method cannot create. For example, in the middle of the top edge in Fig.4a and Fig.6a, Fig.7e, there are some branch roofs which are not slanting and drop 90 degrees vertically, i.e., gable roofs. These are not created by the straight skeleton.

In order to create various shape of roof, we propose a couple of schemes to create roofs by straight skeleton computation or partitioning or separating of orthogonal polygons. A complicated orthogonal building polygon can be partitioned or separated into a set of rectangles. Our proposed system partitions orthogonal polygons into a set of rectangles and places various shapes of roofs which include gable & hipped roofs or the roof whose ridges are perpendicular to nearby long edges on these partitioned rectangles. Thus, in order to create the 3D building models that have hipped and gable branch roofs, the future work is for developing the system in which we can select a couple of various schemes; dividing or separation

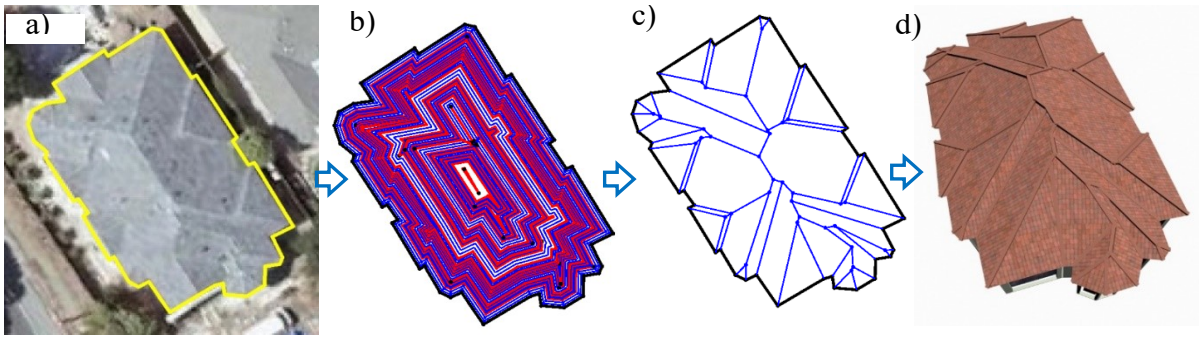


Figure 6: ***Complicated shape polygon in shrinking process and a straight skeleton, a roof model generated.***

a) Complicated building polygon drawn on an orthophoto. b) Shrinking polygons (blue) by no event and red ones by events. c) The straight skeleton (blue) of the building polygon (bold). d) A roof model automatically generated

ISBN 978-80-86943-37-4

scheme or straight skeleton scheme to automatically create the different styles of roofs for one building footprint.

6 REFERENCES

[Aic95] Aichholzer O., Aurenhammer F., Alberts D., Gärtner B. A novel type of skeleton for polygons. Journal of Universal Computer Science (1995), 1 (12): 752–761.

[Ede14] Johannes Edelsbrunner, Ulrich Krispel, Sven Havemann, Alexei Sourin. Constructive Roof Geometry. Cyberworlds (CW) 2014,·DOI: 10.1109/CW.2014.17

[Epp99] Eppstein David, Erickson Jeff. Raising roofs, crashing cycles, and playing pool: applications of a data structure for finding pairwise interactions. Discrete and Computational Geometry (1999), 22 (4): 569–592

[Hel17] Helda M., Palfradera P. Straight Skeletons with Additive and Multiplicative Weights and Their Application to the Algorithmic Generation of Roofs and Terrains. Computer-Aided Design, Elsevier B.V. (Nov 2017), Vol 92, pp. 33-41

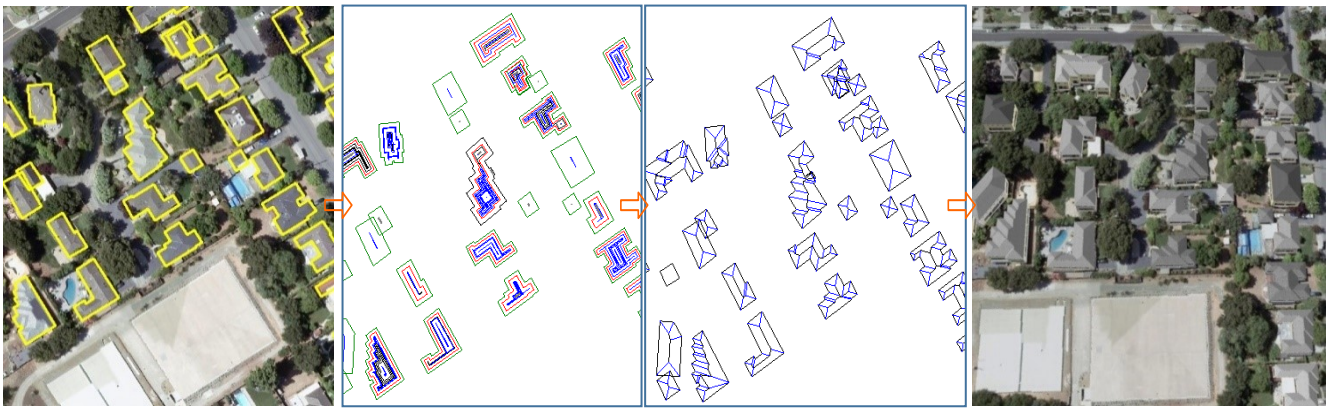
[Jia09] Jiang N., Tan P., Cheong L. F. Symmetric architecture modeling with a single image. Proc. SIGGRAPH Asia '09 (2009), papers No. 113

[Kel11] Kelly T., Wonka P. Interactive Architectural Modeling with Procedural Extrusions. ACM Transactions on Graphics (TOG) (2011), 30(2), 14-28.

[Mue06] Mueller P., Wonka P., Haegler S., Ulmer A., Gool L.V. Procedural modeling of buildings. ACM Transactions on Graphics (2006), 25, 3, 614–623.

[Par01] Parish Y. I. H., Müller P. Procedural modeling of cities. Proceedings of ACM SIGGRAPH (2001), ACM Press, E. Fiume, Ed., New York, 301–308

 ISBN 978-80-86943-37-4

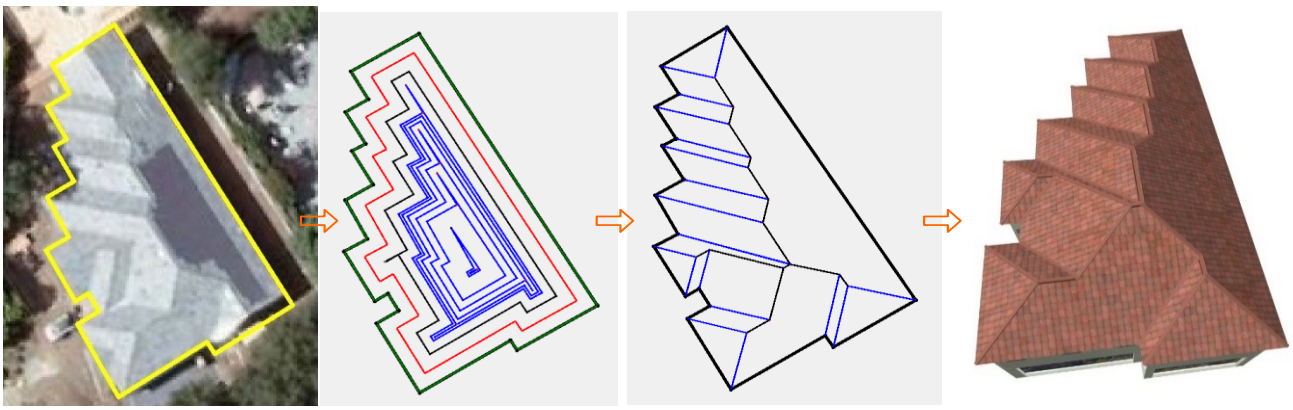


(a) Building polygons on 2D Digital Map: Most of them are orthogonal.

(b) Set of receding polygons by $\mathbf{d}_{shri}$ calculated by equation (1)

(c) The straight skeleton formed as the union of the pieces of angular bisectors

(d) Automatically generated 3D building model based on monotone polygons

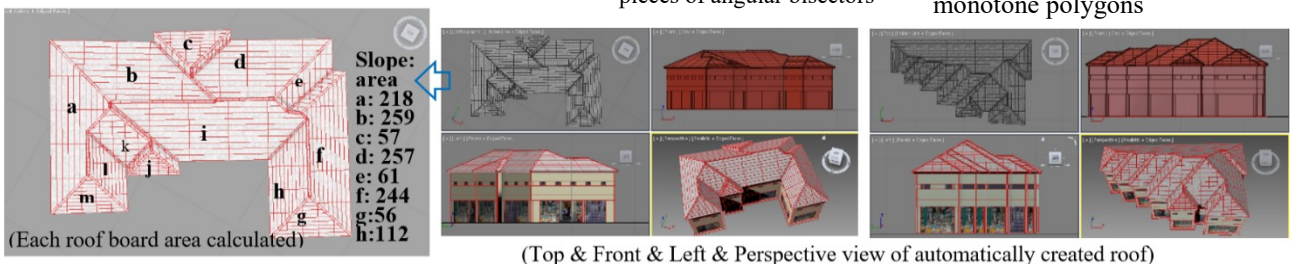


(e) Enlarged building polygons: orthogonal

(f) Set of receding polygons by $\mathbf{d}_{shri}$ calculated by equation (1)

(g) The straight skeleton formed as the union of the pieces of angular bisectors

(h) Automatically generated 3D building model based on monotone polygons



(i) Automatically created roof report for damage evaluation

(Top & Front & Left & Perspective view of automatically created roof)

(j) Ground [floor] plan (Top view) of automatically created 3D building model

Figure 7: Shrinking process and a straight skeleton, a roof model generated, roof report

ISBN 978-80-86943-37-4

 ISBN 978-80-86943-37-4

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

Π-surfaces: products of implicit surfaces towards constructive composition of 3D objects

Adriano N. Raposo and Abel J.P. Gomes
Instituto de Telecomunicações and Universidade da Beira Interior
R. Marquês de Ávila e Bolama
6200-001, Covilhã, Portugal
anraposo@ubi.pt, agomes@di.ubi.pt

ABSTRACT

Implicit functions provide a fundamental basis to model 3D objects, no matter they are rigid or deformable, in computer graphics and geometric modeling. This paper introduces a new constructive scheme of implicitly-defined 3D objects based on products of implicit functions. This scheme is in contrast with popular approaches like *blobbies*, *meta balls* and *soft objects*, which rely on the sum of specific implicit functions to fit a 3D object to a set of spheres.

Keywords

Implicit surfaces, surface modeling, constructive modeling, 3D modeling, computational geometry, geometric modeling.

1 INTRODUCTION

Both implicit surfaces and parametric surfaces have been widely used in computer-aided design, geometric modeling, visualization, animation, and computer graphics [GVJ+09]. Implicit surfaces benefit from the valuable properties in modeling, namely: closure and point membership. Indeed, applying boolean operations (intersection, union, and difference) results in another well-defined implicit surface [GRM99, BGA04]. Moreover, it is easy to check whether a point behind, on, or beyond an algebraic implicit surface, as needed in collision detection.

However, unlike piecewise parametric surfaces, implicit surfaces are not akin to local shape changes interactively. Also, rendering an implicitly-defined algebraic surface poses some difficulties because it requires its preliminary triangulation [RV85, PK89, Gom03], particularly for implicit surfaces defined by high degree polynomial functions; hence, the low degree algebraic surface patches or algebraic splines used for geometric modeling [Baj88, BX97, LPP06, CCD00, War89, WZ00].

In this paper, we introduce a new method to model complex shapes through products of implicitly-defined algebraic surfaces (e.g., spheres, ellipsoids, cylinders, hyperboloids, and tori), called Π-*surfaces*. These Π-surfaces allow us: (1) to represent a surface as the product of two or more patches; (2) to globally edit the shape of each patch (e.g. patch replacement, patch removal, patch insertion); (3) to locally edit the shape of each patch (e.g. deformations like recesses, saliences, and so forth); (4) to blend two patches using a single blending parameter; and (5) to better control bulging effects inherent to sum-based surfaces (e.g., ∑-surfaces like Gaussian surfaces) via the blending parameter.

In short, we propose a new constructive method for 3D objects through products of implicit functions, which differs from the dominant method found in the literature, which is based on ∑-surfaces (e.g., *blobby molecules* [Bli82], *metaballs* [NHK+85], *soft objects* [WMW86], *blobby model* [Mur91], piecewise implicit surface patches [Baj88, BX97], and homotopy-generated surfaces [Bed92] [HH85].

2 Π-SURFACES

Π-*surfaces* can be used to approximate any given smooth and bounded object in $\mathbb{R}^3$ whose surface is defined by a single polynomial as a product of subsidiary polynomials. In other words, we can design any smooth object with a single algebraic surface. Let us denote the defining polynomials as $f_i \in \mathbb{R}[x_1,\ldots,x_n](i=1,\ldots,k)$. Then, the approximating object is defined by the polynomial

$$F(x,y,z) = \prod_i f_i(x,y,z) - r \qquad (1)$$

where $r \in \mathbb{R}$ stands for the blending parameter that controls the approximating error.

From Eq. (1), which is the core equation of this paper, we can find that the shape of the approximating surface depends on the primitive surfaces f_i and the parameter r. That is, we have a product (Π) of functions f_i, where each function represents an arbitrary geometric primitive i, and r is the approximation parameter for the entire surface.

2.1 Implicit Primitives

Π-*surfaces* are built up from arbitrary implicit surface primitives. However, unlike traditional distance-based methods, Π-surface primitives are not limited to sets of spheres (or ellipsoids) to compose 3D objects. Instead, Π-surfaces use primitives like planes, quadrics, tori, and other more complicated surfaces. These primitives split into two different groups: *bounded primitives* (e.g., sphere, ellipsoid, torus, and so forth) and *unbounded primitives* (e.g., plane, cylinder, cone, hyperboloid of 1 sheet, hyperboloid of 2 sheets, paraboloid, hyperbolic paraboloid, and the like). Both bounded and unbounded primitives may be combined into the same 3D object.

2.2 Shape Operations

There are three different types of shape operations: *shape repositioning*, *global shrinking or inflation*, and *local bulges or concavities*.

2.2.1 Shape Repositioning

The shape of a Π-surface can also be adjusted by controlling the position of the primitives. This is somehow intuitive, though the blending of the primitives only depends on the distance between them and the blending parameter r. For example, both spherical primitives in Figure 1 are combined into a Π-surface (in transparent gray). Obviously, considering the blending parameter $r = 0.25$ remains unchanged, the resulting Π-surface depends on the distance between both primitives.

2.2.2 Global Shrinking vs. Global Inflation

Varying the blending parameter r in Eq. (1) makes the entire Π-surface shrink or inflate. Specifically, decreasing the value of r to zero makes the Π-surface shrink to the surface of the union of implicit surface primitives. Conversely, increasing the value of r inflates the Π-surface globally. Figure 2 shows how distinct values of r affect the global shape of a Π-surface.

Similar to $\sum$-surfaces, undesired bulges about the intersection regions of primitives may occur in Π-surfaces. The difference is that Π-surfaces allow the control on bulges by reducing the value of the approximation parameter r, as illustrated in Figure 3.

2.2.3 Local Deformations

Unlike $\sum$-surfaces, Π-surfaces are very efficient in performing local deformations without affecting the global shape of the objects. Such deformations are performed by the action of primitives on other primitives. There are two main types of local deformations: *local protrusions* and *local depressions*.

Local protrusions are generated by adding a relatively small primitive to the Π-surface, but the small primitive needs to be close enough to the Π-surface in order to affect it. This mechanism is illustrated by the red object in Figure 4(a), where a small sphere imposes a spherical bump to the plane $x = 0$.

Local depressions are a little bit trickier to obtain with Π-surfaces. In this case, we need to invert the sign of the implicit function of the small primitive, yet inverting the original implicit function of the global shape is also feasible. This local deformation mechanism is illustrated in Figure 4(b)-(c), where we also used a small sphere to obtain a concavity in the plane $x = 0$.

2.3 Interactive Shape Control

As seen above, adding a primitive is as simple as including its defining implicit function in the product of Π-surface. This procedure only changes the already existing object locally. Besides, the position and geometric parameters of the primitive can also be interactively adjusted on the fly by the user. There two types of shape composition control: additive and subtractive.

Additive shape composition means we interactively add positive primitives to a given object without losing the control of its overall shape, i.e., without losing the capability of anticipating how the inclusion of a specific primitive will affect the global shape the object, as illustrated in Figure 5. In this case, the user creates the handle of the cup by interactively adding a torus to an existing bowl composed by a flattened ellipsoid and another torus. As depicted in Figure 5, when the user interactively displaces the handle until its final position, the handle remains perfectly blended with the cup. If needed, the user might also interactively change the "thickness" of the handle just by changing the smaller radius of the torus.

Let us now see how three more objects are obtained from the composition of positive primitives.

Example 1. The hammer shown in Figure 6(a) was built using two implicit primitives so that $F = f_1 \,.\, f_2 - c$, where $c = 1$ is the blending parameter, and f_1 defines the first primitive (a block) and f_2 the second primitive (an ellipsoidal handle) as follows:

$$f_1 = (x-5)^6 + \left(\frac{y}{2}\right)^6 + z^6 - 100; f_2 = \left(\frac{x}{8}\right)^2 + y^2 + z^2 - 1.$$

Example 2. The sword pictured in Figure 6(b) is defined by $F = f_1 \,.\, f_2 \,.\, f_3 - 5$, where f_1, f_2, and f_3 define three

 ISBN 978-80-86943-37-4

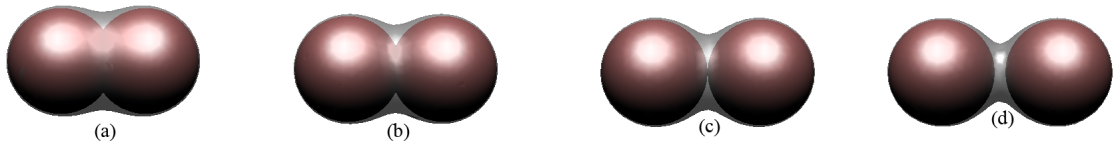


Figure 1: Repositioning of two spheres of the surface $F(x,y,z) = ((x-d)^2+y^2+z^2-1)\cdot((x+d)^2+y^2+z^2-1)-0.25$: (a) $d = 0.8$; (b) $d = 0.9$; (c) $d = 1.0$ and (d) $d = 1.1$. The positions of the surface primitives were adjusted, but the parameter r remained unchanged. The Π-surface appears in translucent gray.

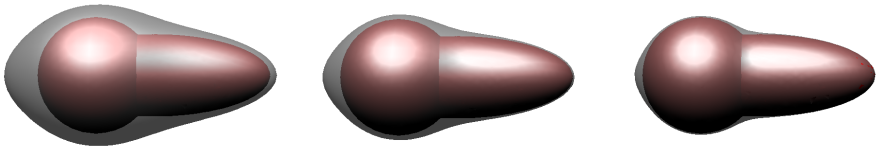


Figure 2: "Eggplant" surface defined by the function $F(x,y,z) = (\frac{1}{64}x^2+\frac{1}{8}y^2+\frac{1}{8}z^2-0.25)\cdot((x+2)^2+y^2+z^2-4)-r = 0$ for different values of r: (a) 1.0; (b) 0.25; and (c) 0.05. The Π-surface appears in translucent gray.

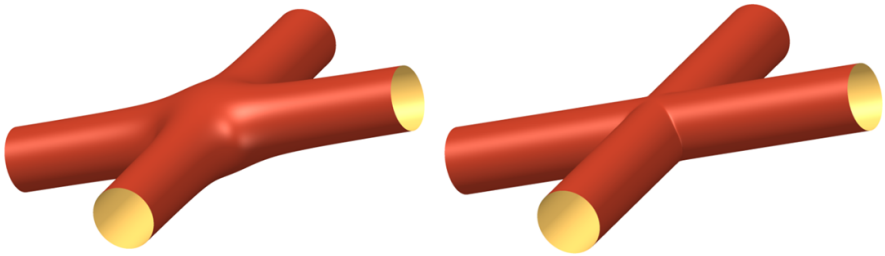


Figure 3: The parameter r is used to reduce an undesired bulge that occurs about the intersection of two crossed cylinders defined by the function $F(x,y,z) = (x^2+z^2-1)\cdot(y^2+z^2-1)-r = 0$: (a) $r = 1$; (b) $r = 0.001$

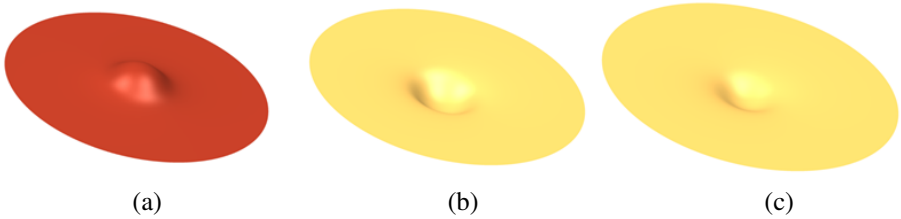


Figure 4: Local deformations obtained by placing a small sphere near the $x = 0$ plane : (a) protrusion defined by a sphere in $(f(x,y,z) = (x^2+y^2+z^2-1)\cdot(x)-1 = 0$; (b) depression obtained by inverting the sign of the original plane as in $f(x,y,z) = (x^2+y^2+z^2-1)\cdot(-x)-1 = 0$; and (c) concavity obtained by inverting the sign of the sphere as in $f(x,y,z) = (-x^2-y^2-z^2-0.01)\cdot(x)-1 = 0$ (right)

implicit primitives, two crossed ellipsoids and a sphere, as follows:

$$f_1 = (x+17)^2+y^2+z^2-1;$$

$$f_2 = (2x+24)^2+\left(\frac{y}{6}\right)^2+z^2-1;$$

$$f_3 = \left(\frac{x}{16}\right)^2+\left(\frac{y}{2}\right)^2+(2z)^2-1.$$

Example 3. The table shown in Figure 6(c) is defined by $F = f_1\cdot f_2\cdot f_3\cdot f_4\cdot f_5-1$, that is, by five implicit primitives, a rounded parallelepiped for the table top and four ellipsoids for the table legs, where

$$f_1 = 0.1\cdot x^6+0.1\cdot y^6+(8z+5)^6-1;$$

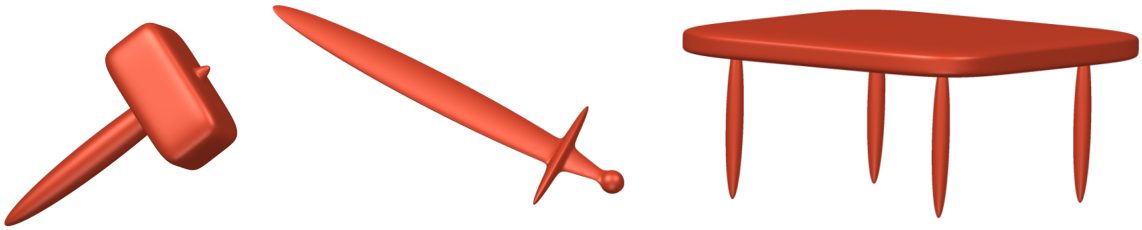
$$f_{2,3,4,5} = \left(\frac{x\pm1}{0.1}\right)^2+\left(\frac{y\pm1}{0.1}\right)^2+\left(\frac{z}{0.9}\right)^2-0.5.$$

Subtractive shape composition means we interactively add negative primitives to an existing Π-surface, as shown in Figure 7. Interestingly, the negative primitive used to create the concavity shrinks as it gets close

ISBN 978-80-86943-37-4

(a) (b) (c)

Figure 5: Interactive edition of a moving handle (or torus) to create a cup as given by the function $F(x,y,z) = (((x-2.2\cdot a)^2+(z+1)^2+y^2+0.5-0.01)^2-2\cdot((x-2.2\cdot a)^2+(z+1)^2))\cdot(\frac{1}{8}\cdot x^2+\frac{1}{8}\cdot y^2+(z+2)^2-0.1)\cdot(((z+1)^2+10\cdot x^2+10\cdot y^2+19)^2-800\cdot(x^2+y^2))-10$.



(a) (b) (c)

Figure 6: Additive shape composition of Π-surface objects: (a) hammer; (b) sword; and (c) table.

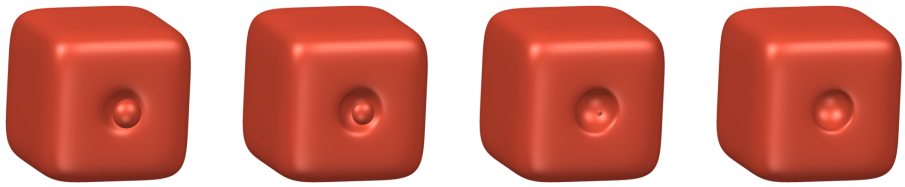


Figure 7: Interactive edition of a depression on a rounded cube surface by moving a small negative sphere. The resulting object is defined by the following Π-surface: $F(x,y,z) = (x^6+y^6+z^6-1)\cdot(-(x+a)^2-(y+b)^2-(z+c)^2+0.05)-0.01$, where (a,b,c) is the position of the sphere.

to the existing object, and even vanishes at some point. This is a very useful feature because it eliminates undesired components in the final object; specifically, the negative sphere in Figure 7 is used to interactively create a concavity in one of the faces of a 3D cube. Notice how the sphere becomes smaller and smaller to a point at which it finally vanishes in the proximity of the cube.

2.4 Geometry Evaluation

The geometry evaluation (also known as boundary evaluation [RV85]) builds upon ray casting to solve two difficult problems in implicit surface modeling:

- *Rendering*. We can immediately visualize the surface on screen with a significant realism and geometry fidelity, including singularities like apices, intersection curves, and the like, though without explicitly solving such singularities (i.e. the loci where the first derivatives vanish), as well as multi-component surfaces (see, for example, [RG06]).
- *Triangulation*. The point sampling inherent to ray casting allows us to reconstruct the surface because we classify the surface points as either regular or singular using the gradient (i.e. first derivatives). Essentially, this problem consists in triangulating a point cloud whose points are samples of the surface (see, for example, [BMR+99, LHRS05, KBH06]). Note that the nearby points are associated with adjacent pixels on the screen so that the triangulation is more amenable to carry out than in standard surface reconstruction algorithms lacking information like normals or partial derivatives. Moreover, knowing the partial derivatives at such points, we can quickly identify singularities like apices, sharp features, self-intersections, and so forth. Summing up, we use a novel image-based surface reconstruction, though we also may use algorithms like those described by Skala et al. in [ČS04, ČS05, ČS07]. Obviously, after triangulating a surface, we no longer need ray casting for the purpose of the graphics output.

ISBN 978-80-86943-37-4

Thus, our renderer combines ray casting and triangulation approaches into a single renderer.

2.5 Molecular Π-surfaces

Molecular modelling is an important application for Π-surfaces. Knowing that each atom has a corresponding *van der Waals* (VDW) radius [Bon64][Bat01], a molecule can be seen as a set of overlapping spheres representing the atoms.

2.5.1 *Molecular Π-surface Formulation*

The molecular surface S of a molecule can be then described by the following Π-surface:

$$S = \prod_{i=1}^{n} f_i - r \tag{2}$$

where

$$f_i = (x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2 - {R_i}^2 \tag{3}$$

represents the spherical surface of the i-th atom of radius R_i and center (x_i, y_i, z_i). This formulation is in contrast with the Gaussian Σ-surface proposed by Blinn [Bli82] to represent molecular surfaces.

It is worthy noting that for $r > 0$, Equation 2 represents a smooth molecular surface, so there is no need to use ray casting to sample the surface. Instead, we can directly triangulate the surface on-the-fly using triangulation algorithms as those described in [RG06, RQG09, DG11, DG15, DNJG17].

2.5.2 *Examples of Molecular Π-surfaces*

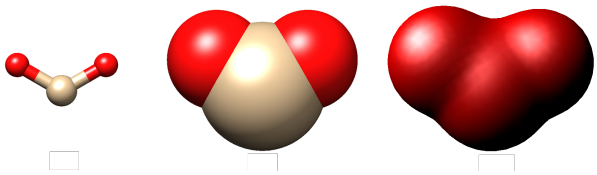
Let us now see how to model the molecular structures of two real drugs using Π-surfaces: (a) *formic acid* and (b) *acetic acid*.

Formic acid. This drug occurs naturally and is present in the venom of bees and ant stings. Using the atomic structure obtained from the DrugBank and the VDW radii from [Bat01] (1.52 Å for oxygen and 1.70 Å for carbon), the spheres have the following center points and radii:

$$c_1 = (2.310, -1.334, 0.000), R_1 = 1.52;$$
$$c_2 = (3.644, -2.104, 0.000), R_2 = 1.70;$$
$$c_3 = (4.977, -1.334, 0.000), R_3 = 1.52.$$

Then, using Equation 2 (with $r = 1$) we obtain the Π-surface that represents the *formic acid* molecule as follows:

$$\begin{aligned} S = ((x-2.310)^2 + (y+1.334)^2 + z^2 - 1.52^2)\cdot \\ ((x-3.644)^2 + (y+2.104)^2 + z^2 - 1.70^2)\cdot \\ ((x-4.977)^2 + (y+1.334)^2 + z^2 - 1.52^2) - 1 \end{aligned}$$



(a) (b) (c)

Figure 8: Formic acid: (a) ball-and-stick; (b) atom-describing spheres (VDW representation); and (c) molecular Π-surface.

The formic acid Π-surface is shown in Figure 8 (c).

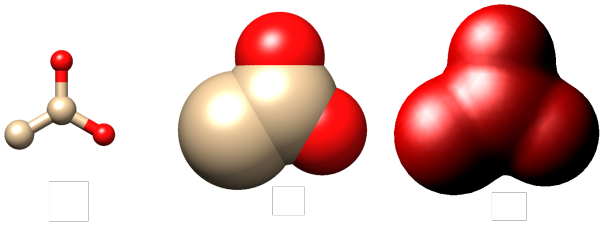
Acetic acid. This drug results from the oxidation of ethanol and destructive distillation of wood. As seen in the previous example, the center points and radii of its atoms are as follows:

$$c_1 = (24.214, -24.150, 0.000), R_1 = 1.70;$$
$$c_2 = (25.548, -23.380, 0.000), R_2 = 1.70;$$
$$c_3 = (26.881, -24.150, 0.000), R_3 = 1.52;$$
$$c_4 = (25.548, -21.840, 0.000), R_4 = 1.52$$

Then, using Equation 2 (with $r = 1$), we obtain the algebraic surface that represents the *acetic acid* molecule as follows:

$$\begin{aligned} S = ((x-24.214)^2 + (y+24.150)^2 + z^2 - 1.70^2)\cdot \\ ((x-25.548)^2 + (y+23.380)^2 + z^2 - 1.70^2)\cdot \\ ((x-26.881)^2 + (y+24.150)^2 + z^2 - 1.52^2)\cdot \\ ((x-25.548)^2 + (y+21.840)^2 + z^2 - 1.52^2) - 1 \end{aligned}$$

The *acetic acid* Π-surface is depicted in Figure 9 (c).



(a) (b) (c)

Figure 9: Acetic acid: (a) ball-and-stick; (b) atom-describing spheres (VDW representation); and (c) molecular Π-surface.

The *Molecular Π-surface* formulation can also be used, for the sake of simplification, to model the four nucleotides (adenine, cytosine, guanine and thymine) used as DNA building blocks in [RG12, RG14, RG15], replacing the Gaussian Σ-surface model.

ISBN 978-80-86943-37-4

3 CONCLUSIONS AND FUTURE WORK

This paper presents a new general method to model complex 3D objects as the product of algebraic functions representing simpler implicit surface primitives. The resulting surfaces are here called Π-surfaces. Unlike traditional methods used in implicit modeling, this new method is not based on a sum of specific distance functions. Instead, the 3D objects are built upon a product of algebraic functions representing arbitrary implicit surfaces such as spheres, ellipsoids, and tori. This new approach allows us to interactively perform controlled deformations on the object shape both locally and globally. Two possible applications are proposed for this new model: *constructive building of 3D objects* and *molecular surfaces modeling*. As future work, and knowing that an increase in the number of primitives might generate higher degree algebraic surfaces, it is our intention to develop some strategies to overcome possible limitations that may occur in these cases. As a final remark, we believe that this new method is an important contribution for the geometric modeling research field.

4 ACKNOWLEDGEMENTS

This research has been partially supported by the Portuguese Research Council (Fundação para a Ciência e Tecnologia), under the FCT Project UID/EEA/50008/2019. We are also grateful to reviewers for their suggestions to improve this paper.

5 REFERENCES

[Baj88] Chandrajit L. Bajaj. Geometric modeling with algebraic surfaces. In *Proceedings of the 3rd IMA Conference on the Mathematics of Surfaces, Keble College, Oxford, September 19-21, 1988*, pages 3–48, 1988.

[Bat01] S.S. Batsanov. Van der Waals radii of elements. *Inorganic Materials*, 37(9):871–885, 2001.

[Bed92] Sanjeev Bedi. Surface design using functional blending. *Computer-Aided Design*, 24:505–511, 09 1992.

[BGA04] Aurélien Barbier, Eric Galin, and Samir Akkouche. Complex skeletal implicit surfaces with levels of detail. *Journal of WSCG*, 12(1-3), 2004.

[Bli82] James F. Blinn. A generalization of algebraic surface drawing. In *SIGGRAPH '82: Proceedings of the 9th Annual Conference on Computer Graphics and Interactive Techniques*, page 273, New York, NY, USA, 1982. ACM.

[BMR+99] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.

[Bon64] A. Bondi. Van der Waals Volumes and Radii. *J Phys Chem*, 68(3):441–451, 1964.

[BX97] Chandrajit L. Bajaj and Guoliang Xu. Spline approximations of real algebraic surfaces. *Journal of Symbolic Computation*, 23(2-3):315–333, 1997.

[CCD00] F.L. Chen, C.S. Chen, and J.S. Deng. Blending pipe surfaces with piecewise algebraic surfaces. *Chinese Journal of Computers*, 23(9):911–916, 2000.

[ČS04] Martin Čermák and Václav Skala. Curvature dependent polygonization by the edge spinning. In Antonio Laganá, Marina L. Gavrilova, Vipin Kumar, Youngsong Mun, C. J. Kenneth Tan, and Osvaldo Gervasi, editors, *Computational Science and Its Applications – ICCSA 2004*, pages 325–334, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[ČS05] Martin Čermák and Vaclav Skala. Polygonization of implicit surfaces with sharp features by edge-spinning. *The Visual Computer*, 21:252–264, 05 2005.

[ČS07] Martin Čermák and Vaclav Skala. Polygonisation of disjoint implicit surfaces by the adaptive edge spinning algorithm of implicit objects. *Int. J. Comput. Sci. Eng.*, 3(1):45–52, July 2007.

[DG11] Sérgio E.D. Dias and Abel J.P. Gomes. Graphics processing unit-based triangulations of blinn molecular surfaces. *Concurrency and Computation: Practice and Experience*, 23(17):2280–2291, 2011.

[DG15] Sérgio E.D. Dias and Abel J.P. Gomes. Triangulating molecular surfaces over a lan of gpu-enabled computers. *Parallel Computing*, 42:35–47, February 2015.

[DNJG17] Sérgio E.D. Dias, Quoc Trong Nguyen, Joaquim Jorge, and Abel J.P. Gomes. Multi-gpu-based detection of protein cavities using critical points. *Future Generation Computer Systems*, 67:430–440, February 2017.

[Gom03] Abel Gomes. A concise b-rep data structure for stratified subanalytic objects. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (SGP'03), Aachen, Germany, June 23-25, pages 83–93. Eurographics Associ-

 ISBN 978-80-86943-37-4

ation, 2003.

[GRM99] Abel Gomes, Chris Reade, and Alan Middleditch. A mathematical model for boundary representations of n-dimensional geometric objects. In *Proceedings of the 5th Symposium on Solid Modeling and Applications* (SPM'99), Ann Arbor, Michigan, USA, June 8-11, pages 270–277. ACM Press, 1999.

[GVJ+09] Abel Gomes, Irina Voiculescu, Joaquim Jorge, Brian Wyvill, and Callum Galbraith. *Implicit Curves and Surfaces: Mathematics, Data Structures and Algorithms*. Springer Publishing Company, Inc., 1st edition, 2009.

[HH85] C. Hoffmann and J. Hopcroft. Automatic surface generations in computer aided design. *The Visual Computer*, 1(2):92–100, August 1985.

[KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the 4th Eurographics Symposium on Geometry Processing* (SGP'06), Cagliari, Sardinia, Italy, June 26-28, pages 61–70, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[LHRS05] Florian Levet, Julien Hadim, Patrick Reuter, and Christophe Schlick. Anisotropic sampling for differential point rendering of implicit surfaces. In *WSCG The 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*. UNION Agency - Science Press, Plzen, Czech Republic, 2005.

[LPP06] S. Lazard, L. M. Peñaranda, and S. Petitjean. Intersecting quadrics: an efficient and exact implementation. *Computational Geometry: Theory and Applications*, 35(1-2):74–99, 2006.

[Mur91] Shigeru Muraki. Volumetric shape description of range data using "blobby model". *SIGGRAPH Computer Graphics*, 25(4):227–235, July 1991.

[NHK+85] Hiromitsu Nishimura, Masashi Hirai, Tsuyoshi Kawai, Tory Kawata, Isao Shirakawa, and Kengo Omura. Object modelling by distribution function and a method of image generation. *Transactions of the IEICE Japan*, J68-D(4):718–725, 1985.

[PK89] Markus Pilz and Hussein Kamel. Creation and boundary evaluation of CSG-models. *Engineering with Computers*, 5(2):105–118, 1989.

[RG06] Adriano N. Raposo and Abel J.P. Gomes. Polygonization of multi-component non-manifold implicit surfaces through a symbolic-numerical continuation algorithm. In *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia* (GRAPHITE'06), Kuala Lumpur, Malaysia, November 29 - December 02, pages 399–406, New York, NY, USA, 2006. ACM.

[RG12] Adriano N. Raposo and Abel J. P. Gomes. 3D molecular assembling of B-DNA sequences using nucleotides as building blocks. *Graph. Models*, 74(4):244–254, July 2012.

[RG14] Adriano N. Raposo and Abel J.P. Gomes. Efficient deformation algorithm for plasmid DNA simulations. *BMC Bioinformatics*, 15:301, Sep 2014.

[RG15] Adriano N. Raposo and Abel J. P. Gomes. isDNA: A tool for real-time visualization of plasmid DNA monte-carlo simulations in 3D. In *Bioinformatics and Biomedical Engineering*, pages 566–577. Springer International Publishing, 2015.

[RQG09] Adriano N. Raposo, J. Queiroz, and Abel J.P. Gomes. Triangulation of molecular surfaces using an isosurface continuation algorithm. In *Proceedings of the 2009 International Conference on Computational Science and Its Applications*, ICCSA '09, pages 145–153. IEEE Computer Society, 2009.

[RV85] Aristides Requicha and Herbert Voelcker. Boolean operations in solid modeling: Boundary evaluation and merging algorithm. *Proceedings of the IEEE*, 73(1):30–43, 1985.

[War89] J. Warren. Blending algebraic surfaces. *ACM Transactions on Graphics*, 8(4):263–278, 1989.

[WMW86] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, Aug 1986.

[WZ00] T. R. Wu and Y. S. Zhou. On blending of several quadratic algebraic surfaces. *Computer Aided Geometric Design*, 17:759–766, 2000.

 ISBN 978-80-86943-37-4

ISBN 978-80-86943-37-4

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

Collateral effects of the Kalman Filter on the Throughput of a Head-Tracker for Mobile Devices

Maria Francesca Roig-Maimó
University of Balearic Islands
Cra. Valldemossa, km 7.5.
Spain 07122, Palma
xisca.roig@uib.es

Ramon Mas-Sansó
University of Balearic Islands
Cra. Valldemossa, km 7.5.
Spain 07122, Palma
ramon.mas@uib.es

ABSTRACT

We have developed an image-based head-tracker interface for mobile devices that uses the information of the front camera to detect and track the user's nose position and translate its movements into a pointing metaphor to the device. However, as already noted in the literature, the measurement errors of the motion tracking leads to a noticeable jittering of the perceived motion. To counterbalance this unpleasant and unwanted behavior, we have applied a Kalman filter to smooth the obtained positions. In this paper we focus on the effect that the use of a Kalman filter can have on the throughput of the interface. Throughput is the human performance measure proposed by the ISO 9241-411 for evaluating the efficiency and effectiveness of non-keyboard input devices. The softness and precision improvements that the Kalman filter infers in the tracking of the cursor are subjectively evident. However, its effects on the ISO's throughput have to be measured objectively to get an estimation of the benefits and drawbacks of applying a Kalman filter to a pointing device.

Keywords

Kalman filter, head-tracker, throughput, Fitts' law, HCI, mobile devices.

1 INTRODUCTION

Head-trackers provide a hands-free way to interact with devices through the movements of the head and so, they have a direct application in assistive tools for motor-impaired users. In the assistive domain technologies, such interfaces are widely used for desktop computers [MYPVP10, MGiSLVG06] and in several commercial mobile applications [DSLKT03, GB].

Research on head tracker interfaces based on image sensors for desktop computers is a mature discipline and has been conducted for a long time for HCI purposes [Toy98, BGF02, CMM+09, VMYP08]. Nevertheless, nowadays the advent of integrated frontal cameras has focused this kind of research on mobile devices.

We have developed an image-based head-tracker interface for mobile devices [RMMYV16] that only uses the information of the front camera to detect and track the user's nose position and translate its movements into a pointing metaphor to the device. However, as already noted in the literature [CRV12], the measurement errors of the motion tracking leads to a noticeable jittering of the perceived motion. To counterbalance this unpleasant and unwanted behavior, we have applied a Kalman filter to smooth the obtained positions.

In this paper we focus on the effect that the use of a Kalman filter can have on the throughput of the developed interface. Throughput is the human performance measure proposed by the ISO 9241-411 [ISO12] for evaluating the efficiency and effectiveness of non-keyboard input devices.

The softness and precision improvements that the Kalman filter infers in the tracking of the cursor is subjectively evident. Nevertheless, its effects on the final throughput also have to be measured objectively to get an unbiased estimation of the benefits and drawbacks of applying a Kalman filter to a pointing device.

There have been some attempts to generally depict the lag that filtering inherently introduces [CRV12] but, to the best of our knowledge, there are no clues on the effects of the Kalman filter on ISO's throughput.

2 HEAD-TRACKER INTERFACE

FaceMe [RMMYV16] is a head-tracker interface for mobile devices that uses the information of the front camera to detect and track the user's nose position and translate its movements into interaction actions to the device (Figure 1).

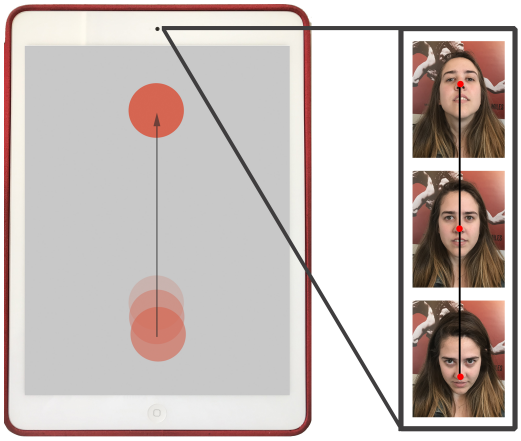


Figure 1: Example of using FaceMe as a pointing device.

A version of SINA system [VMYP08], a camera-based head-tracker interface for desktop environment, was adapted and optimized for mobile devices.

The interface is based on facial feature tracking instead of tracking the overall head or face. The selected facial feature region is the nose, because it has specific characteristics to allow tracking, it is not occluded by facial hair or glasses, and it is always visible while the user is interacting with the mobile device (even when the head is rotated).

The process is divided into two stages: the *User detection* stage and the *Tracking* stage. In the *User detection* stage we process the initial frames from the camera to detect the user's facial features to be tracked. After detection, the *Tracking* stage performs the tracking and filtering. Finally, the average of all the features (i.e., the nose point) is sent to a transfer function. This transfer function is responsible of the translation of the coordinates' change of the nose point to a coordinates' change on the device screen.

2.1 User Detection

In this step no calibration is needed, the only requirement is that the user must keep the head steady for a small predefined number of frames to allow the system to automatically detect the face region (see "User detected" in Figure 2).

The main face is defined as the one with the biggest area (see "Main face region" in Figure 2). To ensure a steady user for a proper algorithm initialization and to avoid false positives, we use a temporal consistency scheme (see "Temporal consistency" in Figure 2).

According to anthropometrical measurements of the human face [Sat16], the nose region occupies the second third of the facial region (see "Nose region" in Figure 3). Inside this region, the nostrils and the corners of the nose are selected as the initial facial features to track (see "Facial features" in Figure 3).

Figure 2: Illustrated theoretical stages for the detection of the main user face.

Changing light conditions can lead to the selection of unstable features, therefore we need to re-select the initial facial features using symmetry constraints (respect to the vertical axis). This leads to a more robust tracking process.

The finally chosen nose point is the average of all the facial features being tracked, which will be centered on the nose, between the nostrils (see "Nose point" in Figure 3).

Figure 3: Simulated steps of the *User detection* stage.

The *User detection* stage works in a wide range of lighting conditions (dark or clear), users particularities (skin color, glasses or facial hair) and backgrounds (homogeneous or heterogeneous).

2.2 Tracking

In the *Tracking* stage, there is no need for the face to be fully visible, as only an small region surrounding the nose is used.

We get the best image registration exploding the spatial intensity gradient information of the images using a pyramidal implementation of the Lukas-Kanade algorithm [Bou01]. Since the algorithm is robust to rotation, scaling and shearing, the user can move in a flexible way. However, fast head movements can cause the lost or displacement of features to track. If we detect a feature abnormally separated from the average point, this feature is discarded (see "Filtered of displaced feature" in Figure 4). In case there are not enough features to track, the *User detection* stage restarts.

We follow a typical Bayesian approach to sensor fusion, combining measurements in the representation of a posterior probability. For each new frame, we combine the tracked nose features with newly detected features (see "Fusion" in Figure 4).

After this stage, we apply the velocity constant Kalman filter to get rid of the jittering.

ISBN 978-80-86943-37-4

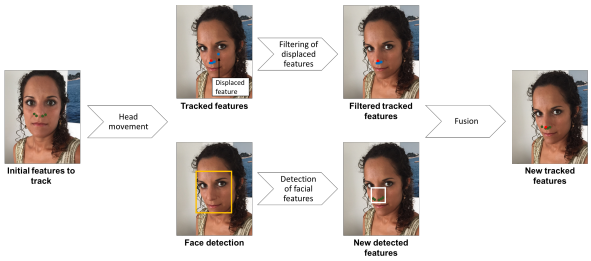


Figure 4: Simulated steps of the *Tracking* stage.

Our tracking stage is able to run in real-time on current mobile devices with a variety of CPU platforms.

A detailed description of the system is found in other sources [RMMYV16].

2.3 The Kalman filter

The Kalman filter is a powerful mathematical tool to be used when working with real world inaccurate measurements. It was first introduced in 1960 [Kal60] and it is still commonly used in a broad range of disciplines including satellite navigation systems [SHiS14], object and people tracking [PAHEM09] [SR11] or autonomous navigation [LFL+18].

The Kalman filter is an optimal estimation of the state of a process, in a way that minimizes the mean of the squared error. Its implementation is very fast and its memory requirements are very low, as there is no need to reprocess previously observed data.

Kalman filter algorithms work in the continuous iteration of two steps. In the first step, we update the state of our system using the dynamic model (prediction), and in the second step we update our measurement with the observation model (correction).

Our goal when using the Kalman filter is to find an estimation of the cursor position such that we obtain a smoother motion, reducing the jittering. So, in our implementation of the filter, the state of our system at time t is described with a position p and a velocity v, defining the state of the nose:

$$\vec{x_t} = (p, v)$$

The position and the velocity are correlated (the higher the velocity, the farther the motion and the slower the velocity, the nearer the motion). This correlation is described in a covariance matrix P_t where each element corresponds to the level of correlation between the couples position-velocity:

$$P_t = \begin{pmatrix} C_{pp} & C_{pv} \\ C_{vp} & C_{vv} \end{pmatrix}$$

At a time t, we need to know an estimation of the state of the system $\hat{x_t}$:

$$\hat{x_t} = \begin{pmatrix} p \\ v \end{pmatrix}$$

And, from the current state, we have to predict the next one $\hat{x_t}$.

Let use simple kinematics:

$$p_t = p_{t-1} + \Delta t v_{t-1}$$

$$v_t = v_{t-1}$$

From which we can build a prediction matrix F_t:

$$\hat{F_t} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix},$$

such that,

$$\hat{x_t} = F_t \hat{x}_{t-1} \tag{1}$$

At time t, we also need to keep track of the covariance matrix (i.e. the prediction of the new uncertainty). We have to compute the new covariance matrix using the prediction matrix. If we multiply every element in a distribution by the prediction matrix, we get:

$$P_t = F_t P_{t-1} F_t^T$$

At this point, we can also add some additional uncertainty from the process noise expanding the covariance by adding the term Q_t:

$$P_t = F_t P_{t-1} F_t^T + Q_t \tag{2}$$

Equation 1 and Equation 2 are used to estimate the state of the system and the covariance projecting them from time t to time $t-1$ in the prediction step.

In the correction step, we first have to compute the Kalman gain K, using the matrix H that models the sensors relating the state with the measurements and the covariance of the observation noise R:

$$K = H_t P_t H_t^T (H_t P_t H_t^T + R_t)^{-1}$$

And now, we can state the equations for the correction step:

$$P_t' = P_t - K' H_t P_t \tag{3}$$

$$\hat{x_t'} = \hat{x_t} + K'(\vec{z_t} - H_t \hat{x_t}) \tag{4}$$

Where $\vec{z_t}$ is the reading we have observed.

We have tuned the noise parameters so that jittering is correctly compensated in most use conditions.

Figure 5 depicts the desired trajectory, the raw data trajectory (No Kalman) and the Kalman filtering results. Although using a short path, the jittering of the red measures are clearly visible and very user noticeable in the interactive application.

ISBN 978-80-86943-37-4

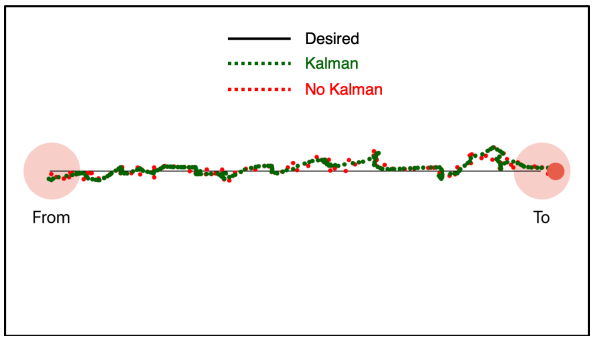


Figure 5: Desired, measured and filtered trajectories.

3 ISO TESTING AND THE CALCULATION OF THROUGHPUT

ISO 9241-411 [ISO12] describes performance tests for evaluating the efficiency and effectiveness of existing or new non-keyboard input devices[1]. The primary tests involve target-select tasks using throughput as a dependent variable.

The calculation of throughput is performed over a range of amplitudes (A) and with a set of target widths (W) involving tasks for which computing devices are intended to be used.

The ISO standard proposes a one-directional target-select test and a multi-directional target-select test. Due to the two-dimensional nature of the pointing metaphor, the multi-directional test is better suited for our requirements.

3.1 Multi-directional Target-select Test

The multi-directional test evaluates target-select movements in different directions. The user moves the cursor across a layout circle to sequential targets of width W equally spaced around the circumference of the circle with diameter A (see Figure 6). Each sequence of trials begins and ends in the top target and alternates on targets moving across and around a layout circle.

3.2 The Calculation of Throughput

The ISO standard specifies throughput (TP) as the performance measure and it is calculated as follows:

$$TP = \frac{\text{Effective index of difficulty}}{\text{Movement time}} = \frac{ID_e}{MT}, \qquad (5)$$

where ID_e is computed from the movement amplitude (A) and target width (W) and MT is the per-trial movement time averaged over a sequence of trials.

[1] ISO 9241-411 [ISO12] is an updated version of ISO 9241-9 [ISO02]. With respect to performance evaluation, the two versions of the standard are the same.

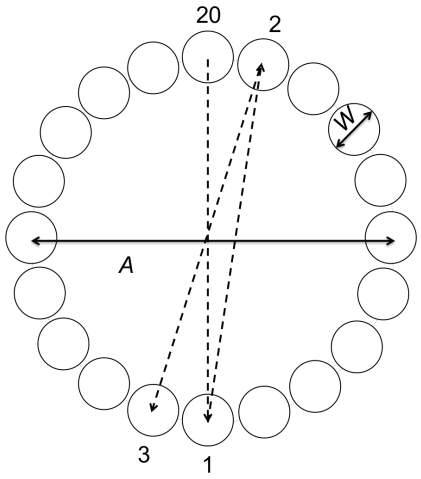


Figure 6: ISO Multi-directional target-select test.

The effective index of difficulty is a measure, in bits, of the difficulty and user precision achieved in accomplishing a task:

$$ID_e = log_2\left(\frac{A_e}{W_e} + 1\right), \qquad (6)$$

where W_e is the effective target width, calculated from the width of the distribution of selection coordinates made by a participant over a sequence of trials. The effective target width is calculated as follows:

$$W_e = 4.133 \cdot S_x, \qquad (7)$$

where S_x is the standard deviation of the selection coordinates in the direction that movement proceeds. The effective value is used to include spatial variability in the calculation. The effective amplitude (A_e) can also be used if there is an overall tendency to overshoot or undershoot. A_e is calculated as the mean movement distance from the start-of-movement position to the end points [SM04].

Using the effective values, throughput is a single human performance measure that embeds both the speed and accuracy in human responses. A detailed description of the calculation of throughput is found in other sources [SM04, Mac15, RMMMYV17].

4 THE EXPERIMENT

The main goal of the experiment is to evaluate the mobile head-tracker interface following the recommendations described in the ISO standard in order to obtain a benchmark value of throughput. This will allow the comparison between the two different implementations of the head-tracker interface: by using the position obtained using the Kalman filter or by using the raw position directly.

ISBN 978-80-86943-37-4

4.1 Participants

Twelve participants (5 females) were recruited from the local town university campus in Spain. Ages ranged from 22 to 52 with a mean of 31.25 years (SD = 10.67). There were no requirements on prior experience to participate in the study. None of the participants had previous experience with head-tracker interfaces.

4.2 Apparatus

The experiment was conducted on an Apple *iPad Air* with a resolution of 2048 × 1536 px and a pixel density of 264 ppi. This corresponds to a resolution of 1024 × 768 Apple points.[2] All communication with the tablet was disabled during testing.

The software implemented the ISO multi-directional target-select test (see Figure 7 for details).

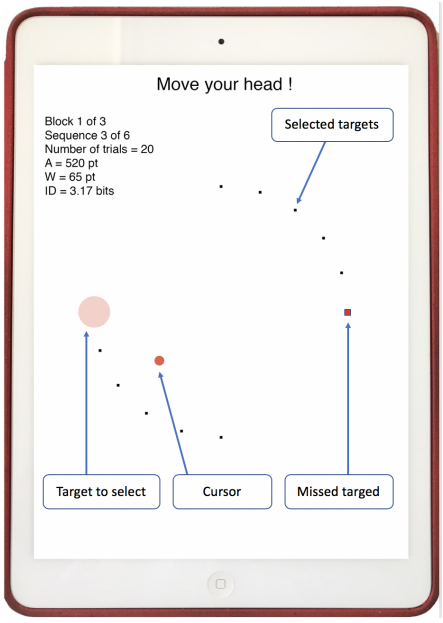


Figure 7: Screenshot of the experiment software: example target condition with annotations (A = 1040 px, W = 260 px).

User input combined the mobile head-tracker for pointing and touch for selection.

Each sequence consisted of 20 targets with the target to select highlighted for each trial. Upon selection, a new target was highlighted. Selections proceeded in a pattern moving across and around the layout circle until all targets were selected. If a target was missed, a small red square appeared in the center of the missed target; otherwise, a small black square appeared showing a correct selection. The target was highlighted in green when the cursor was inside it.

[2] Apple's point (pt.) is an abstract unit that covers two pixels on retina devices. On the *iPad Air*, one point equals 1/132 inch (Note: 1 mm ≈ 5 pt.).

4.3 Procedure

After signing a consent form, participants were briefed on the goals of the experiment and were instructed to sit and hold the device in portrait orientation in a comfortable position (see Figure 8). The only requirement was that their entire face was visible by the front camera of the device.

Figure 8: Participant performing the experiment.

The experiment task was demonstrated to participants, after which they did a few practice sequences. They were instructed to move the cursor by holding the device still and moving their head. Selection occurred by tapping anywhere on the display surface with a thumb when the cursor was inside the target. Testing began after they felt comfortable with the task and the interaction method.

Participants were asked to select targets as quickly and accurately as possible and to leave errors uncorrected. They were told that missing an occasional target was OK, but that if many targets were missed, they should slow down. They were allowed to rest as needed between sequences. Testing lasted about 20 minutes per participant.

4.4 Design

The experiment was fully within-subjects with the following independent variables and levels:

- Filtering mode: Kalman, No Kalman.
- Block: 1, 2, 3.
- Amplitude: 260, 520, 1040 px.
- Width: 130, 260 px.

The primary independent variable was filtering mode: by applying a velocity constant *Kalman* filter to smooth the positions returned by the head-tracker interface (*Kalman filtering mode*) or by using the raw positions

ISBN 978-80-86943-37-4

directly (*No Kalman filtering mode*). Block, amplitude, and width were included to gather a sufficient quantity of data over a reasonable range of task difficulties (with *ID*s from 1.00 to 3.17 bits).

For each condition, participants performed a sequence of 20 trials. The two filtering modes were assigned using a Latin square with 6 participants per order. The amplitude and width conditions were randomized within blocks.

The dependent variables were throughput, movement time, and error rate.

The total number of trials was 12 participants × 2 interaction modes × 3 blocks × 3 amplitudes × 2 widths × 20 trials = 8,640.

5 RESULTS

In this section, results are given for throughput, movement time and error rate.

5.1 Learning Effects

Since head-tracking was unfamiliar to all participants, a learning effect was expected. Figure 9 shows the learning effect for throughput by filtering mode. The learning effect (i.e., block effect) was statistically significant ($F_{2,22} = 11.36, p < .001$), confirming the expected improvement with practice. The effect was more pronounced between the 1st and 2nd blocks, with 8.65% increase in throughput, compared to a almost indiscernible decrease of 0.97% between the 2nd and 3rd blocks. A Scheffé post hoc analysis confirmed that the effect was not significant after block 1. As throughput is the dependent variable specified in ISO 9241-411, subsequent analyses are based on the pooled data from the 2nd and 3rd blocks of testing.

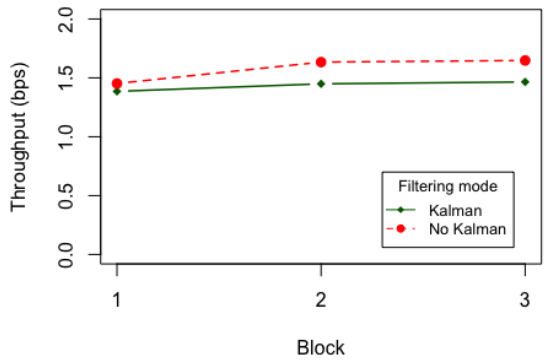


Figure 9: Results for filtering mode and block for throughput.

5.2 Throughput

The grand mean for throughput was 1.55 bps. This value is within the expected range for head input on mobile and desktop environments (from 1.28 bps to 2.10 bps [MFM15, DSLKT03, RMMMYV18]).

The mean throughput for the *No Kalman* filtering mode was 1.58 bps, which was 10.5% higher than the mean throughput of 1.43 bps for the *Kalman* filtering mode. The difference was statistically significant ($F_{1,11} = 7.63, p < .05$).

5.3 Movement Time

The grand mean for movement time was 1.44 s per trial. By filtering mode, the means were 1.58 s (*Kalman*) and 1.42 s (*No Kalman*). The difference was statistically significant ($F_{1,11} = 5.92, p < .05$).

5.4 Error Rate

The grand mean for error rate was 5.58% per sequence. By filtering mode, the means were 5.70% (*Kalman*) and 5.37% (*No Kalman*). The difference was not statistically significant ($F_{1,11} = 0.37$, ns).

6 CONCLUSION AND DISCUSSION

In this contribution, we show that to indiscriminately apply a Kalman filter to our data may lead to a decrease on the human performance in terms of the throughput of our head-tracker.

Our results show that when using the Kalman filter to smooth the positions returned by the head-tracker interface, the throughput is up to a 9.5% lower than when using the raw positions detected in the original images. Therefore, it has a negative effect on the throughput of the interface. Whether this effect is compensated by the very noticeable absence of jitter, it has to be decided depending on the application.

Results also show that although the use of the Kalman filter had no effect on the accuracy of the head-tracker in terms of error rate, it also has a significant negative effect in terms of velocity.

In the near future we are planning to evaluate the effect that some low-pass filters like the 1€ Filter [CRV12] can have on the throughput of the head-tracker used as a pointing device.

7 ACKNOWLEDGMENTS

This work has been partially supported by the project TIN2016-81143-R (AEI/FEDER, UE). We also thank the Balearic Islands University and its Department of Mathematics and Computer Science for their support.

 ISBN 978-80-86943-37-4

8 REFERENCES

[BGF02] M. Betke, J. Gips, and P. Fleming. The camera mouse: Visual tracking of body features to provide computer access for people with severe disabilities. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10(1):1–10, March 2002.

[Bou01] J. Y. Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5(1-10):4, 2001.

[CMM+09] Fernando Caballero, Iván Maza, Roberto Molina, David Esteban, and Aníbal Ollero. A robust head tracking system based on monocular vision and planar templates. *Sensors*, 9(11):8924–8943, 2009.

[CRV12] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 1€ filter: A simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2527–2530, New York, NY, USA, 2012. ACM.

[DSLKT03] Gamhewage C De Silva, Michael J Lyons, Shinjiro Kawato, and Nobuji Tetsutani. Human factors evaluation of a vision-based facial gesture interface. In *Proceedings of the Computer Vision and Pattern Recognition Workshop - CVPRW 2003*, pages 52–52, New York, 2003. IEEE.

[GB] Google and Beit Issie Shapiro. Go Ahead project.

[ISO02] ISO. 9241–9. 2000. Ergonomics requirements for office work with visual display terminals (VDTs) – part 9: Requirements for non-keyboard input devices. *International Organization for Standardization*, 2002.

[ISO12] ISO. 9241–411. 2012. Ergonomics of human-system interaction – part 411: Evaluation methods for the design of physical input devices. *International Organization for Standardization*, 2012.

[Kal60] R E Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, mar 1960.

[LFL+18] Yahui Liu, Xiaoqian Fan, Chen Lv, Jian Wu, Liang Li, and Dawei Ding. An innovative information fusion method with adaptive kalman filter for integrated ins/gps navigation of autonomous vehicles. *Mechanical Systems and Signal Processing*, 100:605 – 616, 2018.

[Mac15] I Scott MacKenzie. Fitts' throughput and the remarkable case of touch-based target selection. In *Proceedings of the 17th International Conference on Human-Computer Interaction - HCII 2015*, pages 238–249, Switzerland, 2015. Springer.

[MFM15] John Magee, Torsten Felzer, and I Scott MacKenzie. Camera Mouse + ClickerAID: Dwell vs. single-muscle click actuation in mouse-replacement interfaces. In *Proceedings of the 17th International Conference on Human-Computer Interaction - HCII 2015*, pages 74–84, Switzerland, 2015. Springer.

[MGiSLVG06] César Mauri, Toni Granollers i Saltiveri, Jesús Lorés Vidal, and Mabel García. Computer vision interaction for people with severe movement restrictions. *Human Technology: An Interdisciplinary Journal on Humans in ICT Environments*, 2(1):38–54, 2006.

[MYPVP10] Cristina Manresa-Yee, Pere Ponsa, Javier Varona, and Francisco J. Perales. User experience to improve the usability of a vision-based interface. *Interacting with Computers*, 22(6):594–605, 2010.

[PAHEM09] Saira Saleem Pathan, Ayoub Al-Hamadi, Mahmoud Elmezain, and Bernd Michaelis. Feature-supported multi-hypothesis framework for multi-object tracking using kalman filter. In *WSCG 2009: Full Papers Proceedings: The 17th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, WSCG '09, pages 197–202, University of West Bohemia, Plzen, Czech Republic, 2009.

[RMMMYV17] Maria Francesca Roig-Maimó, I. Scott MacKenzie, Cristina Manresa-Yee, and Javier Varona. Evaluating fitts' law performance with a non-iso task. In *Proceedings of the XVIII International Conference on*

 ISBN 978-80-86943-37-4

Human Computer Interaction, Interacción '17, pages 5:1–5:8, New York, NY, USA, 2017. ACM.

[RMMMYV18] Maria Francesca Roig-Maimó, I. Scott MacKenzie, Cristina Manresa-Yee, and Javier Varona. Head-tracking interfaces on mobile devices: Evaluation using fitts'law and a new multi-directional corner task for small displays. *International Journal of Human-Computer Studies*, 112:1 – 15, 2018.

[RMMYV16] Maria Francesca Roig-Maimó, Cristina Manresa-Yee, and Javier Varona. A robust camera-based interface for mobile entertainment. *Sensors*, 16(2), 2016.

[Sat16] Robert T Sataloff. *Sataloff's Comprehensive Textbook of Otolaryngology: Head & Neck Surgery: Facial Plastic and Reconstructive Surgery*, volume 3. JP Medical Ltd, 2016.

[SHiS14] Halil Ersin Soken, Chingiz Hajiyev, and Shin ichiro Sakai. Robust kalman filtering for small satellite attitude estimation in the presence of measurement faults. *European Journal of Control*, 20(2):64 – 72, 2014.

[SM04] R William Soukoreff and I Scott MacKenzie. Towards a standard for pointing device evaluation: Perspectives on 27 years of Fitts' law research in HCI. *International Journal of Human-Computer Studies*, 61(6):751–789, 2004.

[SR11] Beril Sirmacek and Peter Reinartz. Kalman filter based feature analysis for tracking people from airborne images. In *ISPRS workshop high-resolution earth imaging for geospatial information, Hannover, Germany*, 2011.

[Toy98] Kentaro Toyama. "look, ma - no hands!" hands-free cursor control with real-time 3d face tracking. *PUI98*, 1998.

[VMYP08] Javier Varona, Cristina Manresa-Yee, and Francisco J. Perales. Hands-free vision-based interface for computer accessibility. *Journal of Network and Computer Applications*, 31(4):357 – 374, 2008.

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

Performance Evaluation and Comparison of Service-based Image Processing based on Software Rendering

Ole Wegen
Hasso Plattner Institute,
Faculty of Digital Engineering,
University of Potsdam, Germany
ole.wegen@student.hpi.de

Matthias Trapp
Hasso Plattner Institute,
Faculty of Digital Engineering,
University of Potsdam, Germany
matthias.trapp@hpi.de

Jürgen Döllner
Hasso Plattner Institute,
Faculty of Digital Engineering,
University of Potsdam, Germany
juergen.doellner@hpi.de

Sebastian Pasewaldt
Digital Masterpieces GmbH
Germany
sebastian.pasewaldt@digitalmasterpieces.com

ABSTRACT

This paper presents an approach and performance evaluation of performing service-based image processing using software rendering implemented using Mesa3D. Due to recent advances in cloud computing technology (w.r.t. both, hardware and software) as well as increased demands of image processing and analysis techniques, often within an eco-system of devices, it is feasible to research and quantify the impact of service-based approaches in this domain w.r.t. cost-performance relation. For it, we provide a performance comparison for service-based processing using GPU-accelerated and software rendering.

Keywords

cloud-rendering, image processing, software rendering, performance comparison

1 INTRODUCTION

1.1 Motivation

Today, image processing is a common task with various applications ranging from processing high-quality professional content to User-Generated Content (UGC). Within recent years, simultaneous developments with respect to the following major directions can be observed: (1) an increase of mobile hardware and processing capabilities, (2) an increase in cloud-processing capabilities and infrastructure, as well as (3) the expected increase of network throughput and infrastructure, with new transmission standards such as 5G and support for mobile devices such as Google Cloud Messaging (GCM).

There are numerous applications to service-based provisioning of image processing functionality, both Business-to-Customer (B2C) and Business-to-Business (B2B) application scenarios. Besides the protection or Digital Rights Management (DRM) of processing functionality internals or software source code [12], the most prominent is the integration of such services into collaborative web-based [2] and mobile applications as well as cloud-processing services. However, such a service-based provision has to account for two major aspects:

GPU-based Processing: For efficient processing, image processing components often rely on hardware-acceleration based on Graphics Processing Units (GPUs), thus require dedicated graphics hardware (GPU). This is often the case for real-time or high-performance applications. It should be taken into account that not only the actual processing time influences whether an application is real-time capable. Also the network speed needs to be considered.

Scalability: Providing image-processing functionality via cloud-processing services to end users relies on scalability features offered by cloud-service providers. However, currently dedicated GPU-instances are costly, thus infrastructures do not scale well for usability in the B2C market.

Taken both of the aspects into account when providing service-based image processing for customers, the financial impact varies significantly between hosting a dedicated GPU-based server or rely on scalable GPU-based cloud-computing services such as Amazon AWS Elastic Compute Cloud (EC2) (ranging from approx. €120 for a dedicated server to $2000 for a scalable one). However, one important aspect of service-based provisioning of any functionality in general, is the coverage of operational cost for the server infrastructure. With respect to this, one observation that can be made is the cost span between server with and without dedicated GPU hardware. To reduce costs while maintaining scalability simultaneously, *Software Rendering (SWR)* is a promising alternative to servers and services that support dedicated GPUs.

1.2 Problem Statement

Compared to GPU-based rendering, the run-time performance of SWR is exspected to be significantly inferior. Thus, especially for high spatial resolutions of input images, SWR is not suitable for applications that require fast, on-demand results, but it can be used for processing tasks where speed or output quality plays not a crucial role, e.g., for batch processing or preview generation. This work relies on an image-processing framework based on C++ and OpenGL that is an integral component of various desktop and mobile image-abstraction applications; given an input image and a description of an image-processing operation, it computes a transformed output image using GPU-acceleration.

However, depending on (1) the *type of spatial processing technique*, i.e., pixel-based, neighborhood-based, or global operations, as well as (2) the choice of *implementation*, there are approaches and frameworks that can counterbalance some of the negative run-time performance impact. Nevertheless, when using SWR there are arguments — specific to software development aspects — to rely on standard rendering and graphics Application Programming Interfaces (APIs) for implementation of image processing techniques:

Standardization: Using standardized APIs backed by industry and research allows for fast adaption to new software and hardware technology as well as ease the integration of new processing algorithms and techniques.

Software Maintenance: Software maintenance effort and costs can be lowered by relying only on a single framework for image-processing.

1.3 Approach and Contributions

This paper approaches the challenge of enabling SWR for image processing as follows: first, it integrates software rendering by using GPU emulation via The Mesa 3D Graphics Library (Mesa3D). This allows for a wide support of cloud-computing providers and thus facilitates vertical (adding processing power) and horizontal (adding computing instances) scaling. Based on this integration approach, performance measurements are obtained for comparing a dedicated server with GPU support and standard servers running the GPU-emulation.

The remainder of the paper is structured as follows. Section 2 reviews related work regarding service-based approaches for image and video processing systems and techniques. Section 3 describes the approach and implementation details for integrating hardware and software rendering. Section 4 presents and discusses results of a performance comparison between SWR and GPU-based processing in a service-based environment. Finally, Section 5 concludes this paper and discusses future research directions.

2 RELATED WORK

2.1 Software Rendering Approaches

Besides special approaches for the implementation of high-performance software rasterization using GPUs [18, 14], only a few research focus on software rendering in general.

Mileff and Dudra presents an overview of performance improvement methods for Central Processing Units (CPUs) by utilizing specific instruction sets and describe how these methods can be applied in (tile-based) software rendering [20]. Following this, the authors reviews problems and opportunities of two-dimensional rendering and propose and evaluate an efficient, software-based rasterization method for textures [21]. Mesa3D in particular, is used for in-situ visualization in a particle-based volume rendering Kyoto Visualization System [8].

2.2 Service-based Image Processing

Several software architectural patterns are feasible for implementing service-based image-processing [4, 25]. However, one prominent style of building a web-based processing system for any data is the service-oriented architecture [36]. It enables server developers to set up various processing endpoints, each providing a specific functionality and covering a different use case. These endpoints are accessible as a single entity to the client, i.e., the implementation is hidden for the requesting clients, but can be implemented through an arbitrary number of self-contained services.

Since web services are usually designed to maximize their reusability, their functionality should be simple and atomic. Therefore, the composition of services [10] is critical for fulfilling more complex use cases [16]. The two most prominent patterns for implementing such composition are *choreography* and

 ISBN 978-80-86943-37-4

orchestration [26]. The choreography pattern describes decentralized collaboration directly between modules without a central component. The orchestration pattern describes collaboration through a central module, which requests the different web services and passes the intermediate results between them [28].

In the field of image analysis, Wursch *et al.* [40, 41] present a web-based tool that enables users to perform various image analysis methods, such as text-line extraction, binarization, and layout analysis. It is implemented using a number of Representational State Transfer (REST) web services and application examples include multiple web-based applications for different use cases.

Further, the viability of implementing image-processing web services using REST has been demonstrated by Winkler *et al.* [38], including the ease of combination of endpoints. Another example for service-based image-processing is Leadtools (https://www.leadtools.com), which provides a fixed set of approx. 200 image-processing functions with a fixed configuration set via a web API. In this work, however, a similar approach using REST is chosen, although with a different focus in terms of granularity of services.

Applications with respect to medical image processing are presented by Yuan *et al.* . [43] as well as Moulick and Gosh [22]. They propose a web-based platform to present and process medical images by using server-side computing for a series of image processing algorithms.

Further, in the field of geodata, the Open Geospatial Consortium (OGC) set standards for a complete server-client ecosystem. As part of this specification, different web services for geodata are introduced [23]. Each web service is defined through specific input and output data and the ability to self-describe its functionality[42]. In contrast, in the domain of general image-processing there is no such standardization yet. However, it is possible to transfer concepts from the OGC standard, such as unified data models. These data models are implemented using a platform-independent effect format [3]. In the future, it is possible to transfer even more concepts set by the OGC to the general image-processing domain, such as the standardized self-description of services.

2.3 Image Abstraction Techniques

In this work, we focus on edge-aware and content-preserving image-processing as a fundamental tool in computational photography and non-photorealistic rendering for abstraction and artistic stylization for application and testing purposes. Typical approaches that operate in the spatial domain for abstraction use a kind of anisotropic diffusion [27, 37] and are designed for parallel execution, such as approximated by the bilateral filter [35] and guided filter [9]. A plenitude of stylization techniques exist using these filters as building blocks to simulate traditional painting media and effects [13], such as cartoon [39] and oil paint [33]. However, these may become computationally expensive when applied in an iterative multi-stage process. This particularly applies to techniques using global optimizations to separate detail from base information, e.g., based on weighted least squares [6] or locally weighted histograms [11], and recent techniques that separate style from content using neural networks [7]. Because of their global optimization scheme, they are typically not suited for real-time application, in particular not on mobile devices. To this end, we implemented a variety of these techniques using the proposed image-processing service including stylization, High Dynamic Range (HDR) tone mapping and compression, JPEG artifact removal and colorization, to demonstrate its versatile application. We used a representative subset of these techniques for performance evaluation.

3 SOFTWARE RENDERING

This section describes the approach for enabling software rendering for an Open Graphics Library (OpenGL)-based rendering framework for image processing techniques [29]. In particular, based on our system requirements, this comprises justification of middle-ware choices and specifics for the deployment process that is suitable for cloud-computing providers.

3.1 Software Rendering using Mesa3D

A common approach for enabling software rendering for OpenGL-based applications [31] is using Mesa3D [19]: an open source 3D graphics library implementing OpenGL, Vulkan and other graphics API specifications. It offers multi-platform support and is used in Linux, Windows, and macOS. Basically, there are two architectures for Mesa3D driver implementation. The older one uses Mesa's Direct Rendering Infrastructure. Gallium3D represents the new architecture and API driver implementation and development by abstracting from specific hardware and operating systems. The following software drivers are available for Mesa3D:

SWRast: represent the original/legacy software rasterizer for Mesa3D implemented in C. It supports only the fixed-function rendering pipeline that was used in OpenGL before shaders were introduced.

OpenSWR: is a fast CPU OpenGL-compatible renderer developed by Intel [30] with advantages in cluster setups for large datasets.

 ISBN 978-80-86943-37-4

Table 1: Comparison of Mesa3D software rendering drivers.

	Software drivers for software rendering			
Aspect	**SWRast**	**OpenSWR**	**Softpipe**	**LLVMpipe**
API Features	OpenGL 2.0 (no GLSL)	OpenGL 3.3 Core + OpenGL 3.0 Compatibility	OpenGL 3.3 (70.9% Extension Coverage)	OpenGL 3.3 (64.8% Extension Coverage)
Performance	Not tested	High	Low	High

Softpipe: is a full Gallium reference driver for a CPU supporting the programmable graphics pipeline by implementing code generation for shader programs and closely modeling of hardware behavior.

LLVMpipe: is a fast software rasterizer that supports a sufficient amount of OpenGL core functionality and extensions. It uses LLVM for x86 Just-in-Time (JIT) code generation and is multi-threaded. It is based on LLVM [17], which is a collection of modular and reusable compiler and tool chain technologies.

Since the framework is not limited to OpenGL[31], also Vulkan[32] should be supported by SWR, which unfortunately none of the available software renderer currently supports. For choosing a software rendering approach we have to account for the following aspects:

Level of Feature-Completeness: This depends on the *up-to-dateness* of a specific software renderer. Apart from SWRast, all available software renderer fully support OpenGL 3.3.; OpenSWR is currently more actively developed while Softpipe on the other hand supports additional OpenGL extensions. Nevertheless, also LLVMpipe supports a sufficient amount of extensions.

Processing Performance: The run-time performance of SWR approaches represents the most crucial aspect and is required being as high as possible.

Table 1 shows a comparison of the available Mesa3D software drivers, based on aggregated information obtained from the Mesa3D feature set [5]. The preliminary performance values are obtained using the approach described in Section 4. Specifically, the machine used for the preliminary test has the following hardware specifications: an Intel i5-8400, 2.8 GHz processor with six cores, and 16 GB Random Access Memory (RAM). Only a separated morphological closing operation with a kernel size of three was tested as representative for multi-pass processing techniques that are based on neighborhood sampling operations (Figure 2c). Figure 1 shows the resulting performance values at a logarithmic scale. It can be observed that Softpipe was significantly slower than LLVMpipe while OpenSWR was also slower than LLVMpipe but only by a smaller factor.

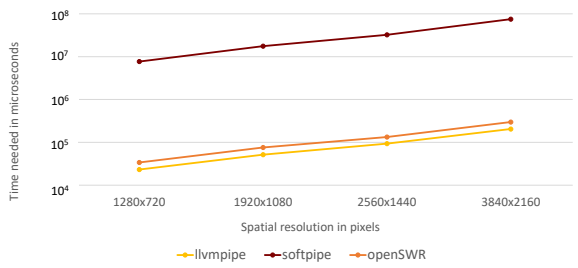


Figure 1: Runtime performance comparison in microseconds of software drivers (LLVMpipe vs. Softpipe vs. OpenSWR) using morphological closing operation with a kernel size of three (logarithmic scale).

Based on this analysis, the LLVMpipe backend for SWR is chosen for the image-processing techniques because the supported features are sufficient and the performance is compared to the other drivers the best one for this task.

3.2 Deployment Process

Similar to approaches used for real-time 3D rendering [24], the deployment process basically comprises the following steps that can be managed using an automatic approach. At first, LLVM, is compiled and linked. Subsequently, Mesa3D with LLVMpipe backend is compiled and linked. After that, the Mesa3D OpenGL dynamic linked library for Windows targets (`opengl32.dll`) or shared library for UNIX-like systems (`libGL.so`) is placed in the directory next to the executable.

Following to that, OpenGL-specific application calls are resolved using the LLVMpipe driver. Finally, for patching the application, a docker image [1] is prepared and deployed to the respective cloud-service providers.

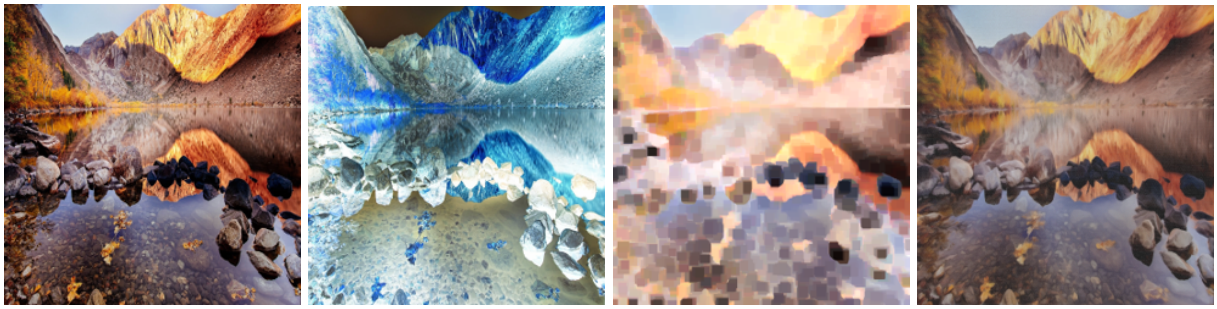
4 PERFORMANCE EVALUATION

This section evaluates, compares, and discusses the runtime performance of the service-based image processor deployed to (1) a dedicated GPU server and (2) different standard cloud-computing platforms.

4.1 Test Data and Processing Techniques

Different image resolutions were tested with different image-processing techniques to estimate the performance of software rendering regarding the spatial resolution of an image as well as the complexity of processing techniques. The following common resolutions in pixels were chosen: 1280×720 (HD), 1920×1080 (FHD), 2560×1440 (QHD), and 3840×2160 (4K).

ISBN 978-80-86943-37-4



(a) Original. (b) Color invert. (c) Morphological closing. (d) Oilpaint.

Figure 2: Exemplary results of different image-processing techniques (b) – (d) that are used for performance evaluation and comparison applied to an input image (a).

In addition to various spatial resolutions, different image processing techniques (Figure 2) have been tested in order to cover a broad spectrum and obtain variant estimates on software rendering performance and behavior with respect to different types of processing tasks:

Color Invert: This operation is pixel-based and inverts the color value of every pixel. This requires only a single sampling operation followed by an absolute value of a single arithmetic operation (Figure 2b).

Morphological Closing: This operation is neighborhood-based. It's a morphological operation and comprises a dilation followed by an erosion. Separated kernels are used. Three different kernel sizes were tested: 3, 14, and 90 (Figure 2c).

Oilpaint Abstraction: This multi-pass processing technique [33] consists mainly of color palette extraction, colorization, luminance quantization, edge detection, flow-field computation, smoothing and blending techniques (Figure 2d).

4.2 Overview of Test Systems

We evaluate the run-time performance of above processing techniques applied to the test input data using servers provided by different vendors. Table 2 (next page) shows an overview of the hardware and software specifications of the test systems:

Dedicated GPU Server: This server has dedicated graphics hardware and was used to measure the competitive performance of GPU-based processing as well as processing using SWR.

Amazon EC2 t2.large: This is a general purpose AWS EC2 instance of the type t2.large.

Amazon EC2 c4.4xlarge: This is a compute-optimized AWS EC2 instance of the type c4.4xlarge.

Amazon EC2 c5.18xlarge: This is highly compute-optimized AWS EC2 instance of the type c5.18xlarge. It has additional RAM and an increased number of virtual CPUs (vCPUs) compared to the c4.4xlarge instance.

4.3 Test Setup

In order to obtain the performance measurements, two docker containers were launched. As shown in Figure 3, one container runs the actual image processor instance, while the other container exposes a REST interface for communication with the client. It consists of a NodeJS [34] server that communicates with the image processor instance via WebSockets [29]. The image processor itself had a *profiler* implementation that could also be configured using the REST interface and performed measurements whenever an image was processed. Also the obtained performance results could be queried using this interface.

After the docker containers were launched, the profiler was configured to measure only the duration of the actual processing of an image – not the setup of the operation or the transmission of the image. In a real-world scenario the transmission and loading times are of course relevant and should be taken into consideration but these are not the focus of this work. The profiler uses OpenGL query objects to obtain the measurements of the actual processing time and not only the time spend to call the asynchronous OpenGL requests.

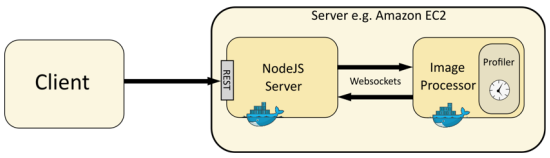


Figure 3: Setup and deployment for service-based performance testing. Via REST interface, a client communicates with the image-processing service that controls an image processing instance using WebSockets.

ISBN 978-80-86943-37-4

Table 2: Overview of cloud-computing provider used for performance evaluation (all 64 bit architecture).

Aspect	Dedicated GPU Server	AWS EC2 t2.large	AWS EC2 c4.4xlarge	AWS EC2 c5.18xlarge
Processor	Xeon E5-2637 v4, 3.5 GHz	Xeon, 3.0 GHz	Xeon E5-2666 v3, 2.9 GHz	Xeon Platinum, 3.0 GHz
Cores/Threads	8 cores	2 vCPUs	16 vCPUs	72 vCPUs
Memory	64 GB	8 GB	30 GB	144 GB
GPU	NVIDIA Quadro M6000 24 GB	None	None	None

When using software rendering, this obviously makes no difference because everything that would be processed by a GPU is processed by the CPU itself. For every combination of resolution and processing technique six identical image processing requests were sent and the results of the measurements queried and saved.

4.4 Test Results

From the six measurements obtained for every combination of image resolution and processing technique, the first one was ignored and the average of the remaining ones was calculated. This was done due to the fact that the first of these six measurements usually was significantly higher than the remaining (ranging from a factor of 1.2 up to a factor of over 1000) because OpenGL optimizes after the first run of processing an image with a specific processing technique. This, of course, is an important aspect that should be taken into account when deploying a service-based image processing service for multiple users that probably want to process images with different resolutions and different processing techniques.

Figure 4 shows the results of the measurements. The first four charts show the absolute processing time in microseconds for a specific resolution and processing technique comparing GPU-based rendering (solid lines) to SWR (dotted lines) on the different machines. All charts are shown using a logarithmic scale.

Figure 5 shows the *speed factor* for software rendering on each of the four test machines, i.e., how much slower software rendering was comparedto GPU-based rendering. The factor was calculated by dividing the duration of GPU-based processing by the duration using software rendering. In the case that SWR was faster, its duration of was divided by the duration of the GPU-based approach. The result then was negated to emphasize that software rendering was actually faster. Note that in case of the color invert, the values are negative, i.e., for this processing technique SWR was actually faster than GPU-based rendering.

4.5 Discussion of Test Results

At first, the processing duration of software rendering compared to GPU-based rendering is discussed. Subsequently, we discuss the computed speed factors.

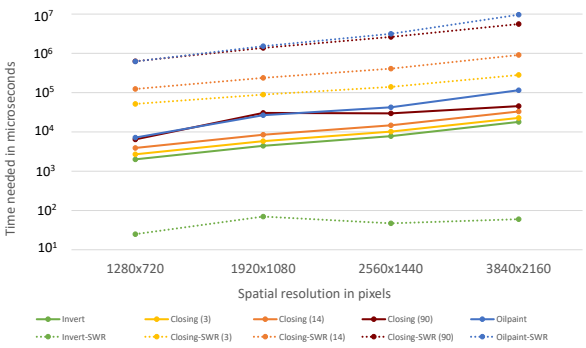
It can be observed that SWR generally is significantly slower than GPU-based rendering, but the relations between the different processing techniques and image resolutions for SWR are similar to GPU-based rendering. For example, for both SWR and GPU-based rendering, oilpaint abstraction, and morphological closing with kernel size 90 have a similar processing duration for smaller image resolutions, but for higher image resolutions the oilpaint abstraction exhibit high runtimes. This shows that SWR introduces an impact to the processing performance, but it does not change how the different processing techniques and image resolutions compare to each other regarding performance, e.g., which processing technique has the shortest duration for a specific resolution. While SWR is slower for most of the processing techniques, interestingly for color invert this is not the case. There are two possible explanations for this: first of all we only tested GPU-based processing using one specific GPU. It could be that on other GPUs color invert would be faster. Another explanation could be that transferring the image to the GPU takes longer compared to perform an invert on the CPU, even if the CPU cannot parallelize this similar to the GPU [15].

The measurements show that an approach for reducing the processing time when using SWR is to increase the number of vCPUs. Figure 6 shows the speed increase by increasing vCPUs based on the taken measurements for morphological closing with kernel size of 14 on a Full High Definition (FHD) image on the three Amazon EC2 instances. This speed factor is computed by dividing the processing duration using the EC2 t2.large instance (which has two vCPUs) by the processing duration if using additional vCPUs. It can be observed that the speed does not increase linearly with an increasing amount of vCPUs. For a final conclusion, more measurements on different machines would be required but there probably exists an upper bound to which the speed factor can be pushed.

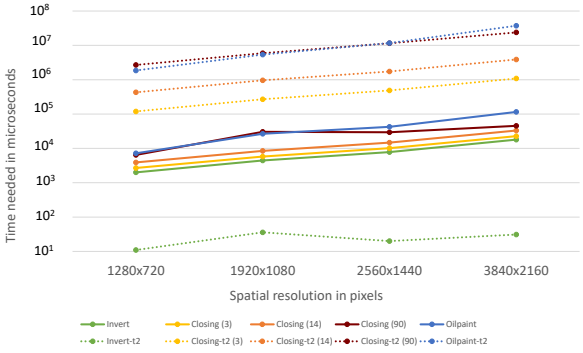
Regarding the speed factor for SWR compared to GPU-based rendering, it can be observed that for single or multi-pass processing techniques based on pixel-based or neighborhood-sampling of small kernel-sizes the factor remains stable. This allows for assessing the performance impact when using SWR for these processing techniques. However, for complex processing techniques it can be observed that with increasing image resolution, GPU-based rendering performance is superior compared to SWR.

Thus, for more complex processing techniques, the performance impact depends strongly on the spatial res-

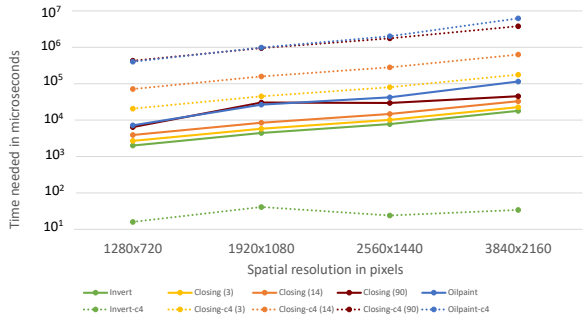
 ISBN 978-80-86943-37-4



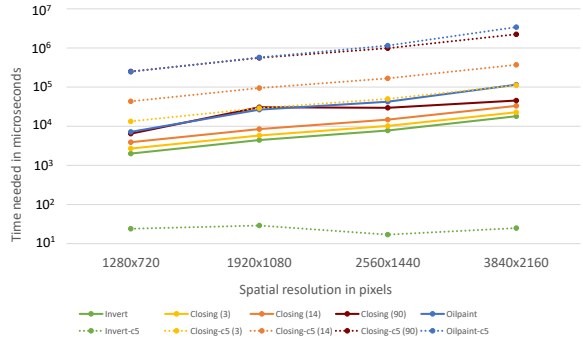
(a) GPU vs. SWR on dedicated server.



(b) GPU vs. AWS EC2 t2.large.

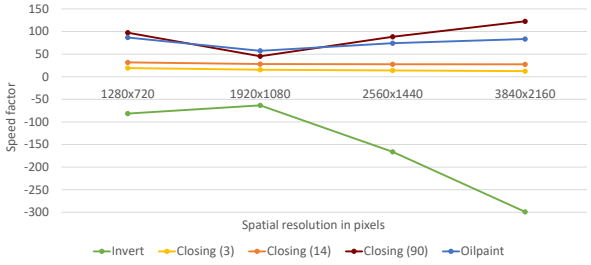


(c) GPU vs. AWS EC2 c4.4x large.

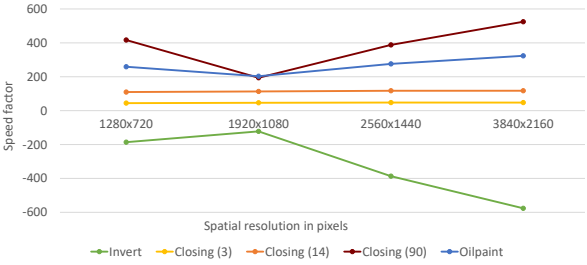


(d) GPU vs. AWS EC2 c5.18xlarge.

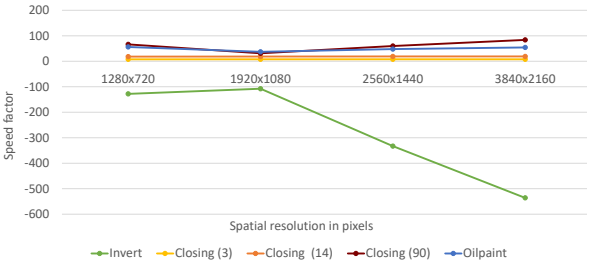
Figure 4: Comparing processing duration of GPU-based rendering to software rendering on different test systems.



(a) SWR on dedicated server.



(b) AWS EC2 t2.large.



(c) AWS EC2 c4.4xlarge.

(d) AWS EC2 c5.18xlarge.

Figure 5: Speed factor of software rendering compared to GPU-based rendering for different machines.

ISBN 978-80-86943-37-4

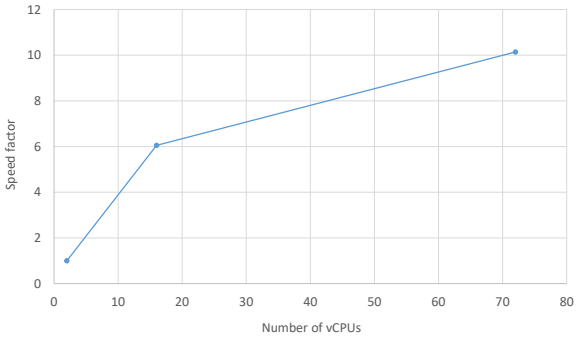


Figure 6: Speed factor for a specific amount of vCPUs compared to two vCPUs.

olution of the input image. It's interesting that for complex processing techniques (e.g., oilpaint abstraction and morphological closing with kernel size of 90), the speed factor for SWR on all test machines when processing a FHD image is noticeable lower, which is reflected by the bend in the graph. The origin of this bend lies in a higher processing duration for FHD images compared to Quad High Definition (QHD) images when using GPU-based rendering.

It should be mentioned that a FHD image has 2.25 times more pixels than a High Definition (HD) image and a 4K image also has 2.25 times more pixels than a QHD image, while a QHD image only has approx. 1.7 times more pixels than a FHD image. This means that regarding that number-of-pixels the measurements results for QHD images would be closer to the FHD images, compressing the graph in the horizontal direction. This of course does not explain why the processing duration for QHD is lower than the processing time for FHD. For this, we found no sufficient explanation. The reason may lie within the specific GPU we used for testing.

5 CONCLUSIONS

This paper presents the results of feasibility study and rendering performance evaluation of service-based image processing techniques based on OpenGL software rendering. It compares the rendering performance results with a dedicated GPU-accelerated service deployment. The results show a significant negative performance of software rendering compared to GPU-accelerated processing. However and to a certain extend, this limitations can be attenuated by increasing the number of vCPUs/Threads.

Further, the performance of software rendering compared to GPU-based rendering strongly depends on the implementation complexity of the processing technique, i.e., for less complex processing operations the speed factor remains stable. Furthermore, for less complex processing techniques using pixel-based sampling, software rendering can be faster.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable feedback. This work was funded by the Federal Ministry of Education and Research (BMBF), Germany, for the AVA project 01IS15041.

REFERENCES

[1] Carl Boettiger. An introduction to docker for reproducible research. *SIGOPS Oper. Syst. Rev.*, 49(1):71–79, January 2015.

[2] Aleksey Bragin, Alexander Dubanov, and Alexander Rechitskiy. User Interface Design for a Web-based Image Processing and Analysis System. In C. Wernhard S. Hölldobler, A. Malikov, editor, *YSIP2 - Proceedings of the Second Young Scientist's International Workshop on Trends in Information Processing*, Dombai, Russian Federation, 2017. CEUR.

[3] Tobias Dürschmid, Maximilian Söchting, Amir Semmo, Matthias Trapp, and Jürgen Döllner. Prosumerfx: Mobile design of image stylization components. In *SIGGRAPH Asia 2017 Mobile Graphics & Interactive Applications*, SA '17, pages 1:1–1:8, New York, NY, USA, 2017. ACM.

[4] Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[5] Romain Failliot, Tobias Droste, and Robin McCorkell. The OpenGL vs Mesa matrix. `https://mesamatrix.net`. Accessed: 2019-04-26.

[6] Zeev Farbman, Raanan Fattal, Dani Lischinski, and Richard Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics*, 27(3):67:1–67:10, 2008.

[7] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, Los Alamitos, 2016. IEEE Computer Society.

[8] Kengo Hayashi, Naohisa Sakamoto, Jorji Nonaka, Motohiko Mastuda, and Fumiyoshi Shoji. An In-Situ Visualization Approach for the K computer using Mesa 3D and KVS. In *ISC Workshop on In-Situ Visualization 2018 (WOIV 2018)*, 2018.

[9] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In *Proc. European Conference on Computer Vision (ECCV)*, pages 1–14. Springer, 2010.

[10] Alexander Jungmann and Bernd Kleinjohann. Automatic composition of service-based image processing applications. In *Proc. IEEE Interna-*

 ISBN 978-80-86943-37-4

tional Conference on Services Computing (SCC), pages 106–113. IEEE Computer Society, 2016.

[11] Michael Kass and Justin Solomon. Smoothed local histogram filters. *ACM Transactions on Graphics*, 29(4):100:1–100:10, 2010.

[12] David Koller, Michael Turitzin, Marc Levoy, Marco Tarini, Giuseppe Croccia, Paolo Cignoni, and Roberto Scopigno. Protected Interactive 3D Graphics via Remote Rendering. *ACM Trans. Graph.*, 23(3):695–703, August 2004.

[13] Jan Eric Kyprianidis, John Collomosse, Tinghuai Wang, and Tobias Isenberg. State of the "art": A taxonomy of artistic stylization techniques for images and video. *IEEE Transactions on Visualization and Computer Graphics*, 19(5):866–885, 2013.

[14] Samuli Laine and Tero Karras. High-performance software rasterization on gpus. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, HPG '11, pages 79–88, New York, NY, USA, 2011. ACM.

[15] Victor W. Lee, Changkyu Kim, Jatin Chhugani, Michael Deisher, Daehyun Kim, Anthony D. Nguyen, Nadathur Satish, Mikhail Smelyanskiy, Srinivas Chennupaty, Per Hammarlund, Ronak Singhal, and Pradeep Dubey. Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU. In *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ISCA '10, pages 451–460, New York, NY, USA, 2010. ACM.

[16] Angel Lagares Lemos, Florian Daniel, and Boualem Benatallah. Web service composition: A survey of techniques and tools. *ACM Computing Surveys*, 48(3):33:1–33:41, 2015.

[17] The LLVM Compiler Infrastructure. `http://llvm.org`. Accessed: 2019-04-26.

[18] Kwan-Liu Ma and Steven Parker. Massively Parallel Software Rendering for Visualizing Large-Scale Data Sets. *IEEE Computer Graphics and Applications*, 21(4):72–83, July 2001.

[19] The Mesa 3D Graphics Library. `https://www.mesa3d.org`. Accessed: 2019-04-26.

[20] Peter Mileff and Judit Dudra. Efficient 2D Software Rendering. *Production Systems and Information Engineering*, 6(2):55–66, May 2012.

[21] Peter Mileff and Judit Dudra. Modern Software Rendering. *Production Systems and Information Engineering*, 6(2):99–110, May 2013.

[22] Himadri Nath Moulick and Moumita Ghosh. Medical image processing using a service oriented architecture and distributed environment. *American Journal of Engineering Research (AJER)*, 2(10):52–62, 2013.

[23] Matthias Mueller and Benjamin Pross. *OGC WPS 2.0.2 Interface Standard*. Open Geospatial Consortium, 2015. http://docs.opengeospatial.org/is/14-065/14-065.html.

[24] Christopher Niederauer, Mike Houston, Maneesh Agrawala, and Greg Humphreys. Non-invasive interactive visualization of dynamic architectural environments. In *Proceedings of the 2003 Symposium on Interactive 3D Graphics*, I3D '03, pages 55–58, New York, NY, USA, 2003. ACM.

[25] Mike P. Papazoglou and Willem-Jan Heuvel. Service oriented architectures: Approaches, technologies and research issues. *The VLDB Journal*, 16(3):389–415, 2007.

[26] Chris Peltz. Web services orchestration and choreography. *Computer*, 36(10):46–52, October 2003.

[27] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[28] Ricardo Queirós and Alberto Simões. SOS - Simple Orchestration of Services. In Ricardo Queirós, Mário Pinto, Alberto Simões, José Paulo Leal, and Maria João Varanda, editors, *6th Symposium on Languages, Applications and Technologies (SLATE 2017)*, volume 56 of *OpenAccess Series in Informatics (OASIcs)*, pages 13:1–13:8, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[29] Marvin Richter, Maximilian Söchting, Amir Semmo, Jürgen Döllner, and Matthias Trapp. Service-based Processing and Provisioning of Image-Abstraction Techniques. In *Proceedings of the 26th International Conference on Computer Graphics, Visualization and Computer Vision*, Proceedings International Conference on Computer Graphics, Visualization and Computer Vision (WSCG), pages 97–106, 2018.

[30] Timothy Rowley. Software Rasterizer (SWR). In *Presented at the Intel HPC Developers Conference at SC14*, 2014.

[31] Mark Segal and Kurt Akeley. *The OpenGL Graphics System: A Specification (Version 4.6 (Core Profile) - May 14, 2018)*. The Khronos Group Inc., May 2018.

[32] Mark Segal and Kurt Akeley. *Vulkan 1.0.107: A Specification*. The Khronos Group Inc., April 2019.

[33] Amir Semmo, Daniel Limberger, Jan Eric Kyprianidis, and Jürgen Döllner. Image stylization by interactive oil paint filtering. *Computers & Graphics*, 55:157–171, 2016.

 ISBN 978-80-86943-37-4

[34] Lambert M. Surhone, Mariam T. Tennoe, and Susan F. Henssonow. *Node.Js*. Betascript Publishing, Mauritius, 2010.

[35] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Proc. International Conference on Computer Vision (ICCV)*, pages 839–846. IEEE, 1998.

[36] Mircea-Florin Vaida, Valeriu Todica, and Marcel Cremene. Service oriented architecture for medical image processing. *International Journal of Computer Assisted Radiology and Surgery*, 3(3):363–369, 2008.

[37] Joachim Weickert. *Anisotropic diffusion in image processing*, volume 1. Teubner Stuttgart, 1998.

[38] Robert P. Winkler and Chris Schlesiger. Image processing rest web services. Technical Report ARL-TR-6393, Army Research Laboraty, Adelphi, MD 20783-119, 2013.

[39] Holger Winnemöller, Sven C. Olsen, and Bruce Gooch. Real-time video abstraction. *ACM Transactions on Graphics*, 25(3):1221–1226, 2006.

[40] M. Würsch, R. Ingold, and M. Liwicki. Sdk reinvented: Document image analysis methods as restful web services. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pages 90–95, 2016.

[41] Marcel Würsch, Rolf Ingold, and Marcus Liwicki. Divaservices - a restful web service for document image analysis methods. *Digital Scholarship in the Humanities*, 32(1):i150–i156, 2017.

[42] Xiaoxia Yang. Remotely sensed image processing service automatic composition. State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, 2009.

[43] Rong Yuan, Ming Luo, Zhi Sun, Shuyue Shi, Peng Xiao, and Qingguo Xie. Rayplus: a web-based platform for medical image processing. *J. Digital Imaging*, 30(2):197–203, 2017.

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

Automated Object Tracking in Sterile Pharmacy Compounding

Zhe Qi
Dept. of Computer Science
221 Hayes Hall
Bowling Green State Univ.
Bowling Green, OH 43403,
USA
qizhe@bgsu.edu

Jong Kwan Lee
Dept. of Computer Science
221 Hayes Hall
Bowling Green State Univ.
Bowling Green, OH 43403,
USA
leej@bgsu.edu

ABSTRACT

A new method to automatically track objects in the process of sterile pharmacy compounding is introduced. The method performs the automatic object tracking by applying a simple object detection followed by an object tracking algorithm. The advantages of the method are that it enables both single and multiple object tracking without requiring large sets of templates or training sets. The experimental results show that the method reasonably tracks the objects accurately.

Keywords

Object Detection/Tracking, Image Processing, Pharmacy Compounding Application

1 INTRODUCTION

Object detection and object tracking have been utilized for many application domains (e.g., [**mgi04**, **bps05**, **azl11**]). They play a very important role for the various applications since identifying and tracing the feature(s) of interest are the primary steps for further analysis or new findings. The target application of this paper is the sterile pharmacy compounding.

A majority of medications administered in hospitals, long-term care facilities, patient center medical homes, and emergency medicine are given patients as an injection. Such compounding sterile products are predominantly prepared by mixing, compounding, and manipulating pre-manufactured products in an intravenous compounding room in the hospital or a special compounding center. (Figure 1 shows an example of sterile compounding workspace in a clean room. There is very minimal environmental disruption.) Errors and omissions in this process occur often, leading to severe physical implications for patients (some can be fatal).

Continual monitoring, evaluation, and correction during the compounded sterile product preparation relies on direct (or indirect) human supervision that is provided by an outside participant or a sole practitioner. There are many pharmacy compounding rules [**gjc13**] to follow, including objects that put on the compounding table should not be close to the table edge; there is a minimum distance between two objects on the table; etc. However, the human factors often result in not following some of the rules and/or lead to errors when compounding products.

Figure 1: An Example of Sterile Compounding Workspace (image credits given to M. Soper [**sop18**]).

In this paper, we introduce a method for tracking objects in sterile pharmacy compounding process. Such a method can assist reducing human errors–it can support the observation of the physician's operations in the real hospital setting as well in the training setting for education. For example, an effective and efficient automated tracking method can signal warnings when human errors occur. Figure 2 shows the simulated sterile compounding workspace setting used in our application.

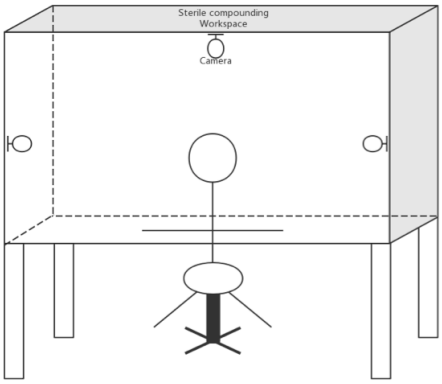


Figure 2: Sterile Compounding Workspace Setting with Cameras.

(While multiple cameras can be used, only one camera was used for this paper.) The video frames taken from the camera can be processed by our method in real time.

The paper is organized as follows. In Section 2, the related work is discussed. The new automated object tracking method for assisting sterile pharmacy compounding is described in Section 3. In Section 4, the experimental results and analysis are presented. Section 5 concludes this paper.

2 RELATED WORK

Many methods for object detection or object tracking in other application domains have been reported in the literature. In this section, some of the recent work are discussed briefly. Here, we note that we intentionally discuss a wide range of literature since our application covers both object detection and object tracking.

Object tracking is often used in intelligent video surveillance and motion recognition in human-computer interaction. One example is the traffic monitoring. Typically traffic monitor employs object tracking methods to identify the traffic-causing vehicle (e.g., accidents) or may predict traffic based on where the vehicles are moving on the road. For example, Rouhani et al. [**rmk17**] have introduced an automated re-configurable framework for real-time intelligent video surveillance. Their framework enables efficient and accurate object tracking by utilizing image processing, principle component analysis, and adaptive learning dictionary matrix for foreground estimation in their intelligence transportation system. Their framework allows efficient processing by employing a field-programmable gate array (FPGA) with accompanying APIs which allow efficient processing.

Panchal et al. [**ppp15**] have reviewed several object detection and tracking methods, including a background subtraction method, real-time background subtraction and shadow detection technique theory, and template matching and shape-based method. Their review included varying environmental settings that can happen in different object recognition and compared the methods. The summary of their review included that (1) the background subtraction was widely used, simple to implement, provided fast recovery with low memory requirement, allowed objects become a part of the background without altering the existing background, but it was not good when shadow and many other objects were present, could not handle very quick objects changes well; (2) template matching was the best method for a specific environment, such as a background invariant environment, but it only supported one-to-one match and worked only if the object appeared in all video frames, it had slow process for recognize new variation of a pattern; (3) frame differencing performed well for static background, it was very easy and gives high accuracy, but it required a background without moving objects.

Recently, Held et al. [**hts16**] introduced a method called GOTURN, which enabled very efficient object tracking using a machine learning-based scheme. Their scheme trained the GOTURN tracker entirely offline with video sequences and images (i.e., no online training was required). It also utilized a local genetic object detector to find target objects and performed frame-by-frame comparisons for tracking. Their method did not include any object labeling and need object type information, which were typically need for other tracker algorithms.

A popular feature detection domain is fingerprint identification and face recognition (e.g., [**olp03**, **kun04**]). The research in this domain has been very active in many decades, but still is one of the important applications. For example, these days, fingerprint-enabled door lock and smartphone fingerprint identification use photoelectric conversion equipment and image processing technique to collect, analyze and compare fingerprints in order to authenticate the users. For example, Kundargi et al. [**kuk18**] recently introduced an automated fingerprint recognition system that overcomes the problem of fake verification caused by an artificial fingerprint from synthetic finger fabricates (e.g., silicone or latex). Their system included a specialized hardware module and a specialized software module for liveness detection, employing a texture feature analysis for verification.

For the work briefly discussed above, most of them utilized a machine learning-based processing. Thus, they needed a large amount of datasets for training. The new automated method introduced in this paper performs object detection and object tracking without a large dataset training process. To our knowledge, no automated method for assisting sterile pharmacy compounding has been presented yet in the literature.

 ISBN 978-80-86943-37-4

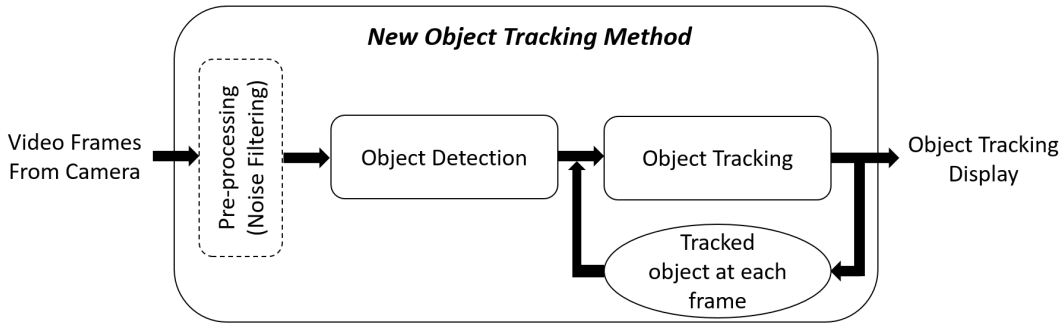
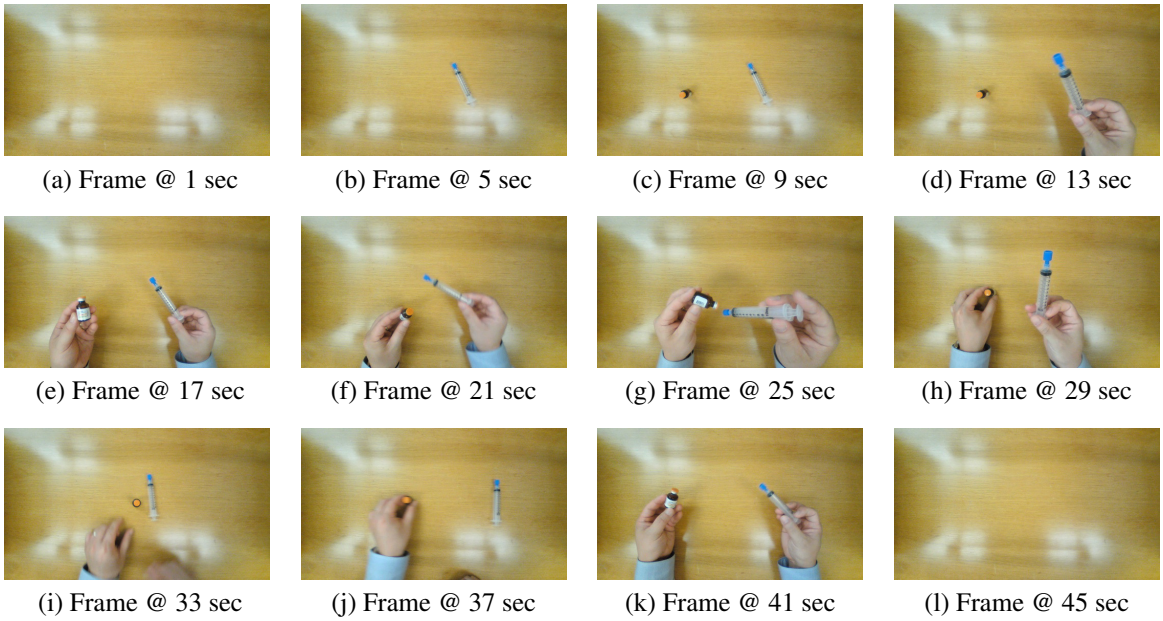


Figure 3: New Automated Tracking Method: the Overview.



(a) Frame @ 1 sec (b) Frame @ 5 sec (c) Frame @ 9 sec (d) Frame @ 13 sec

(e) Frame @ 17 sec (f) Frame @ 21 sec (g) Frame @ 25 sec (h) Frame @ 29 sec

(i) Frame @ 33 sec (j) Frame @ 37 sec (k) Frame @ 41 sec (l) Frame @ 45 sec

Figure 4: An Example of Simulated Sterile Compounding Video Frames.

3 NEW AUTOMATED METHOD

Next, the new automated tracking method for assisting sterile pharmacy compounding is described.

The overview of the new method is shown in Figure 3. The image frames of video (i.e., the table scene of sterile compounding) taken from the camera are continuously fed into the tracking steps. The new tracking steps include (1) a pre-processing step to “clean-up" the image frame, (2) the object detection step to initialize the location of objects, and (3) the object tracking step to keep tracking the location of objects.

In our application, two consecutive image frames are considered at each iteration of the steps and the latter frame is re-considered in the next iteration. Figure 4 shows an example of simulated compounding video frames captured at every 4 seconds. (Before subfigures (b) and (c), each object was put on the table by a physician.) As shown in the figures, there are also other image effects (e.g., lights, reflection, shadows) in the video frames. Here, we note that all videos used in our paper assumed that the table was empty at the beginning (which will be the case in the real setting). Thus, the very first frame is used as the background frame.

3.1 Pre-processing Step

The primary goal of the pre-processing step is to clean up image noise. In our method, a Gaussian filter [**gow02**] is used to de-noise video frames. (We assumed that there would not be complex image noise in the real sterile compounding video frames since it will be taken in a clean room setting. However, we assumed that there would be some normal distributed random noise from video camera sensor.)

3.2 Object Detection Step

After the video frames are cleaned, the object detection step is applied to initialize the locations of objects of

ISBN 978-80-86943-37-4

interest. In particular, frame differencing, thresholding, and a morphological closing are applied in the step.

The very first frame in each video is used as the background frame. We denote the background frame as the *B frame*. If a video frame contains any object, the *B* frame can be referenced and compared to find the image regions where the object appears. For example, when a brown bottle appears on frame *N*, the image regions where the bottle are can be determined by examining the background subtracted image (i.e., $|N-B|$ image and we denote the background subtracted image as the *S* image). When iterating the object detection step with *N* and $N+1$ frames, the frame differencing of these consecutive frames gives a local signal of where the objects were moved. We denote the result of this frame differening as the *D image*. In our method, we used a weighted sum of the *S* image and the *D* image to enhance the contrast of the object boundary appeared on the *N* frame.

To locate the object boundary, a simple thresholding is applied (because the image regions with objects will have high value in the object contrast enhanced image) and a contour extraction is applied on the thresholded image. Here, we note that while the physician's hand and arm will also be extracted, we can discard them by removing the larger object contour because they appear as larger contours.

In addition, since light effects (e.g., lights, reflection, shadows) and object occlusion can result in incomplete object contours, a morphological closing is applied to connect the disconnected object regions.

At the end of the detection step, the bounding boxes of the objects are determined by considering the maximum and minimum coordinates of the object contours. Here we note that, the parameters used in this step were determined empirically (e.g., threshold value of 15 and morphological structure of size 5 by 5 were used).

3.3 Object Tracking Step

The last step of the new object tracking method is the object tracking step. In this step, the frame-by-frame objects are being tracked by utilizing a combination of a local template matching and a tracker algorithm.

The cross-correlational template matching [**bru09**] is employed to track the object bounding box on the objects. In particular, the bounding box from the previous frame is used as the template for the current frame. In addition, a tracker algorithm is applied with the template matching for more accurate object tracking by exploring the scale-invariant properties of the tracker algorithm. (For example, the objects in the video often move up and down, resulting in different scales.) Our method utilized two different tracker algorithms: the median flow tracker [**kmm13**] and the discriminative correlation filter-based tracker [**dhf14**]. (We have chosen these two tracker algorithms since they are used widely and are available on the OpenCV library.) From empirical testings, we have found that the medial flow tracker produced a better result for a single object tracking and the discriminative correlation filter-based tracker produced a better result for a multi-object tracking. In our method, the combination of template matching and tracker algorithm is applied locally (i.e., applied to nearby image regions of the object bounding boxes) since the objects do not typically move very quickly (i.e., there is no abrupt positional change).

The object tracking step is repeatedly applied frame-by-frame to track the objects for the entire video frames.

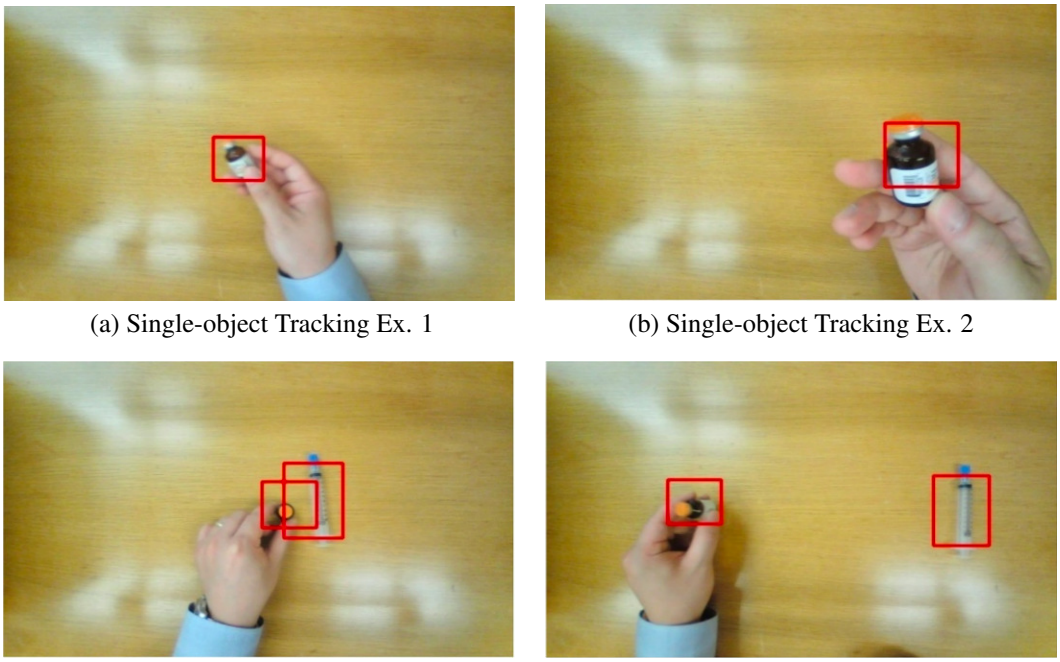
4 EXPERIMENTAL RESULTS

The effectiveness of the our object tracking method has been benchmarked using the real videos in the simulated workspace setting. Both single and multi-object videos are considered. Each video frame was about 1,920 pixels in width and 1,080 pixels in height and each video included about 1,000 to 1,500 frames. The benchmarking included measuring the method's effectiveness by considering the accuracy and the processing time. In our work, we define the accuracy of the object tracking as the percentage ratio of the number of frames that had at least 75% of the objects detected/tracked in each frame and the number of frames that the objects appeared. We used the average accuracy of all object accuracies for the multi-object tracking benchmarkings. We have compared our method's effectiveness with the effectiveness of the cross-correlational template matching [**bru09**]. All experiments were performed on a PC with Intel Core i7 2.7GHz CPU with a 8GB RAM.

Figure 5 shows the examples of object tracking using our method. Figure 5 (a) and (b) are for a single-object tracking (with a brown bottle of sterile compounding product) and (c) and (d) are for a multi-object tracking (with the brown bottle and a needle tube). The single-object and multi-object tracking videos had 904 frames and 1,368 frames, respectively. The red rectangle overlays in each sub-figures are the markers for object tracking. As shown in the figure, even though there were different light effects, the new method detected and tracked the objects accurately- the single-object and multi-object tracking shown in Figure 5 had about 84% and 80% accuracies, respectively. (Additional discussion for improvement is in Section 5.)

While the new tracking method had the tracker makers on all objects for all frames the objected appeared, there were some cases it did not accurately marked the objects. As defined above, we consider the tracking for a frame is a miss if the method didn't mark at least 75% of the object in each frame. Figure 6 shows two examples where the tracking miss occurred. In Figure 6 (a), while the brown bottle is marked, only about 72%

 ISBN 978-80-86943-37-4



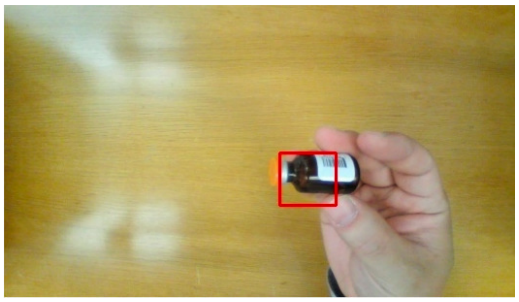
(a) Single-object Tracking Ex. 1

(b) Single-object Tracking Ex. 2

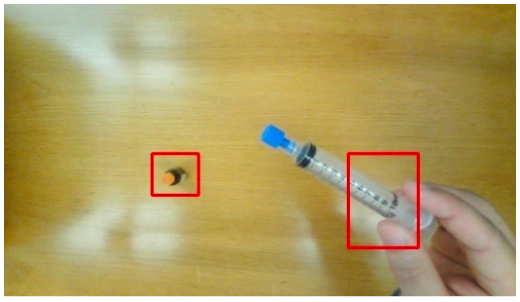
(c) Multi-object Tracking Ex. 1

(d) Multi-object Tracking Ex. 2

Figure 5: Automated Object Tracking Results: Tracking Frames.



(a) Miss in Single-object Tracking



(b) Miss in Multi-object Tracking

Figure 6: Examples of Missing Objects.

of the bottle was included in the tracking marker. In Figure 6 (b), while the brown bottle is marked well, the new method marked only parts of the needle tubes.

For comparison of methods, the cross-correlational template matching had about 66.51% accuracy and 0.0376 seconds for processing time on average and the new tracking method had about 81.66% accuracy and 0.0415 seconds for processing time on average. While the template matching method was slight more efficient than our method, our method outperformed the accuracy.

5 CONCLUSION

We have presented and evaluated a new object tracking method that can be used in the sterile pharmacy compounding application. The method involves use of a simple image processing, object detection, and object tracking algorithms that enable effective object tracking. One of the primary advantages of the new method is that it does not require a large number templates nor training datasets. The new tracking method produced a very reasonable tracking results and outperformed the the cross-correlational template matching.

For the future work, other image processing steps to identify the objects more accurately may be explored (e.g., a better object contour extraction step, more effective step to remove hand and arm, etc). We also plan to explore additional step to more accurately mark the tracked objects. For example, Hough Transform (HT) and its variation [**hou62**, **hoc96**] may be considered in our application by applying a variant in a local fashion way. (Figure 7 shows our ongoing effort on local HT-based tracking makers as an example. As shown in the figure, the markers are very accurately placed on objects.) Fourier descriptors [**gow02**] may also be

 ISBN 978-80-86943-37-4

(a) Object 1 (b) Object 2

Figure 7: Accurate Object Labeling with HT.

considered. For efficient processing of the method, we may also explore GPU processing (e.g., [**sim03**, **urg07**, **deg10**]).

We also plans to compare our method with other techniques and methods used for object tracking. (In our preliminary results of comparing with other machine learning-based algorithms, other algorithms need a large number of training datasets to have a reasonable accuracy.)

6 REFERENCES

[AZL11] Aiping W., Zhi-Quan C., and Li M., Multiple-Cue-Based Visual Object Contour Tracking with Incremental Learning, in Proc., 2011 Int'l Conference on Virtual Reality and Visualization, 2011, pp. 30–37.

[BPS05] Bleser G., Pastamov Y., and Stricker D., Real-time 3D Camera Tracking for Industrial Augmented Reality Applications, in Proc., The 13th Int'l Conference in Central Europe on Compute Graphics, Visualization and Computer Vision (WSCG 2005), Jan. 31–Feb. 4, 2005, pp. 47–54.

[Bru09] Brunelli R., Template Matching Techniques in Computer Vision: Theory and Practice, 1st Ed., Wiley, April 2009.

[Deg10] Degirmenci, M., Complex Geometric Primitive Extraction on Graphics Processing Unit, Journal of WSCG, Vol. 18, 2010, pp. 129–134.

[DHF14] Danelljan M., Hager G., Khan F., and Felsberg M., Accurate Scale Estimation for Robust Visual Tracking, in Proc., British Machine Vision Conf., Nottingham, Sep. 2014, pp. 65.1–65.11.

[GJC13] Gudeman J., Jozwiakowski M., Chollet J., and Randell M., Potential Risks of Pharmacy Compounding, Drugs in R&D, Vol. 13 (1), March 2013, pp. 1–8.

[GoW02] Gonzalez, R. and Woods, R., Digital Image Processing, 2nd Ed., Prentice Hall. 2002.

[HTS16] Held D., Thrun S., and Savarese S., Learning to Track at 100 fps with Deep Regression Networks, in Proc., European Conf. on Comp. Vision, Springer, Cham, Oct. 8, 2016, pp. 749–765.

[HoC96] Ho, C. and Chen, L., A High-speed Algorithm for Elliptical Object Detection, IEEE Transaction on Image Processing, Vol. 5 (3), 1996, pp. 547–550.

[Hou62] Hough, P., Method and Means for Recognizing Complex Patterns, U.S. Patent, 3,069,654, 1962.

[KMM10] Kalal Z., Mikolajczyk K., and Matas J., Forward-backward Error: Automatic Detection of Tracking Failures, in Proc., IEEE 20th International Conference on Pattern Recognition (ICPR), Aug. 2010, pp. 2756–2759.

[KuK18] Kundargi J. and Karandikar R.G., Integrating Liveness Detection Technique into Fingerprint Recognition System: A Review of Various Methodologies Based on Texture Features, Progress in Intelligent Computing Techniques: Theory, Practice, and Appls., 2018, pp. 295–305.

[KuN04] Kukharev G. and Nowosielski A., Visitor Identification - Elaborating Real Time Face Recognition System, in Proc., The 12th Int'l Conference in Central Europe on Compute Graphics, Visualization and Computer Vision (WSCG 2004), Feb. 2–6, 2004, pp. 157–164.

[MGI04] Misu T., Gohshi S., Izumi Y., Fujita Y., and Naemura M., Robust Tracking of Athletes Using Multiple Features of Multiple Views, Journal of WSCG, Vol. 12, 2004, pp. 285–292.

[OLP03] Oh S.-K., Lee J.-J., Park C.H., Kim B.-S., and Park K.-H., New Fingerprint Image Enhancement Using Directional Filter Bank, Journal of WSCG, Vol. 11 (1), 2003, pp. 1–8.

[PPP15] Panchal P., Prajapati G., Patel S., Shah H., and Nasriwala J., A Review on Object Detection and Tracking Methods, International Journal for Research in Emerging Science and Technology, Vol. 2 (1), Jan. 2015, pp. 7–12.

[RMK17] Rouhani B.D., Mirhoseini A., and Koushanfar F., Rise: An Automated Framework for Real-time Intelligent Video Surveillance on FPGA, ACM Trans. on Embedded Computing Systems (TECS), Vol. 16 (5s), Sep. 2017, Article No. 158.

[SIM03] Strzodka, R., Ihrke I., and Magnor, M., A Graphics Hardware Implementation of the Generalized Hough for Fast Object Recognition, Scale, and 3D Pose Detection, 12th Int'l Conf. on Image Analysis and Processing,2003, pp. 188–193.

[Sop18] Soper M., Why Should Compounding Pharmacies Continuously Monitor Particles?, www.setra.com/blog/why-should-compounding-pharmacies-continuously-monitor-particles, May 03, 2018, accessed on Jan. 30, 2019.

[URG07] Ujaldon, M., Ruiz, A., and Guil, N., On the Computation of the Circle Hough Transform by a GPU Rasterizer, Pattern Recog. Letters, Vol 29, (3), 2007, pp. 309–318.

 ISBN 978-80-86943-37-4

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

Compressed Exposure Sequences for HDR Imaging

Selin Sekmen
ASELSAN A.Ş.
Department of Computer Engineering
Middle East Technical University
06800, Ankara, TURKEY
ssekmen@aselsan.com.tr

Ahmet Oğuz Akyüz
Department of Computer Engineering
Middle East Technical University
06800, Ankara, TURKEY
akyuz@ceng.metu.edu.tr

ABSTRACT

High dynamic range (HDR) imaging techniques allow photographers to capture the luminance distribution in the real-world as it is, freeing them from the limitations of capture and display devices. One common approach for creating HDR images is the multiple exposures technique (MET). This technique is preferred by many photographers as multiple exposures can be captured with off-the-shelf digital cameras and later combined into an HDR image. In this study, we propose a storage scheme in order to simplify the maintenance and usability of such sequences. In our scheme, multiple exposures are stored inside a single JPEG file with the main image representing a user-selected reference exposure. Other exposures are not directly stored, but rather their differences with each other and the reference is stored in a compressed manner in the metadata section of the same file. This allows a significant reduction in file size without impacting quality. If necessary the original exposures can be reconstructed from this single JPEG file, which in turn can be used in a standard HDR workflow.

Keywords

HDR imaging, multiple exposures, JPEG, image metadata

1 INTRODUCTION

HDR imaging has been a rapidly emerging research field now finding applications in consumer photography and electronics [CNEa]. There are many approaches to create HDR images and videos including those involving dedicated HDR capture hardware [AlllC16a]. However, such devices are not only rare, but are also not preferred by most users due to their high cost. This typically leaves the photographers with the option of creating HDR images from multiple exposures. In this technique, a bracketed sequence of exposures are captured with a standard digital camera, with each exposure taken with a different exposure time. In such a sequence, short exposures contain detail in low-luminance regions whereas long exposures contain detail in high-luminance regions. By merging them into a single HDR image, the entire luminance range of the scene can be represented [Rei10a].

Arguably, the most problematic approach of this technique is the requirement of storing multiple images (typically 3 to 9) for each captured scene, together with the HDR image itself. While the individual exposures can be purged after the HDR image is created, keeping them around may be preferable for several reasons. First, as typical display devices are low dynamic range (LDR), storing these LDR images allow one to rapidly view the captured scene, without resorting to tone mapping [Rei07a]. Second, the HDR image may often contain ghosting artifacts due to camera and object motion during the capture process. As new and improved algorithms are proposed continually, keeping the individual images allow one to create higher quality HDR images using these improved algorithms.

However, storing a large number of images for each scene (i.e. individual exposures, the HDR image, tone mapped image) is problematic for different reasons. It not only incurs extra storage costs but also makes photo-album maintenance a tedious task due to image repetition. These storage and maintenance problems may even discourage photographers from taking multiple exposures and shy away from HDR photography.

In this paper, we propose a method known as compressed exposure sequences (CES), which allows embedding the entirety of the information in a bracketed sequence into a single JPEG file. The main image, which will be shown by a standard image viewer, is typically selected as the middle exposure although another exposure or the tone mapped HDR image can be used as well. Differences between the subsequent exposures are stored in a compressed manner in the metadata sec-

tion of the JPEG file. As demonstrated by several experiments, this scheme not only yields a smaller total file size but significantly simplifies the maintenance of exposure sequences while having a minimal impact on image quality.

2 RELATED WORK

There are many approaches to compress HDR images. JPEG-2000 [Sko01a] and JPEG-XR [Duf09a] have been developed to overcome the bit-depth limitation of the original JPEG format. Although the new standards offer superior compression performance to the original JPEG, they failed to achieve the desired level of adoption arguably due to their lack of backward compatibility with the original JPEG format.

All JPEG standards provide additional marker segments to keep an application specific data inside metadata sections [Pen92a, Wal92a]. This ability is used to create backward compatible HDR compression methods for images. JPEG-HDR is an extension to the standard JPEG format to store HDR images in a backward compatible manner [War06a]. The same ideas are extended to the video domain to allow for backward compatible HDR video storage [Man06a]. Beside these, there are dedicated HDR file formats such as RGBE, LogLuv, and OpenEXR [War94a, Lar98a, Kai02a]. However none of these file formats are backward compatible with existing software.

The most recent JPEG format, known as JPEG-XT, also uses JPEG application markers to store HDR images. JPEG-XT is backward-compatible with widely know JPEG format and provides lossy and lossless compression for high dynamic range images [Art15a].

The method proposed here can be seen as an extension of these backward-compatible storage schemes for HDR imaging. But unlike the existing methods that store a tone mapped image as the main image and auxiliary data for reconstructing the HDR image, we propose to store the entirety of information present in an exposure sequence. This simplifies the maintenance and usability of bracketed exposure sequences. It also allows for creating an HDR image at any time in future, using the original sequence stored in an efficient manner.

3 THE PROPOSED METHOD

The proposed method is termed as compressed exposure sequences (CES) as it aims to store multiple exposures inside a single JPEG file where the main image represents a user-selected reference exposure. CES is comprised of two main pipelines, one for compression and the other for decompression. In the compression pipeline, we use the multiple exposures of a scene and create a new JPEG image that stores all of the necessary data to recreate all of the original exposures. The decompression pipeline reads the main image from the JPEG file together with its metadata and reconstructs the original exposures. These pipelines are illustrated in Figures 1 and 2 with the details elaborated below.

Compression Pipeline

The compression pipeline is demonstrated in Figure 1. Initially, all exposures are loaded and sorted according to their exposure times. Thus, we get the image set $\langle I_1, I_2, ..., I_m, ..., I_n \rangle$ from the shortest to the longest exposure, where I_m refers to the middle exposure. All of these exposures are comprised of 24-bits per pixel (i.e., 8-bits per color channel). We also extract the exposure times from the EXIF data during exposure loading, and label them as $\langle E_1, E_2, ..., E_m, ..., E_n \rangle$.

During the compression pipeline, the main aim is to store the minimum necessary data for each exposure inside the JPEG application markers, so that we can recreate each exposure using this auxiliary information. To save space, exposures are not directly stored inside the metadata, instead for each exposure the difference with respect to a reference image is stored. Therefore, at the initial state of the pipeline, we choose a reference image for each exposure according to Equation 1:

$$I_{ref_x} = \begin{cases} I_{x+1} & \text{if } x < mid, \\ I_{x-1} & \text{if } x > mid. \end{cases} \tag{1}$$

In other words, the reference exposure for a given exposure is an adjacent exposure in terms of exposure time. Choosing such adjacent exposures as reference maximizes the coherency between them.

The next stage in the pipeline is to align each exposure with its reference to compensate for the camera movement during the image capture process. The median threshold bitmap (MTB) algorithm [War03a] was used to determine misalignments. This algorithm was selected due to its simplicity, efficiency, and robustness against exposure differences. After the alignment process, we apply the inverse camera response function (iCRF) on aligned nonlinear exposures to linearize them (Equation 2). The resulting linear images are symbolized as L_x. Without loss of generality, we used the radiometric self calibration approach [Mit99a] to recover the camera response function. At this point we move from the integer domain to the floating point domain:

$$L_x = 255 f^{-1}\left(\frac{I_x}{255}\right). \tag{2}$$

The difference between an exposure and its reference is computed as defined in Equation 3:

$$L_{dif_x} = \left| L_x - k_x L_{ref_x} \right|. \tag{3}$$

ISBN 978-80-86943-37-4

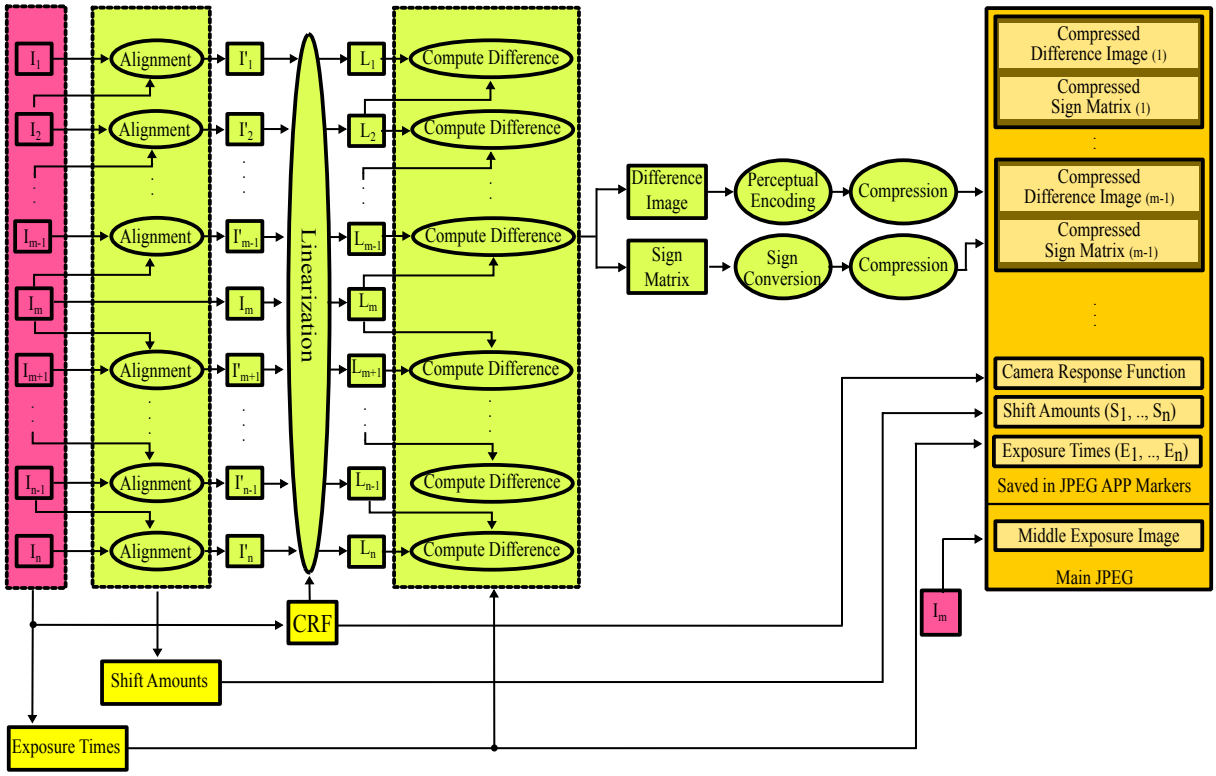


Figure 1: The compression pipeline of the CES algorithm.

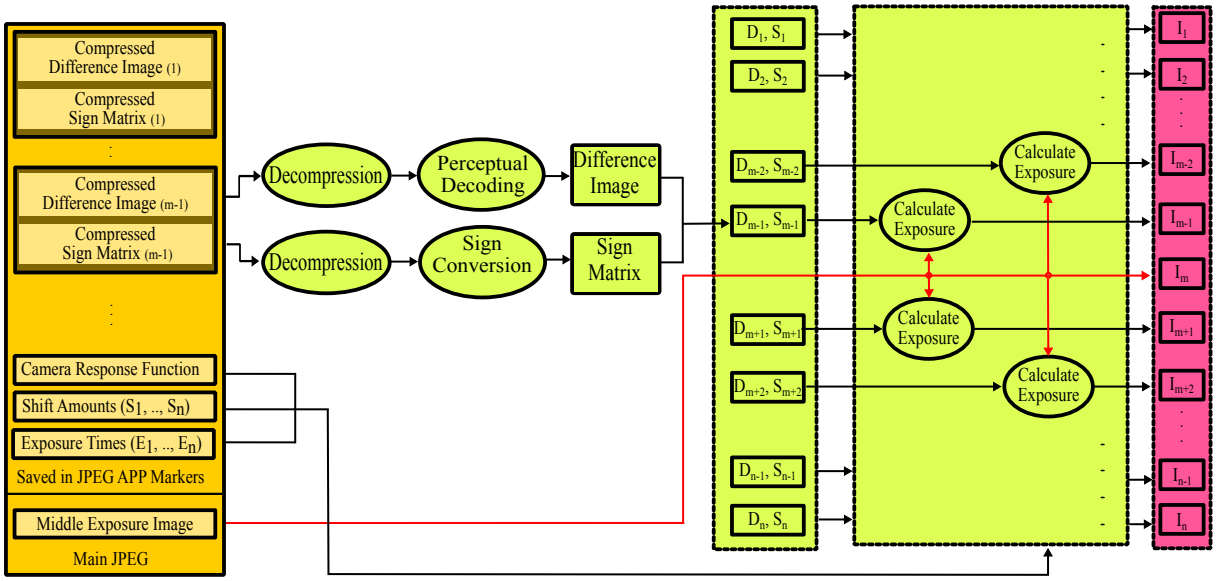


Figure 2: The decompression pipeline of the CES algorithm.

ISBN 978-80-86943-37-4

Before computation of this difference, we find the best exposure ratio k_x that minimizes the mean value of this difference using the gradient descent algorithm:[1]

$$k_x = \arg\min_k \left| L_x - kL_{ref_x} \right|. \quad (4)$$

This is done to bring the difference values as close as possible to zero to obtain higher compression rates.

The sign matrix stores the sign of the difference matrix for each pixel. The values for the sign matrix are selected as either 0 or 10, with the former indicating a positive difference and the latter a negative one. The reason for selecting these values instead of 0 and 1 is that if we apply lossy compression for the sign matrices, we can interpret values less than 5 as positive and the others as negative. This provides robustness against the lossy compression artifacts:

$$L_{sign_x} = \begin{cases} 10 & \text{if } L_x - k_x L_{ref_x} < 0 \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

As explained above, the result of the computed differences yield two data set for each exposure, namely the difference and sign matrices. Before applying compression to difference matrices, we apply gamma correction to them to simulate a simple form of perceptual encoding as shown in Equation 6. This provides a larger pixel value range for dark pixels for which the quantization artifacts would be more noticeable. In this step, we move back to the integer domain which enables better compression at the cost of introducing quantization errors:

$$P_{dif_x} = 255 \left(\frac{L_{dif_x}}{255} \right)^{1.0/2.2} \quad (6)$$

After perceptual encoding, we apply compression to the difference and sign matrices to decrease the data storage size. Our method allows for both lossless and lossy compression. Lossless compression is done using the ZLIB's Huffman coding based compression algorithm [Deu96a], whereas for lossy compression we use JPEG compression. The results of compressions are saved inside the JPEG metadata sections. In addition to the difference and sign matrices, we also store the exposure time ratios, alignment data, and the camera response function coefficients inside the JPEG application markers as well. The middle exposure, I_m, is stored as the main JPEG image. It is important to note that this middle exposure is not re-encoded, but copied as a binary data stream to prevent re-encoding artifacts.

[1] Note that the actual exposure time ratios are not always the best choice for k due to the presence of under- and over-exposure regions.

Decompression Pipeline

The decompression pipeline is the reverse of the compression pipeline. Initially, the main JPEG image and its metadata are loaded followed by decompressing the metadata to recover the sign and difference matrices (Figure 2).

This is followed by applying perceptual decoding to undo the effect of perceptual encoding that was performed during compression:

$$L_{dif_x} = 255 \left(\frac{P_{dif_x}}{255} \right)^{2.2}. \quad (7)$$

Next, each exposure is reconstructed one by one starting from the exposures closest to the middle exposure by using Equation 8:

$$L_x = k_x L_{ref} + L_{dif_x} L'_{sign_x} \quad (8)$$

where

$$L'_{sign_x} = \begin{cases} 1 & L_{sign_x} < 5 \\ -1 & \text{otherwise} \end{cases} \quad (9)$$

At this point, all of the computed data is still in linearized form and must be converted to the nonlinear camera response domain using the recorded camera response function:

$$I_x = 255 f \left(\frac{L_x}{255} \right) \quad (10)$$

Finally, we align the exposures using the stored shift amounts to reconstruct a close approximation of the original JPEG exposures.

4 RESULTS

In this section, we present the results of the experiments conducted using a dataset of 10 exposure sequences representing various scene conditions (Figure 3).

The quality measurements are done using two well-known objective image quality metrics, namely structural similarity index (SSIM) [Wan04a] and peak signal-to-noise ratio (PSNR) [Teo94a] using the original exposures as ground-truth for both metrics.

Initially, we show the visual quality of our method in lossless compression mode. Then we demonstrate the effect of lossy compression on both file size and recovered image quality.

Lossless Compression

Here we illustrate the lossless compression and decompression results of our algorithm via a test case containing five exposures of the *Mug* image. In Figure 4, the first line shows the original exposure set, the second line displays the recovered exposures by using our

 ISBN 978-80-86943-37-4

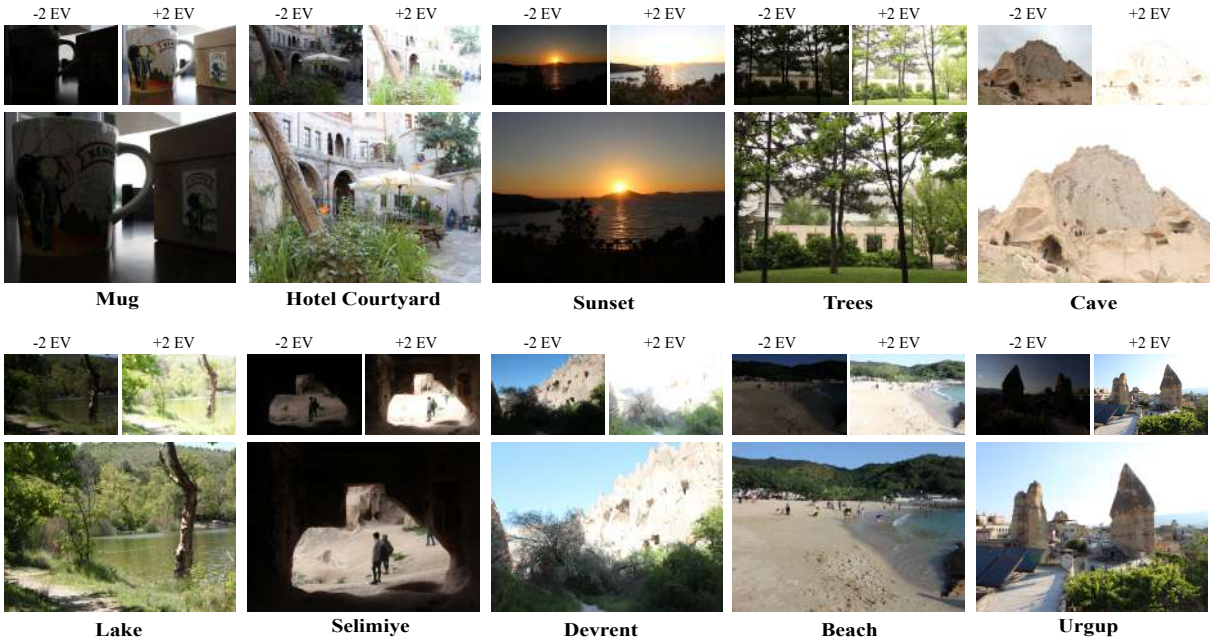


Figure 3: Our quality experiments are conducted using 10 exposure sequences whose middle exposures with two other exposures are shown in this figure.

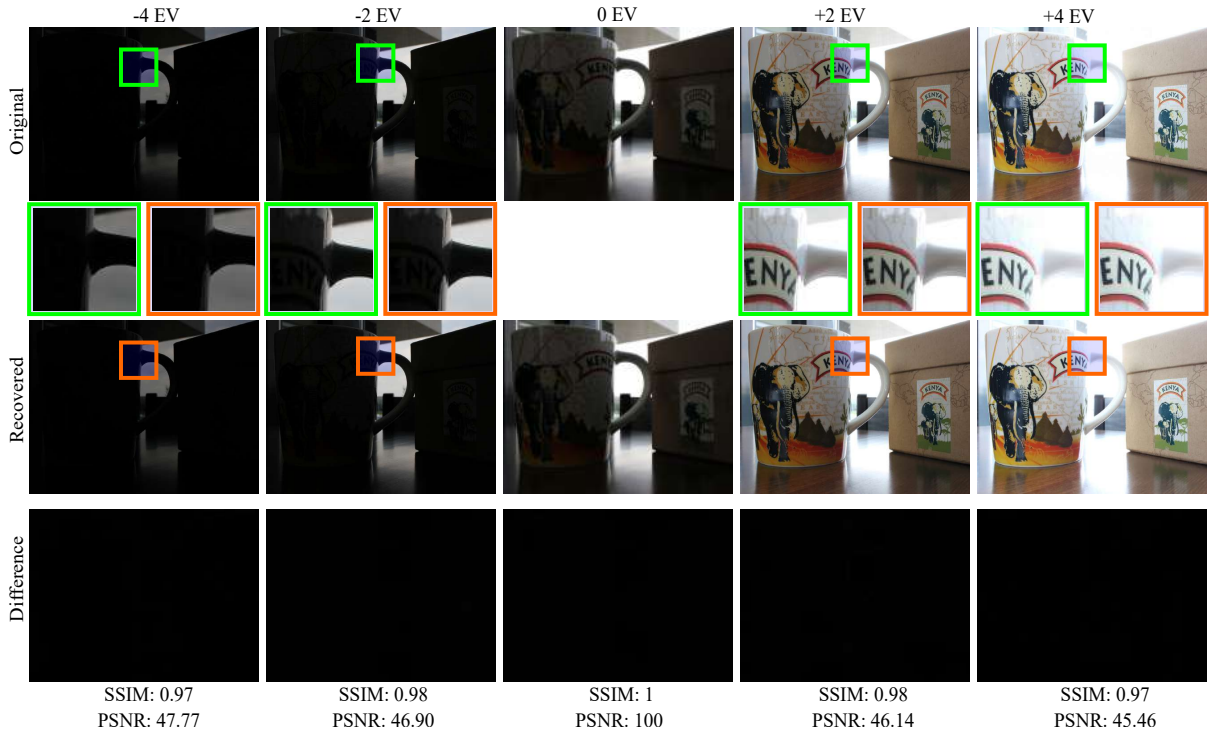


Figure 4: The results of lossless compression for a test image. The insets show the selected regions in the original resolution.

ISBN 978-80-86943-37-4

method, and finally the last line represents the difference between the original and the recovered images.

For each exposure we calculate the SSIM index and the PSNR value to show the perceptual and numerical similarity between the recovered and original images. The results show that the recovered exposures are very close to the original ones in the lossless compression case, as expected. The reason for not obtaining exact equality is due to the quantization of the image differences to whole numbers. If the storage size is not very critical and the main aim is to be able to reconstruct the original exposures, our lossless compression mode offers a plausible solution.

Lossy Compression

For many applications, reducing the image size can be important to minimize the storage and transmission costs, and therefore using the lossless mode may not be desirable. In Figure 5, we report the results obtained in case the image difference data is compressed in a lossy manner. Similar to Figure 4, the top row shows the originals, the middle row our reconstructions, and the bottom row their differences. Although the SSIM and PSNR values can be seen to have dropped compared to the lossless mode, the reconstructions can still be considered close to the originals.

The Effect of Perceptual Encoding

In this section, we demonstrate the effect of using perceptual encoding before compressing the image difference data. In Figure 6, (a) represents the original image, (b) the result obtained with perceptual encoding, (c) the result without perceptual encoding, and (d) the difference between the two. As can be seen both visually and numerically indicated by the SSIM and PSNR values, perceptual encoding has an important contribution to the quality of the reconstructed exposures.

The Effect of Exposure Ratio Optimization

We described in Section 3 that optimizing the exposure ratios to compute image differences leads to better results than using the actual exposure time values (see Equation 4). This can be explained using the following simplified example. Imagine that two pixels' values are 170 and 200 in an exposure captured with Δt exposure time. Due to sensor saturation, the same pixels' values could be both clamped at 255 in the next exposure captured with $2\Delta t$ exposure time. If one uses the actual exposure time ratio to normalize the second image one would obtain 127.5 for both pixels. The difference of these with 170 and 200 would yield 42.5 and 72.5 respectively. These are the values that would be compressed with the JPEG compression algorithm.

On the other hand, if one scales the second image with a ratio of $185/255$, both pixels in the difference image would attain the value of 15. Note that not only this difference would compress better, the reconstruction error from encoding/decoding such a difference would actually be smaller. Although this example is an over-simplification, we observed in practice that it is the very reason that optimizing the exposure ratios improves the reconstruction quality over using the actual exposure times. As demonstrated in Figure 7, the benefit of this optimization is most visible in long exposures with large saturated regions.

Table 1 shows the effect of this optimization for 4 different image sets each comprised of 9 exposures. It not only leads to decrease in the compressed image size, but also produces higher SSIM and PSNR values.

The Effect of JPEG Compression

Lossy compression results are also affected by JPEG compression quality, which control the degree of quantization of the DCT coefficients. We therefore conducted experiments to determine the effect of JPEG compression on both image quality and size. The results are shown in Table 2. When we consider both image quality and the amount of compression achieved, the quality level of 85 yields a good compromise. At this level, the compressed image size was found to stay lower than the total size of the original exposures, while the SSIM and PSNR values were generally high.

5 CONCLUSION & FUTURE WORK

In this study, we presented a scheme called compressed exposure sequences that provides a simple solution for storage and maintenance of bracketed exposure sequences commonly used by HDR photographers. We used JPEG as a container format due to its wide availability and its support for metadata, which makes our solution possible. Our two modes, namely lossless and lossy, provide flexibility to the users. Those who prioritize image quality over storage size may choose to use the lossless mode. Conversely, those users who are mainly concerned with the storage size can opt for the lossy mode. In both modes, the middle exposure is stored exactly as it is produced by the camera so the users can be certain that it remains unaltered by our algorithm.

We have shown that plausible results are obtained by using a dataset of 10 exposure sequences. As one of the primary future works, we plan to extend this dataset with more images. Secondly, the effect of camera and object movement on both image quality and storage size must be better evaluated. Finally, as bracketed sequences resemble video frames that are captured in close succession, video compression algorithms can be used to further reduce the storage size without compromising quality.

 ISBN 978-80-86943-37-4

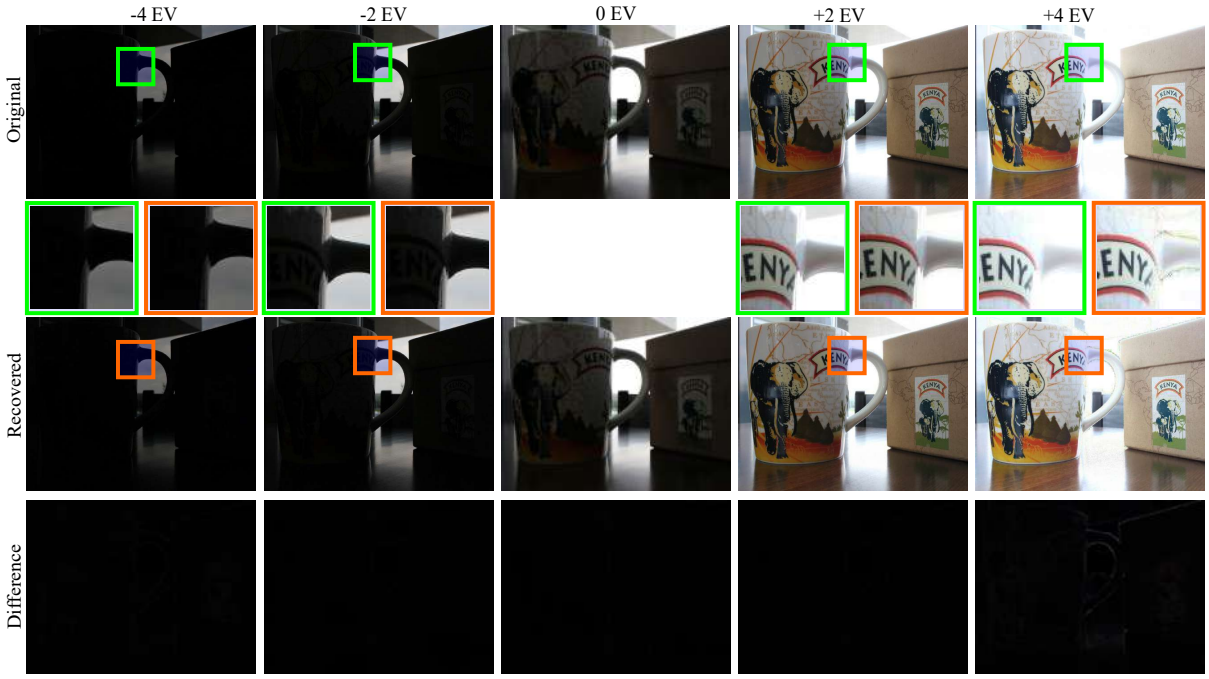


Figure 5: The results of lossy compression with JPEG compression quality value of 85 for a test image.

Figure 6: a: Original image (Urgup), b: Reconstructed image with perceptual encoding, c: Reconstructed image without perceptual encoding, d: Difference between b and c (10 times enlarged).

	Mug (Original size: 32.6 MB)				**Urgup** (41.5 MB)				**Sunset** (39.5 MB)				**Selimiye** (33.7 MB)			
	Computed Exposure Rate 23.4		**Actual Exposure Rate** 27.8		**Computed Exposure Rate** 31.4		**Actual Exposure Rate** 33.7		**Computed Exposure Rate** 31.7		**Actual Exposure Rate** 33.3		**Computed Exposure Rate** 29.8		**Actual Exposure Rate** 32	
CES Size (MB)	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***
Exp 1	0.96	45.36	0.96	44.48	0.96	44.64	0.95	42.66	0.94	37.68	0.93	37.48	0.97	46.49	0.96	43.34
Exp 2	0.96	44.40	0.97	44.91	0.94	41.35	0.93	40.37	0.92	36.01	0.93	36.16	0.96	44.64	0.96	42.90
Exp 3	0.95	42.82	0.96	44.03	0.94	39.46	0.93	38.55	0.90	34.53	0.91	34.85	0.95	41.83	0.95	41.68
Exp 4	0.95	41.33	0.96	43.14	0.94	38.41	0.93	37.94	0.93	34.87	0.94	35.17	0.94	40.62	0.94	40.94
Exp 5	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
Exp 6	0.96	42.35	0.96	41.36	0.93	33.53	0.91	32.71	0.95	32.26	0.94	31.68	0.93	37.84	0.94	38.69
Exp 7	0.95	40.04	0.90	36.68	0.90	29.88	0.83	27.21	0.92	29.80	0.89	27.68	0.90	34.82	0.89	34.21
Exp 8	0.95	39.51	0.83	32.96	0.88	28.15	0.76	23.30	0.92	30.25	0.83	25.17	0.92	36.06	0.82	31.85
Exp 9	0.95	38.50	0.77	29.50	0.90	28.43	0.67	20.86	0.92	30.71	0.72	22.27	0.93	34.87	0.72	27.20

Table 1: The effect of exposure rate optimization (Equation 4) on total size and image quality. The *Computed Exposure Rate* column shows the results obtained by using Equation 4 to compute the exposure ratios. The *Actual Exposure Rate* column shows the results if the original exposure time ratios are used. Note that the optimization yields both smaller total size (*CES Size*) and better image quality (*SSIM* and *PSNR*).

ISBN 978-80-86943-37-4

Figure 7: a: Original image (Sunset, shortest exposure), b: Reconstructed image with exposure rate optimization, c: Reconstructed image with the original exposure time ratios, d: Difference between b and c (10 times enlarged). e: Original image (Sunset, longest exposure), f: Reconstructed image with exposure rate optimization, g: Reconstructed image with original the exposure time ratios, h: Difference between f and g (10 times enlarged).

6 ACKNOWLEDGMENTS

We thank to ASELSAN A.Ş. for their support and encouragement.

7 REFERENCES

[AlllC16a] Peter Shirley Alan Chalmers, Patrizio Campisi and Igor Garcia Olaizola. *High Dynamic Range Video: Concepts, Technologies, and Applications*. Academic Press (Elsevier), London, 2016.

[Art15a] Alessandro Artusi, Rafał K. Mantiuk, Thomas Richter, Philippe Hanhart, Pavel Korshunov, Massimiliano Agostinelli, Arkady Ten, and Touradj Ebrahimi. Overview and evaluation of the jpeg xt hdr image compression standard. *Journal of Real-Time Image Processing*, pages 1–16, 2015.

[CNEa] CNET. What is HDR for TVs, and why should you care? https://www.cnet.com/news/what-is-hdr-for-tvs-and-why-should-you-care/. Accessed: 2017-11-25.

[Deu96a] Peter Deutsch and Jean-Loup Gailly. Zlib compressed data format specification version 3.3. Technical report, 1996.

[Duf09a] Frédéric Dufaux, Gary J Sullivan, and Touradj Ebrahimi. The jpeg xr image coding standard [standards in a nutshell]. *IEEE Signal Processing Magazine*, 26(6):195–204, 2009.

[Kai02a] F. Kains, R. Bogart, D. Hess, P. Schneider, and B. Anderson. OpenEXR. https://www.openexr.org/, 2002. Accessed: 2017-11-26.

[Lar98a] Gregory Ward Larson. Overcoming gamut and dynamic range limitations in digital images. *Color and Imaging Conference*, 1998(1):214–219, 1998.

[Man06a] Rafał Mantiuk, Alexander Efremov, Karol Myszkowski, and Hans-Peter Seidel. Backward compatible high dynamic range mpeg video compression. *ACM TOG*, 25(3):713–723, 2006.

[Mit99a] T. Mitsunaga and S. K. Nayar. Radiometric self calibration. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 1, page 380 Vol. 1, 1999.

[Pen92a] William B Pennebaker and Joan L Mitchell. *JPEG: Still image data compression standard*. Springer Science & Business Media, 1992.

[Rei07a] Erik Reinhard, Timo Kunkel, Yoann Marion, Jonathan Brouillat, Rémi Cozot, and Kadi Bouatouch. Image display algorithms for high- and low-dynamic-range display devices. *Journal of the Society for Information Display*, 15(12):997–1014, 2007.

[Rei10a] Erik Reinhard, Greg Ward, Sumanta Pat-

 ISBN 978-80-86943-37-4

			JPEG QUALITY													
			40		50		60		70		80		85		90	
	Original Exposure Size (MB)		CES Size (MB)	%	CES Size (MB)	%	CES Size (MB)	%	CES Size (MB)	(%	CES Size (MB)	%	CES Size (MB)	%	CES Size (MB)	%
Mug	**32.6**		**10.1**	**30.98**	**11.2**	**34.36**	**12.5**	**38.34**	**14.9**	**45.71**	**19.3**	**59.20**	**23.4**	**71.78**	**31**	**95.09**
			SSIM	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***
		Min	0.92	36.02	0.93	36.85	0.93	37.48	0.94	38.37	0.94	39.24	0.95	39.89	0.95	40.56
		Avg	0.94	39.06	0.94	39.87	0.94	40.10	0.95	41	0.96	41.72	0.96	42.32	0.96	42.93
		Max	0.96	44.07	0.96	44.90	0.97	44.89	0.97	45.71	0.97	46.29	0.97	46.74	0.97	47.15
Hotel Courtyard	**50.9**		**21.4**	**42.04**	**23.5**	**46.17**	**25.9**	**50.88**	**29.8**	**58.55**	**36.5**	**83.30**	**42.4**	**83.30**	**53.2**	**104.52**
			SSIM	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***
		Min	0.74	24.74	0.76	25.57	0.77	26.12	0.79	26.78	0.81	27.87	0.83	28.56	0.84	29.6
		Avg	0.80	28.20	0.82	29.27	0.83	29.71	0.84	30.41	0.86	31.61	0.87	32.38	0.89	33.29
		Max	0.93	38.83	0.94	39.91	0.94	40.31	0.95	40.96	0.96	41.54	0.96	42.27	0.96	42.59
Sunset	**39.5**		**14.40**	**36.46**	**16.00**	**40.51**	**17.90**	**45.32**	**21.20**	**53.67**	**26.70**	**67.59**	**31.80**	**80.51**	**40.70**	**103.04**
			SSIM	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***
		Min	0.77	26.06	0.78	26.71	0.78	27.29	0.79	28.10	0.80	29.10	0.81	29.74	0.82	30.66
		Avg	0.83	29.27	0.84	29.97	0.85	30.48	0.85	31.17	0.86	32.06	0.87	32.63	0.88	33.44
		Max	0.88	33.05	0.89	33.53	0.89	33.80	0.90	34.12	0.91	34.71	0.92	35.19	0.93	35.83
Trees	**64.8**		**27.1**	**41.82**	**29.8**	**45.99**	**32.8**	**50.62**	**37.6**	**58.02**	**45.8**	**70.68**	**53**	**81.79**	**65.9**	**101.70**
			SSIM	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***
		Min	0.66	20.20	0.68	20.84	0.70	21.26	0.72	21.8	0.74	22.58	0.76	23.16	0.78	24.04
		Avg	0.72	25.12	0.75	25.84	0.76	26.38	0.78	27.08	0.8	28.05	0.82	28.71	0.84	29.61
		Max	0.83	32.65	0.85	33.45	0.86	33.92	0.88	34.73	0.89	35.6	0.9	36.19	0.91	36.88
Cave	**30.60**		**14.00**	**45.75**	**15.60**	**50.98**	**17.40**	**56.86**	**20.30**	**66.34**	**25.10**	**82.03**	**29.40**	**96.08**	**37.50**	**122.55**
			SSIM	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***
		Min	0.71	27.90	0.73	28.81	0.74	29.36	0.76	30.07	0.79	31.15	0.80	31.84	0.83	32.94
		Avg	0.86	34.24	0.87	34.87	0.88	35.61	0.89	36.62	0.90	37.85	0.91	38.65	0.92	39.72
		Max	0.98	39.84	0.98	39.48	0.98	40.88	0.99	42.81	0.99	44.26	0.99	45.38	0.99	46.60
Lake	**65.6**		**28.6**	**43.60**	**31.7**	**48.32**	**35**	**53.35**	**40.3**	**61.43**	**49.4**	**75.30**	**57.5**	**87.65**	**71.8**	**109.45**
			SSIM	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***
		Min	0.52	17.98	0.55	18.57	0.58	19.07	0.61	19.69	0.64	20.6	0.67	21.27	0.7	22.31
		Avg	0.67	24.67	0.69	25.47	0.71	26.05	0.74	26.74	0.77	27.84	0.79	28.57	0.81	29.52
		Max	0.84	35.35	0.86	36.30	0.87	36.78	0.89	37.51	0.91	38.55	0.92	39.2	0.93	39.95
Selimiye	**33.7**		**12.60**	**37.4**	**14.20**	**42.1**	**16.00**	**47.4**	**19.20**	**56.9**	**24.80**	**73.6**	**29.90**	**88.7**	**39.30**	**116.6**
			SSIM	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***
		Min	0.72	30.78	0.73	31.51	0.73	31.94	0.74	32.48	0.75	33.42	0.75	33.96	0.76	34.66
		Avg	0.82	34.49	0.82	35.24	0.83	35.55	0.84	36.07	0.85	36.83	0.85	37.28	0.86	37.79
		Max	0.88	39.17	0.88	39.85	0.89	39.91	0.90	40.31	0.91	40.86	0.92	41.14	0.93	41.40
Devrent	**57.2**		**23.4**	**40.91**	**25.9**	**45.28**	**28.7**	**50.17**	**33.2**	**58.04**	**40.8**	**71.33**	**47.5**	**83.04**	**59.3**	**103.67**
			SSIM	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***
		Min	0.73	23.63	0.75	24.52	0.76	25.05	0.78	25.84	0.81	26.95	0.83	27.73	0.85	28.92
		Avg	0.79	25.51	0.81	26.47	0.83	27.13	0.85	28.02	0.87	29.23	0.88	30.07	0.9	31.25
		Max	0.86	27.67	0.88	28.51	0.89	28.97	0.91	30.01	0.93	31.47	0.94	32.35	0.95	33.43
Beach	**55.7**		**24.20**	**43.45**	**26.90**	**48.29**	**29.80**	**53.50**	**34.60**	**62.12**	**42.70**	**76.66**	**50.00**	**89.77**	**62.90**	**112.93**
			SSIM	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***
		Min	0.68	22.61	0.71	23.12	0.73	23.66	0.75	24.41	0.76	25.27	0.76	25.91	0.77	26.88
		Avg	0.73	27.41	0.76	28.41	0.77	28.86	0.79	29.54	0.81	30.31	0.83	30.84	0.84	31.59
		Max	0.76	34.63	0.79	35.27	0.81	35.36	0.83	35.54	0.86	36.15	0.88	36.52	0.90	36.88
Urgup	**41.5**		**14.70**	**35.42**	**16.30**	**39.28**	**18.10**	**43.61**	**21.20**	**51.08**	**26.60**	**64.10**	**31.60**	**76.14**	**40.50**	**97.59**
			SSIM	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***	***SSIM***	***PSNR***
		Min	0.77	27.74	0.77	25.51	0.77	25.99	0.77	26.59	0.78	27.51	0.78	28.23	0.78	29.24
		Avg	0.81	30.92	0.82	31.54	0.83	31.89	0.84	32.41	0.85	33.11	0.85	33.60	0.86	34.23
		Max	0.86	37.26	0.87	37.57	0.88	37.67	0.89	37.98	0.90	38.21	0.91	38.38	0.92	38.55

Table 2: Lossy compression results with respect to different JPEG qualities (40 – 90); each JPEG quality column gives the size of the compressed sequence of images *(CES Size)*, the relative size as a percentage of the original total exposure size *(%)*, and the quality metric results *(SSIM & PSNR)*.

tanaik, and Paul Debevec. *High Dynamic Range Imaging: Acquisition, Display and Image-Based Lighting*. Morgan Kaufmann, San Francisco, second edition edition, 2010.

[Sko01a] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001.

[Teo94a] P. C. Teo and D. J. Heeger. Perceptual image distortion. In *Proceedings of 1st International Conference on Image Processing*, volume 2, pages 982–986 vol.2, Nov 1994.

[Wal92a] Gregory K Wallace. The jpeg still picture compression standard. *IEEE Trans. on Consumer Electronics*, 38(1):xviii–xxxiv, 1992.

[Wan04a] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[War94a] Gregory J. Ward. The radiance lighting simulation and rendering system. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 459–472, New York, NY, USA, 1994. ACM.

[War03a] Greg Ward. Fast, robust image registration for compositing high dynamic range photographs from hand-held exposures. *Journal of graphics tools*, 8(2):17–30, 2003.

[War06a] Greg Ward and Maryann Simmons. JPEG-HDR: A backwards-compatible, high dynamic range extension to JPEG. In *ACM SIGGRAPH 2006 Courses*, page 3. ACM, 2006.

 ISBN 978-80-86943-37-4

 ISBN 978-80-86943-37-4

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

The City Metaphor in Software Visualization

Clinton L. Jeffery
Department of Computer Science
University of Idaho
875 Perimeter Drive MS 1010
USA 83844, Moscow, Idaho
jeffery@cs.uidaho.edu

ABSTRACT

A city metaphor has become a popular method of visualizing properties of program code. This paper provides an overview of research projects that employ this metaphor for a wide range of software engineering tasks. Until now, projects employing the city metaphor have primarily focused on visualizing static and semi-static properties of software repositories, such as understanding how a program's source code structure is changing over time, and who is changing what. This paper compares these existing code cities and suggests likely avenues of future research.

Keywords

software visualization; code cities

1 INTRODUCTION

In the 1980's, works of science fiction such as TRON and Neuromancer popularized the notion that code and data could be visualized in 3D and understood by means of animation and anthropomorphization. A metaphor was needed in order for the human viewer to understand the visualization of abstract mathematical constructs that operate on scales beyond our normal cognitive range. Since humans are trained from an early age to understand how to navigate through man-made structures in the real world, it was natural to envision software as a city inhabited by many dynamic entities whose behavior could be understood by means of direct observation.

The first generation of virtual reality hardware in the late 1990's inspired researchers to attempt to implement the metaphor of the software city with headsets and SGI workstations. Almost simultaneously, rapid advances in the game industry and the internet made such shared online virtual 3D spaces popular and useful, even without specialized hardware.

However, the rate of development of software city visualizations has been modest and has followed the boom-bust cycle of virtual reality technologies, rather than the rapid rate at which computer game technology has advanced. The rate of advances in software cities has been limited by the difficulty of programming them, in addition to the varying rate at which funding for VR research has been available.

This paper surveys the published research on software city visualizations. The goal is to understand the range of results achieved, their interconnections, and what gaps or missing pieces are needed in order for this genre of software tools to become more widely developed and used.

By necessity, the paper draws a line at *city* visualizations of software. Thus, it does not consider some fine work exploring other metaphors for software, including metaphors that employ a galactic or solar system space metaphor, an atomic metaphor, a cellular metaphor, or other geographic metaphors such as islands [Kuhn08] [Misi18] [Schr18]. Any metaphor that helps users is valuable, and some of this related research has produced gorgeous images. Nevertheless, although some software might arguably be developed by means that fit one of these other metaphors, most software that gets studied is written as a semi-planned artificial human construct, for which the city metaphor is a better fit than other metaphors where human planning is not so obviously involved.

2 RESEARCH QUESTIONS

The paper first establishes via literature survey what coarse-grained, large-scale static and semi-static software properties have been depicted using software cities. It then asks the following research questions, which focus on the question of dynamic analysis of program execution behavior:

- What fine-grained, dynamic program execution behaviors have been depicted using software cities?
- Is it possible to integrate both large-scale static and fine-grained dynamic information into the same software city visualization?

3 EXISTING IMPLEMENTATIONS

Software World and Component City

Knight and Munro used the Maverik VR toolkit [Hubbold99] to develop a tool called Software World [Knig00] that is an early single-user example of the city metaphor for software visualization. Their layout maps Java classes to *districts*, within which methods are buildings whose height is one storey per 10 LOC. Parameters are depicted by exterior doors. Lighter and darker building colors indicate public and private. The mapping of one building onto one function seems suited to visualization of finer-grained details at the expense of scalability to larger systems.

The work of Knight and Munro was later subsumed by a tool called Component City, which depicts components as buildings, a larger granularity shared by most other research projects that employ the city metaphor [Charters02]. Component City was not based on the Software World implementation, but instead uses XML and XSLT to generate VRML that can be viewed in browsers with a suitable plug-in. The published papers on Software World and Component City do not include an evaluation, but their wide influence can be seen in their number of citations and the number of projects that utilize the same or similar metaphor.

Figure 1 shows two views from Software World. The upper image shows a street level view of many, varying sizes of methods. The lower image shows an overview of a district based on a reasonably sized class, with two methods that contain large amounts of code.

Code City

Wettel et al developed a system called Code City in which a building represents an entire class, with a height proportional to the number of methods, rather than to lines of code [Wett07]. Length and width of the buildings were proportional to the number of member variables. The layout of buildings was performed using a treemap, atop a land whose elevation indicated nesting level of packages.

The increased scalability of representing an entire class as a building allows Code City to visualize large systems. consisting of many thousands of classes. The tool is used in the study of code evolution over time in software repositories. Code City was evaluated by demonstrating its scalability to large real-world software systems such as ArgoUML, Azureus, and VisualWorks, allowing views of as many as 8,000 classes. The work has been heavily cited or directly used in much software city visualization work that came after it. Figure 2 shows views of two large software systems from Code City.

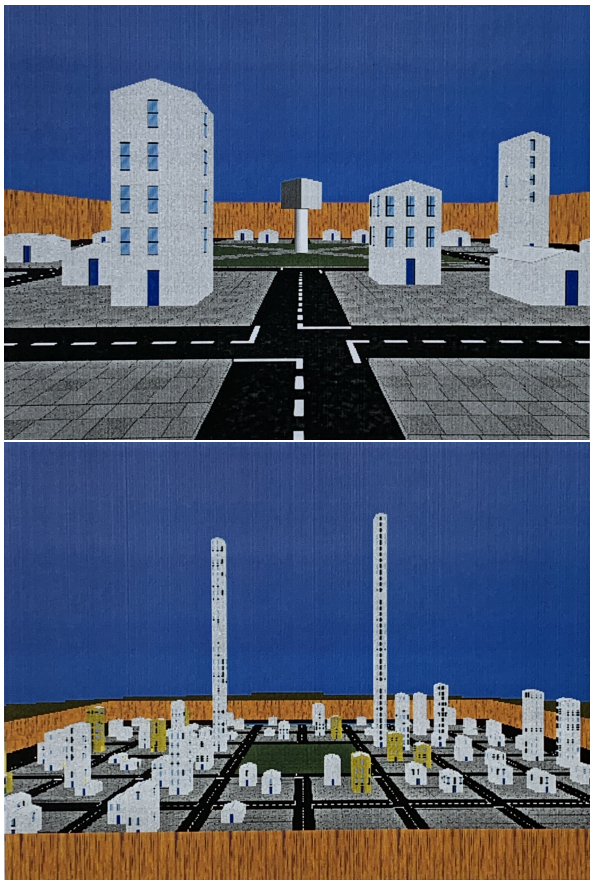


Figure 1: Software World (images courtesy of Claire Knight, from [Knigh00b])

Vizz3D

Panas et al developed an unnamed software city implementation for visualizing C++ programs using their visualization package named Vizz3D [Pana07]. Like Software World, their visualization depicts member functions as buildings; a blue platform serves as a pad for all the functions within a class. The number of lines of code in a member function is indicated by the texture of its exterior walls.

In contrast with most software city implementations, the Vizz3D based tool visually depicts more dependencies between functions and classes, such as “water towers” showing dependencies between header files and the classes defined within them. The system incorporates some dynamic analysis information, including the timing information produced by gprof, to augment a vast amount of static information that is depicted in the visualization.

ISBN 978-80-86943-37-4

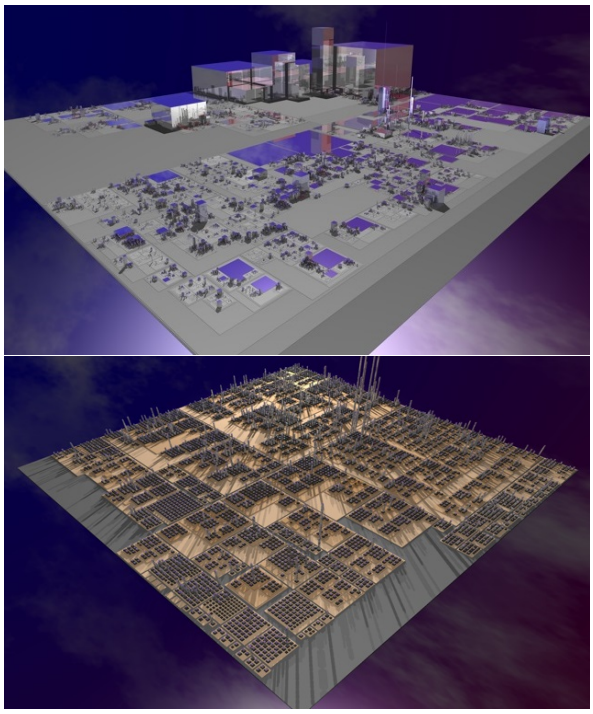


Figure 2: Code City (images courtesy of Michele Lanza)

Figure 3 shows a city generated using Vizz3d. The system is evaluated mainly by means of examples that demonstrate the range of stakeholders and roles (such as managers, testers, and engineers) that can utilize the single-view architectural visualization in performing their required tasks.

UML-City

Lange and Chaudron devised a UML-City visualization by combining two methods of reusing UML diagrams to produce visualizations in a system called MetricView Evolution [Lang07] [Lang07b]. A first method, called MetaView, focused on showing relationships between entities across reduced-size depictions of the entire set of UML diagrams available for a given software project. A second method, called MetricView, superimposed various software metrics data atop UML diagrams, as a vertical height, color, or shape.

The combination of MetaView and MetricView on large software systems produces an enriched, high-level view of the UML diagrams that resembles a city. The usefulness of these views was validated by a detailed user study of 100 M.S. students that demonstrates both improved speed and correctness of understanding UML diagrams when using the 3D visualizations than when trying to understand the software using a plethora of conventional, disconnected 2D UML diagrams.

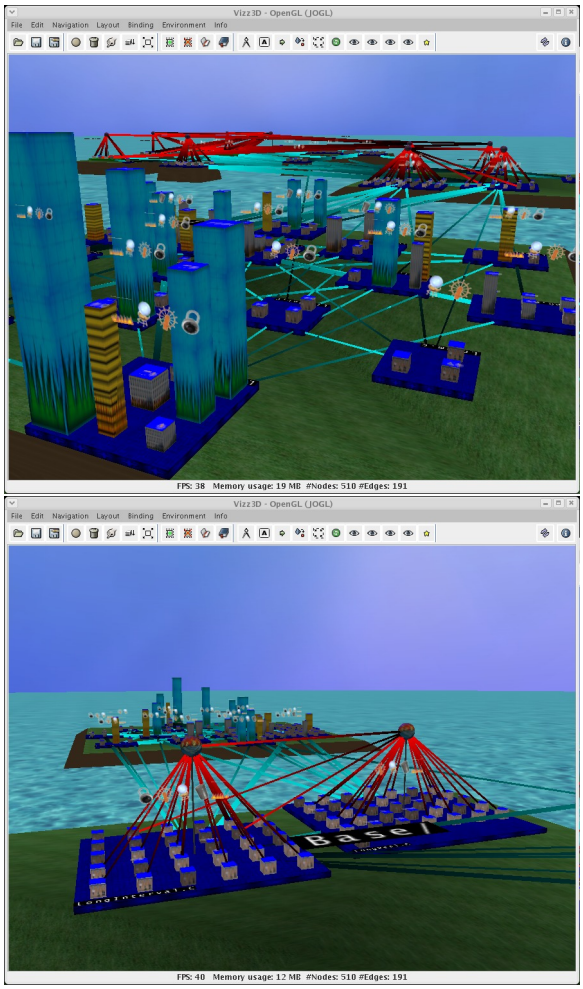


Figure 3: Vizz3d (images courtesy Thomas Epperly)

EvoStreets

Steinbruckner and Lewerentz created a system called EvoStreets [Stei10]. They adapt ideas from cartography, particularly the primary, secondary, and tertiary data models that respectively distinguish raw data, a detailed internal model, and a view-specific model that is render-friendly.

EvoStreets lays out classes, depicted as cylinders or cuboids, around streets, corresponding to directories or packages. The origin date of the various software components is indicated by their elevation given on a topographic map. Arguably, laying out structures around streets is more consistent with actual human cities than are space-filling techniques such as a treemap algorithm, although treemaps scale well.

EvoStreets has been used to study spatial orientation on VR head-mounted displays [Rude18]. Figure 4 shows an EvoStreets layout.

VizzAspectJ-3D

Bentrad and Meslati extended the city metaphor used in CodeCity to support the visualization of aspects in

ISBN 978-80-86943-37-4

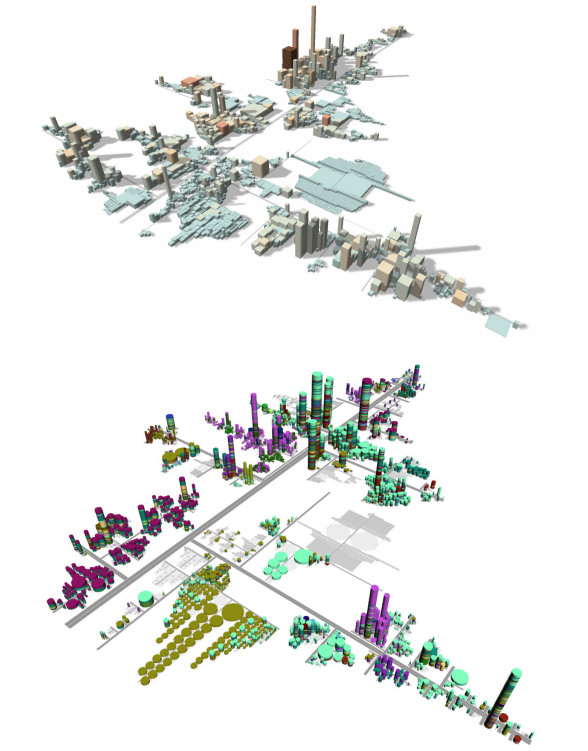


Figure 4: Evostreets (images ©Claus Lewerentz and Markus Uhlig, BTU Cottbus, 2016, by permission)

AspectJ programs [Bent11]. They developed an Eclipse plug-in in which both classes and aspects are buildings. The aspect buildings use the number of pointcuts and advice to determine building height. Different color-codings are used in complexity and dependencies views of the classes and aspects.

SkyscrapAR

Souza et al developed an augmented reality system called SkyscrapAR that visualizes Java systems with a metaphor similar to Code City except with building height denoting *code churn* [Souz12]. Churn refers to the idea that the number of lines of changes to a class may indicate its probability of containing software bugs. The visualization is projected onto a physical marker card that allows direct manipulation of the position, orientation, and scaling of the city, usually on a tabletop.

SynchroVis/ExplorViz

Waller et al invented a piece of software called SynchroVis in which classes are buildings [Wall13]. Each *instance* of a class is visualized as a storey on the building, allowing per-instance depiction of runtime dependencies and behavior.

SynchroVis depicts threads using colored arrows, allowing the visualization of concurrent behavior, communication and synchronization issues such as deadlock. Icons depict reasons why code may be suspended in a given instance, such as a method call to another object, or a wait on a semaphore. Figure 5 shows SynchroVis in action.

The software engineering group that developed SynchroVis also adapted their ExplorViz software to produce an Oculus Rift implementation with which to interact with a city metaphor [Fitt15]. This implementation allows translation, rotation, and scaling of the model using hand gestures input using a Kinect device.

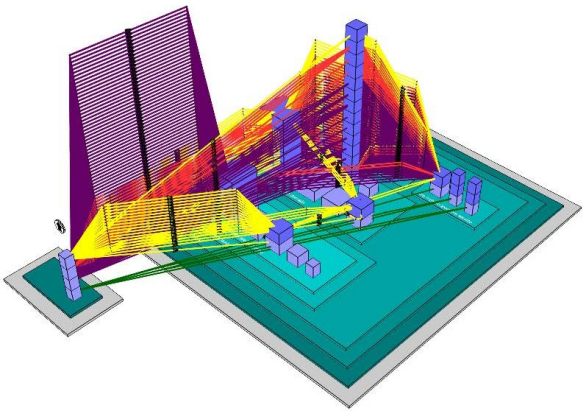


Figure 5: SynchroVis (image courtesy of Wilhelm Hasselbring)

CityVR

Merino et al created a VR application for the HTC Vive using the CodeCity system to generate the 3D visualization [Meri17]. The number of lines of code in a class is indicated via brightness. Source code for any class can be pulled up in a translucent 2D heads-up-display by pointing and clicking on the building. A similar system was used used on the HoloLens to explore whether immersive augmented reality technology will help to overcome usability issues with 3D visualizations in performing software engineering tasks [Meri18].

VR City

Vincur et al wrote a VR application for HTC Vive in which building layout is more organic because the amount of surface area contact between buildings is proportional to the amount of coupling between them [Vinc17]. For highly-coupled classes, the visualization may appear somewhat jigsaw-like. Although classes are represented by entire buidings with storeys representing methods as in Code City, the lengths and widths of storeys depict method metrics rather than class metrics. This simultaneously visualizes more information while resulting in building shapes that are more interesting and less uniform.

ISBN 978-80-86943-37-4

The VR City visualization allows the use of color-coding to highlight revision log information such as author commits, or dynamic information such as execution call traces. Figure 6 shows VR City's elegant use of color coding and translucence.

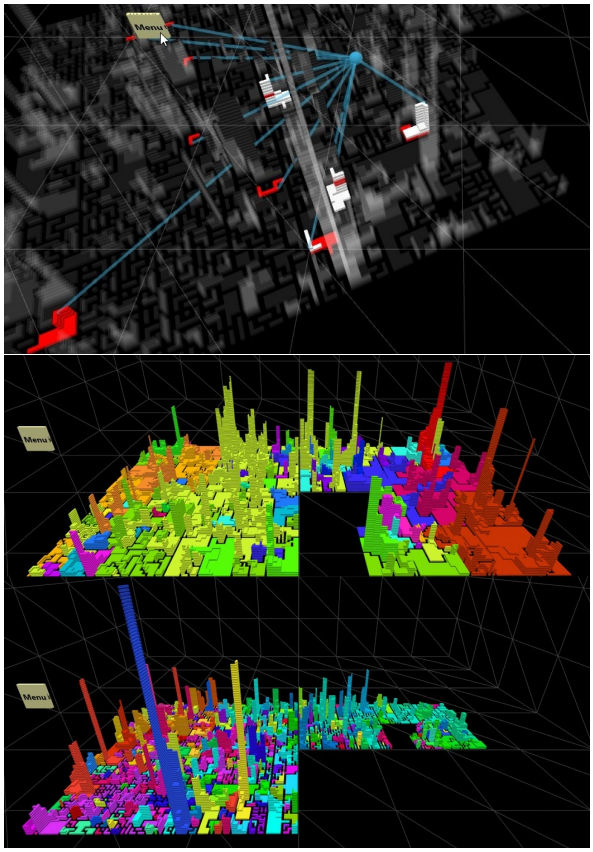


Figure 6: VR City (images courtesy of Ivan Polasek)

Code Park

Khaloo et al developed a system called Code Park that utilizes the city metaphor more for navigation of source code rather than for study of software architecture [Khal17]. Building size is proportional to code size but software engineering metrics are not the point of the tool.

The walls of buildings in Code Park contain source code with IDE-like syntax coloring and code navigation features. Users can toggle between birds-eye-view and first-person view on the ground. Movement between locations such as moving from a variable use to its definition is performed by animating up to birds-eye-view and then down to the new location, improving the user's orientation within the code base, compared with the more hyperlink-like teleportation that would be common in IDEs. Figure 7 shows Code Park's building layout (left) and a depiction on source code inside a building's walls.

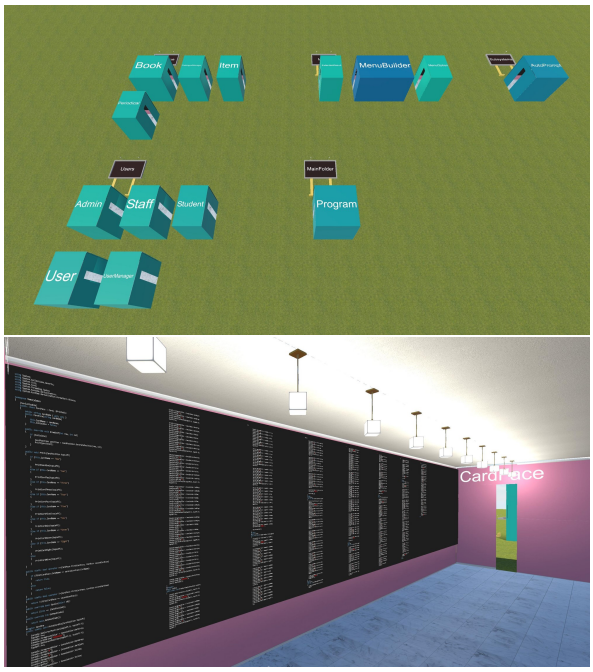


Figure 7: Code Park (images courtesy of Joseph LaViola)

High-Rise

Ogami et al developed a city-metaphor tool to deliver profile timing information in near real-time [Ogam17]. The height of buildings rises and falls over time, proportional to the amount of time consumed by the class in a fixed time frame such as the last L milliseconds. This design blurs the line between a city metaphor and a three dimensional bar chart.

By watching the profiles of different classes fluctuate over time, developers can notice issues that are not portrayed in conventional profiler output. It is clear that this visualization depicts transitions between major phases in an application very well.

LD-City

de Graaf and Khalili developed a city metaphor specifically for the representation of linked data, called LD-City [Graaf17]. Within that context, their visualization is used to detect anti-patterns such as *god classes* or *feature envy*. Their prototype implementation is built atop Three.js and related web technologies.

Comparison

Table 1. presents a comparison of existing software city visualizations. Column *Language* indicates the language(s) supported by the tool. Column *VR* specifies the type of VR hardware required, if any. Column *Building* conveys what a building denotes, such as a class or a function/method. Column *Src* indicates whether the tool depicts or integrates source code.

ISBN 978-80-86943-37-4

Column *Static* shows the static program properties depicted. Column *Dynamic* shows the dynamic program activity depicted. Column *Instr* gives the source/type of instrumentation used for dynamic program execution behavior, if any. An entry of *n/a* means an item is unknown or not applicable.

4 RESEARCH OPPORTUNITIES

Looking at Table 1, several obvious opportunities can be identified for near-term future research on visualizations using the city metaphor. Opportunities to better visualize information from static analysis are discussed first, followed by opportunities that involve incorporation of dynamic analysis information.

In this section, the research opportunities are sometimes couched in terms of suggested potential software-city-metaphor solutions, but really, better solutions may be empirically derived or simply invented by whomever incorporates that aspect into their software city visualization first. Many opportunities suggested here involve adding useful information to the city visualization. The research questions are often: what information is needed in a given context, and how to provide that information in a manner that is intuitive and does not result in cognitive overload.

Additional Static Information

None of the published works on the city metaphor deal with certain aspects of available static analysis information. For example, many of the published research works omit relationships between classes or functions beyond approximate juxtaposition of neighbors from the same file, package, or directory.

Tools that do visualize dependencies such as inheritance or caller-callee relationships risk overwhelming the viewer with too many interconnections, occluding too much of the cityscape. There is an opportunity to depict this information without overload, such as using animation, or an intermittent or context-sensitive depiction. It is possible to identify circumstances under which different dependencies merit visible rendition for a limited period of time, without overwhelming the viewer's other items under observation.

Integrating More Dynamic Information

Various existing tools feed dynamic analysis information into a city visualization post-mortem from log files, or at run-time. Either way, dynamic information has been far less utilitized in city-metaphor visualizations. This is probably due more to the difficulty of extracting fine grained program execution behavior than to the challenges posed to the visualization researcher of how to depict dynamic information in the city.

In any case, it is long overdue for software city visualizations to incorporate more kinds of dynamic information from program executions. Such information is not suggested to reduce or replace the static information currently being visualized, but rather, to complement and leverage the static information visualization in order to visualize dynamic behavior that is intrinsically more difficult to understand.

Function or method calls and returns, timing information and thread operations have been depicted by existing city visualization tools. In future software cities should include depictions of behavior such as data structures and access patterns, stack and heap allocation, and input/output behavior.

Populating the City

In real life, cities exist in order to organize and facilitate the coordinated efforts of large numbers of humans. If the city is found to be useful as a metaphor at all in software visualization, its most likely purpose is to provide structure and context that assists the viewer in interpreting the dynamic aspects of program behavior, avoiding that cognitive overload that is otherwise ubiquitous when viewing software in its abstract complexity.

The population of objects within a software city might be graphic primitives such as spheres or cuboids. Adopting the terminology of videogames, a dynamic rendered object with which the user may interact may be termed an *entity*. It is a research question whether human viewers will find it more effective to visualize software objects via anthropomorphic entities, such as humans, robots, or pets that populate the city, or whether they will find abstract shapes just as effective.

The most obvious category of entity that would naturally construe a citizen of the city would be an application domain object: a class (or record/struct) instance where the corresponding type is from the application domain (as opposed to being an implementation artifact such as a glue data structure type). Criterion for citizenship might also include lifespan, since many or most objects in some programs do not exist long enough to deserve much investment from the visualization subsystem.

Although invoking a method on a citizen means that a thread of execution will be active within that citizen's class, we are often primarily interested in where that citizen object is being accessed from. Rather than drawing an arrow from caller to callee, citizens should stroll toward the code from where they are accessed. Some citizens will always be hanging out within the code (building) of some containing or highly-coupled object, while other citizens may continuously be going to and fro between many different sites from which they are referenced in the code. Such citizens, which may indicate design or performance bugs, will be very evident if they are portrayed walking around on the streets of the city.

 ISBN 978-80-86943-37-4

Tool	Language	VR	Building	Src	Static	Dynamic	Instr
SoftwareWorld	Java	Maverik	function	n/a	LOC #methods public/private # parameters param. types	n/a	n/a
ComponentCity	XML	VRML	compon.	n/a	func. attributes	n/a	n/a
CodeCity	SmallTalk Java C++	n/a	class	n/a	# methods # attributes package struct.	n/a	n/a
Vizz3D	C/C++	n/a	function	n/a	LOC complexity call graphs contains inheritance str. conn. comp.	gprof	none (`-pg`)
SkyscrapAR	Java	AR	class	n/a	LOC churn	n/a	n/a
UML-City	UML	n/a	class various	n/a	various metrics author	n/a	n/a
VizzAspectJ	Java AspectJ	n/a	class aspect	n/a	# methods # pointcuts # advices	n/a	n/a
EvoStreets	Java	n/a	class	n/a	module age coupling # dependencies module size last mod. date author	n/a	n/a
SynchroVis ExplorVis	Java	Rift	class	n/a	inheritance implementation association	instances calls thread op	Kieker traces
CityVR	Java/C++	Vive	class	yes	LOC # methods # attributes	n/a	n/a
VR City	Java	Vive	class	yes	LOC # methods author coupling	trace loc.	inTrace traces
Code Park	C#	n/a	class	yes	size method names	n/a	n/a
High-Rise	Java	n/a	function	no	n/a	time	ASM injection
LD-City	LD-R	n/a	(dynamic)	n/a	#instances # properties	n/a	n/a

Table 1: Feature Sets of City-Metaphor Visualizations

Time Scales

Understanding the execution behavior of software systems includes a need to understand dynamic behavior at widely ranging time scales, perhaps from submicrosecond granularity to things that challenge human patience. A visualization tool can implement yet-another slider to control the speed of execution, and leave the user in control, but users have better things to think about. Alice in Wonderland and Star Trek are suggestive, but perhaps Dr. Who's *Weeping Angels* provide a better metaphor and potential user interface mechanism: the speed of execution behavior should automatically stop while an entity is being observed, either via explicit pointer selection or by entering a user's center of vision.

 ISBN 978-80-86943-37-4

Dealing with Dynamic Data

Some of the more dynamic oriented software cities animate the buildings themselves, with building height depicting the number of instances, or the amount of time spent in a class. Separate from the scalability problem introduced by such approaches, they have a metaphor problem. Although software cities are by no means required to be "realistic" or follow rules that would apply to human cities, the value of the metaphor depends on an analogy. Humans can recognize buildings as solid artificial structures that have a purpose and that can be navigated around.

While it is easy in software to grow and shrink a set of cuboids in an animated 3D bar chart, the notion of buildings rapidly growing and shrinking violates the city metaphor and makes the entities softer and more transient. Cities, streets, and topography are a more natural fit to depict static or semi-static data, providing a context in which to interpret the vast amounts of dynamic information that humans need to understand.

Whereas the challenge of visualizing and making comprehensible very large bodies of code is a semi-static one, a software city can portray large amounts of dynamic data: on the stack, on the heap, or entering and exiting via I/O ports. In most cases, the primary challenge is the depiction of large amounts of information, much of which is very short-lived.

Short-lived information may be categorized by its importance. If it is important enough that the user may need to interact with it, it may require a temporal stop or slowdown so that the user has time to take it in. If user interaction is not required, such information may still be useful, and might by depicted by means of a more subtle visual effect rather than an entity.

Multi-user Shared Viewing

From the earliest example of Software World, researchers have identified a networked multi-user environment as an objective for software city visualizations, but this feature adds enough implementation challenges that it is usually omitted. A substantial opportunity exists for a cloud-based software city whose open infrastucture is accessed and shared by multiple research groups.

Multi-Lingual City Visualization

Several existing projects support two or more mainstream languages such as Java and C++ by means of some standard common tool for obtaining static information, such as an Eclipse IDE plugin. While such multi-lingual support is compelling, no standard exists for high-level language-neutral dynamic analysis information. A standard is needed that provides higher-level dynamic information than that available via virtual machines that support managed code in many languages, such as the JVM or .Net.

In addition, many large software systems are written in a combination of languages. For example, a software city might depict a program written in a higher level language (such as Python or Java) with an understanding that its subterranean or aquatic regions depict the behavior of intermediate or lower-level elements, such as ones invoked through a native interface to libraries written in C or C++.

5 CONCLUSION

The city metaphor has been adopted by a growing number of research groups for the visualization of large, complex software systems. Existing research has been successful in scaling to real-world software with a large number of classes, for which many static properties of the program code can be immediately observed.

Compared with the static information for which they were originally developed, city visualizations have not been applied as much to the task of making dynamic program execution behavior visible and understandable. In that domain software cities are potentially even more useful than the applications where it has thusfar been applied. Tremendous potential exists for the research groups that apply this metaphor to a broader range of dynamic program execution behavior. Such projects will require access to general purpose, high-performance execution frameworks capable of reporting a wide range of execution events, such as JVMTI [JVMTI] or Alamo [Jeffery98].

Several of the recent research advances that use the city metaphor focus on VR, despite head-mounted displays still being inadequate for displaying large quantities of text. This has resulted somewhat in a bifurcation between high-text software cities that run on desktops, and low-text software cities that run on VR systems.

6 REFERENCES

[Bent11] Bentrad, S. and Meslati, D. 2D and 3D Visualization of AspectJ Programs. In 10th International Symposium on Programming and Systems, pp. 183-190, IEEE 2011.

[Charters02] Charters, S.M., Knight, C., Thomas, N., and Munro, M. Visualization for informed decision making; from code to components. In SEKE 02: Intl. Conference on Software Engineering and Knowledge Engineering, pp. 765-772, ACM Press 2002.

[Fitt15] Fittkau, F., Krause, A., and Hasselbring, W. Exploring Software Cities in Virtual Reality, in 3rd Working Conference on Software Visualization, IEEE, pp. 130-134, 2015.

 ISBN 978-80-86943-37-4

[Graaf17] de Graaf, K.A., and Khalili, A. Visualizing Linked Data as Habitable Cities. VOILA 2017, Third Intl. Workshop on Visualization and Interaction for Ontologies and Linked Data, Vienna, CEUR-WS vol. 1947, pp. 131-138, 2017.

[Hubbold99] Hubbold, R., Cook, J., Keates, M., Gibson, S., Howard, T., Murta, A., West, A., and Pettifer, S. GNU/MAVERIK: A micro-kernel for large-scale virtual environments. VRST'99, pp. 66-73, ACM, 1999.

[Jeffery98] Jeffery, C., Zhou, W., Templer, K., and Brazell, M. A Lightweight Architecture for Program Execution Monitoring. Proc. of PASTE'98. ACM SIGPLAN Notices 33(7).

[JVMTI] Java Virtual Machine Tool Interface (JVM TI). https://docs.oracle.com/javase/8/docs/technotes/guides/jvmti.

[Khal17] Khaloo, P., Maghoumi, M., Taranta, E., Bettner, D., and Laviola, J. Code Park: A New 3D Code Visualization Tool, in Working Conference on Software Visualization, IEEE, pp. 43-53, 2017.

[Knig00] Knight, C., and Munro, M. Virtual but Visible Software. 2000 IEEE Conference on Information Visualization. London, IEEE, pp 198-205.

[Knigh00b] Knight, C. Virtual software in reality. Ph.D. Thesis, University of Durham, 2000.

[Kuhn08] Kuhn, A., Loretan, P., Nierstrasz, O. Consistent Layout for Thematic Software Maps, Proceedings of the 15th Working Conference on Reverse Engineering, WCRE '08. pp. 209-218, 2008.

[Lang07] Lange, C. and Chaudron, M. Interactive Views to Improve the Comprehension of UML Models - An Experimental Validation, Proceedings of the 15th IEEE International Conference on Program Comprehension, pp. 221-230, 2007.

[Lang07b] Lange, C., Wijns, M., and Chaudron, M. A Visualization Framework for Task-Oriented Modeling Using UML, 40th Annual Hawaii International Conference on System Sciences, p. 289a, 2007.

[Meri17] Merino, L., Ghafari, M., Anslow, C. and Nierstrasz, O. CityVR: Gameful Software Visualization. in International Conference on Software Maintenance and Evolution (ICSME), Shanghai, IEEE, pp. 633-637, 2017.

[Meri18] Merino, Bergel, and Nierstrasz. Overcoming Issues of 3D Software Visualization through Immersive Augmented Reality, in Working Conference on Software Visualization, pp. 54-64, IEEE, 2018.

[Misi18] Misiak. et al. IslandViz: A Tool for Visualizing Modular Software Systems in Virtual Reality. In Working Conference on Software Visualization, IEEE, pp. 112-116, 2018.

[Ogam17] Ogami, K., Kula, R.G., Hata, H., Ishio, T. and Matsumoto, K. Using High Rising Cities to Visualize Performance in Real-Time, in Working Conference on Software Visualization, Shanghai, IEEE, pp. 33-42, 2017.

[Pana07] Panas, T., Epperly, T., Quinlan, D., Saebjornsen, A. and Vuduc, R. Communicating Software Architecture using a Unified Single-View Visualization, in Proc. ICECCS 2007, Auckland, IEEE, pp. 217-228.

[Rude18] Rüdel, M.O., Ganser, J., and Koschke, R. A Controlled Experiment on Spatial Orientation in VR-Based Software Cities, in Working Conference on Software Visualization, pp. 21-31, IEEE, 2018.

[Schr18] Schreiber and Misiak. Visualizing Software Architectures in Virtual Reality with an Island Metaphor. VAMR 2018, International Conference on Virtual, Augmented, and Mixed Reality, pp. 168-182, 2018.

[Souz12] Souza, R., Silva, B., Mendes, T. and Mendonça, M. SkyscrapAR: An Augmented Reality Visualization for Software Evolution, in II Workshop Brasileiro de Visualizaçao de Software, Natal, RN, Brasilian Computing Society, 2012, pp. 17-24.

[Stei10] Steinbrückner, F. and Lewerentz, C. Representing Development History in Software Cities, in Proceedings of the 5th international symposium on Software visualization, SOFTVIZ 2010, ACM, New York, pp. 193-202.

[Vinc17] Vincur, J., Navrat, P., and Polasek, I. VR City: Software Analysis in a Virtual Reality Environment, in International Conference on Software Quality, Reliability and Security Companion (QRS-C), Prague, IEEE, pp. 509-516, 2017.

[Wall13] Waller, J., Wulf, C., Fittkau, F., Doring, P., and Hasselbring, W. SynchroVis: 3D Visualization of Monitoring Traces in the City Metaphor for Analyzing Concurrency. in First IEEE Working Conference on Software Visualization, Eindhoven, Netherlands, IEEE, pp. 1-4, 2013.

[Wett07] Wettel, R. and Lanza, M. Visualizing Software Systems as Cities, in Proceedings of VISSOFT 2007 (4th IEEE International Workshop on Visualizing Software For Understanding and Analysis), pp. 92-99, IEEE Computer Society Press, 2007.

 ISBN 978-80-86943-37-4

ISBN 978-80-86943-37-4

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

Alignment of Point Clouds for Comparison of Infrastructures in Civil Engineering on Quality Control in Metrology

Hadi Khaksari Haddad
Computer Vision and Systems lab,
Department of Electrical and Computer Engineering, Laval University, Quebec city, Canada
hadi.khaksari-haddad.1@ulaval.ca

Denis Laurendeau
Computer Vision and Systems lab,
Department of Electrical and Computer Engineering, Laval University, Quebec city, Canada
Denis.Laurendeau@gel.ulaval.ca

ABSTRACT

For 3D point cloud registration, Go-ICP [Yang et al., 2016] has been shown to obtain the global optimal solution for a pair composed of a model point cloud and a data point cloud. Go-ICP mostly has been investigated only on standard sets of point clouds. In this paper, we demonstrate the remarkable efficacy of Go-ICP for the alignment of **very complex large-scale** point clouds to their corresponding **deformed** CAD models. In particular, given two distinct sets of point clouds taken from the exterior and the interior of a building, experiments demonstrate that Go-ICP is able to successfully align both of these sets to the point cloud of the CAD model of the whole building (both exterior and interior information included). With the experimentation presented in this paper, we demonstrate that Go-ICP can achieve excellent alignment results and that this approach can be deployed in applications aiming at comparing CAD models of a building ("as designed" model) to the point cloud of the actual building ("as-built" model). Experiments also demonstrate the efficacy of Go-ICP to align a deformed copy of a man-made object to the original object in quality control applications.

Keywords

Global 3D registration large-scale point clouds comparison of models Go-ICP.

1 INTRODUCTION

Point cloud registration is a fundamental problem in computer vision. Given two point clouds in different coordinate systems with different poses, the goal of registration is to find the transformation that best align the first point cloud to the other one. Point cloud registration plays an important role in many vision applications. An application addressed in this paper is the alignment of a large point cloud, obtained by scanning a structure such as a building, with a CAD model of this structure for the purpose of augmented reality. Comparing the "as-designed" structure (i.e. CAD) to the "as-built" structure (i.e. the actual building) is an important problem in civil engineering since it often occurs that, for various reasons, what has been built is not what was originally designed and it is desired to find where the difference occurs. Exploring this application leads to a complementary study that consists of aligning a point cloud to a deformed of copy of itself for quality control and metrology.

3D scanners, Lidars and Structure-from-Motion can be used to collect point clouds from large-scale infrastructure components such as buildings.

The Iterative Closest point (ICP) algorithm [Besl and McKay, 1992] is a well-known algorithm for registering point clouds under Euclidean transformation. ICP is also known for being subject to the problem of local minima and also requires that the relative pose between the point clouds be close to the rigid transformation that is needed to align them for the algorithm to converge.

The work presented in this paper explores a globally optimal solution to the Euclidean registration problem defined by ICP in 3D. The Go-ICP Method exploiting the well-established Branch-and-Bound (BnB) theory for global optimization [Yang et al., 2016] is adapted to the problem of aligning a point cloud collected on a very large structure with a deformed copy of the point cloud of the same structure. Another topic that is studied in the paper is to adapt Go-ICP to the alignment of a large point cloud of a structure (such as a building) with a *CAD model* of this structure. A third application to which Go-ICP is extended is to align the point cloud of man-made objects with deformed copies of themselves in the context of quality control in metrology. The paper also presents a thorough analysis of the Go-ICP hyper-parameters on the quality of registration.

The experiments presented in the paper demonstrate that Go-ICP succeeds in the alignment of a point cloud of a structure such as a building with a deformed copy

of itself or with a CAD model of the structure. This functionality can be used for instance in augmented reality applications in civil engineering.

2 RELATED WORKS

Significant research has been conducted on global alignment methods and global optimal alignment methods. A class of global method is based on shape descriptors. The authors of [Johnson and Hebert, 1999] present a Greedy algorithm for related general shape representation based on spin images on using shape contexts for shape matching in [Belongie et al., 2002]. Authors in [Gelfand et al., 2005] propose a robust global registration approach by using integral volume based on local geometry and BnB.

The authors of [Rusu et al., 2009] present the Fast Point Feature Histogram that finds registration by using Sample Consensus Initial Alignment (SAC-IA). Random Sample Consensus (RANSAC) is used for matching a model to data in an application for Local Determination Problem (LPD) [Gelfand et al., 2005]. Authors in [Irani and Raghavan, 1999] propose a random sampling similar to RANSAC for alignment using 2D points while authors in [Aiger et al., 2008] exploits 4-points congruent sets without pre-filtering or noisy data removal. The authors of [Makadia et al., 2006] find the 3D rotation of two Extended Gaussian Images (EGI) based on global shape descriptors.

Another class of global methods is based on stochastic optimization. The authors of [Sandhu et al., 2010] solved point cloud registration by using particle filtering and stochastic dynamics. Range images registration is achieved by using the Genetic Algorithm (GA) proposed in [Silva et al., 2005]. The work in [Blais and Levine, 1995] presents 3D registration for range images via a cost function maximizing the quality of registration. In [Papazov and Burschka, 2011], stochastic global optimization is used for pairwise rigid point cloud registration based a cost function robust to outliers.

Many approaches for global optimal registration are based on BnB. The authors of [Breuel, 2003] present an algorithm for geometric matching based on BnB for image pattern while authors in [Pfeuffer et al., 2012] propose geometric matching based on BnB for medical images. A framework based on Lipschitz global optimization theory for 3D registration using BnB is presented in [Li and Hartley, 2007]. This approach is very slow for large point-sets. Authors in [Olsson et al., 2009] find the global optimal registration via a framework matching point-to-point, point-to-line, and point-to-plane using BnB. Authors in [Parra Bustos et al., 2014] present an approach for fast search in BnB for geometric alignment. This method proposes the SO(3) spaces that are then searched by the BnB algorithms. Other global methods by Nicolas Mellado [Mellado et al., 2014] and Qian-Yi Zhou [Zhou et al., 2016] have been proposed.

The authors of [Yang et al., 2016] present the Go-ICP algorithm as a global optimal registration method based on BnB via SO(3) spaces. This approach supports 3D Euclidean registration with both translation and rotation (i.e. rigid transformation). It can work directly on dense point clouds without the need for a good initial pose or putative correspondences.

3 PROPOSED APPROACH FOR THE ALIGNMENT OF POINT CLOUDS WITH CAD MODELS AND EXPLORATION OF THE EFFECT OF HYPER-PARAMETERS OF GO-ICP ON THE ALIGNMENT

3.1 Solution of the problem of the alignment of a large point cloud with a CAD model

Registration is the process of aligning 3D data point sets that are captured from different viewpoints [1]. In this section, we focus on aligning two data sets, namely model (reference) points, and data points that are collected with a different pose. Formally, $X = \{x_i\}, i = 1,...,N$ and $Y = \{y_j\}, j = 1,...,M$ represent data points and model points, respectively. Note that it is usually assumed that the number of data points (N) is lower than the number of model points (M). Generally, the objective function for 3D registration is defined on the L_2 square error as follows:

$$E(R,t) = e_i(R,t)^2 = \sum_{i=1}^{N} \|Rx_i + t - y_{j^*})\| \qquad (1)$$

Where $R \in [\pi,\pi]^3$ is the rotation space and $t \in [-\varepsilon,\varepsilon]^3$ is the translation space, and $e_i(R,t)$ indicates the Euclidean distance between x_i transformed by (R,t) to y_{j^*} as the optimal correspondence between x_i and y_i.

I- Global 3D registration

GO-ICP [Yang et al., 2016] proposes to combine the Branch-and-Bound (BnB) optimization algorithm with ICP in order to find the global optimal transformation in Eq. 1. Unlike ICP, this algorithm is not sensitive to the initial transformation. Therefore, it can start from any initial pose and eventually end up to the global optimum solution. The key contribution of Go-ICP is the derivation of registration error bounds. In fact, identifying efficiently the upper and lower bounds of the regions of search space helps to find the global optimum more quickly (Fig. 1 (a)).

 ISBN 978-80-86943-37-4

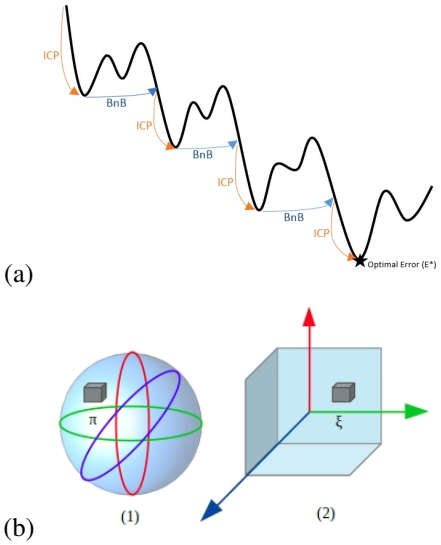


Figure 1: (a): Collaboration of BnB and ICP. (b): The search space consists of the rotation space and the translation space. The gray cubes are the sub cubes from these spaces (1) rotation space, which can be represented by a sphere with radius π. (2) the translation space which can be represented by a cube with length 2ξ. Redrawing from [Yang et al., 2016].

II- Domains for rotation and translation parameters

Generally, the search space of non-rigid registration consists of the rotation space and the translation space. The entire 3D rotation space can be modeled by a sphere with radius π. For the translation space, one can consider a bounded cube $[-\xi,\xi]^3$. The search space is depicted in Fig. 1 (b). Note that this cube encloses the sphere with radius π. The BnB algorithm is exploited to divide the search space, i.e. rotation space and translation space, into sub-cubes. To efficiently and intelligently search the rotation space and translation space, BnB should have access to the upper and lower bounds of sub-cubes. As the authors [Yang et al., 2016] point out, their main contribution is finding efficiently the upper and lower bounds of the search space. In this paper, we skip the theory that is proposed to find the error bounds and refer the reader to [Yang et al., 2016] for details.

3.2 3D Registration of deformed objects using Go-ICP

3.2.1- Data acquisition

In this section, we present the procedure for obtaining the data points and model points of an object in order to demonstrate how Go-ICP can be exploited to find the optimal registration between the point cloud of a man-made object and a modified copy of itself. This object, which was constructed by a 3D printer, consists of a cube, two convex cylinders with different heights, a convex hemisphere, a concave hemisphere, and one concave semi-cylinder (Fig. 2 (a)). We added some parts to the original object to create new objects (i.e."deformed" objects) for aligning the original object with these deformed objects (fig. 2 (b)) using Go-ICP.

We captured data points of the deformed objects and original points using a Creaform Go!scan 50 handheld meteorologic 3D scanner. The reflective markers installed on the the object (Fig.2 (c)) are used by the scanner for self-positioning. They are not used for the estimation of registration by Go-ICP. A total of 3345 data points were scanned on the original object and 6879 model points were scanned on the deformed object. To apply Go-ICP, we need to normalize the data sets in the interval $[-1,1]$. The search space for rotation is $[-\pi,\pi]^3$ and the one for translation is $[-0.5,0.5]^3$. To get the best result from BnB, its hyperparameters must be selected carefully. In the experiments, we considered all data points of the original object, i.e. = 3345. Go-ICP uses two hyper-parameters. Parameter ρ, called the "trimming" factor, controls the robustness to outliers while parameter ε, called the registration error threshold, controls the convergence of the registration process. Parameter ρ was set to 20% and ε was set to 5. The registration of the original object and the deformed object 1 using Go-ICP is executed in 6472.8 seconds with a Root-Mean-Square (RMS) error of 7.48679 while the RMS error of using ICP alone is 14.383 (Fig. 3). Note that with lower ρ values, the registration is not as good as for $\rho = 20\%$ using the RMS error as a criterion. In addition, the algorithm running time for smaller ε values is considerably greater. Finally, the alignment results of Go-ICP are good compared to ICP alone as one can verify qualitatively in Fig. 3. The Go-ICP Method was implemented in C++ on a PC with an Intel core i7 2*4.00 GHz CPU and 32 GB RAM.

3.2.2- Analysis of the Hyperparameters of Go-ICP on the registration of a point cloud with a deformed copy of itself

The effect of trimming

To remove possible outliers, a trimming procedure can be applied on data points so that ρ percent of data points that are far from the model points are removed. Based on our observations, we found that parameter ρ (the trimming factor) has a major influence on the RMS error and running time of the algorithms (BnB and ICP). More precisely, a large value for ρ can result in erroneous registered data points since some inliers that are essential for an accurate registration are removed by the trimming procedure. On the other hand, a very small

ISBN 978-80-86943-37-4

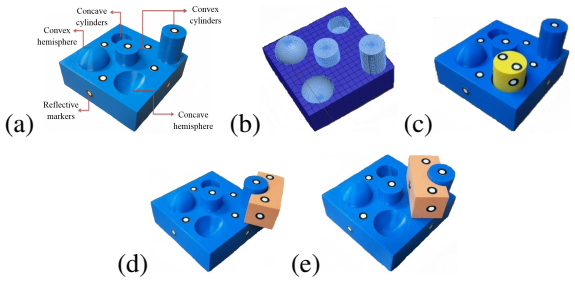


Figure 2: The data and the objects. (a) is the original object (3D print), (b) is the CAD model of the original object (data point cloud), (c) is deformed object 1 obtained by adding a convex cylinder in the concave hemisphere, (d) is deformed object 2 obtained by adding a block to the outside of the tall convex cylinder, and (e) is deformed object 3 obtained by adding a block to the inside of the tall convex cylinder.

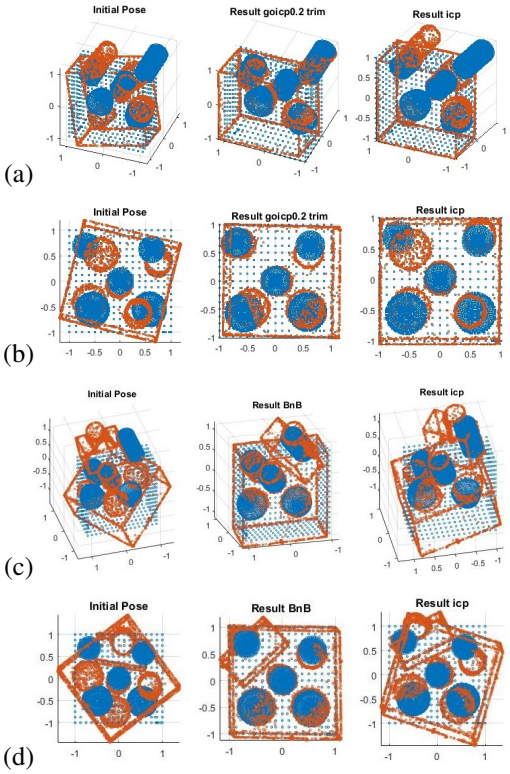


Figure 3: The results of Go-ICP for 3D registration of deformed objects. The blue point clouds are the data and red point clouds are the models (deformed objects). (a) is the 3D registration of deformed object 1 (view: axis of XYZ), (b) is the 3D registration of deformed object 1 (view: axis of xz), (c) is the 3D registration of deformed object 3 (view: axis of XYZ), and (d) is the 3D registration of deformed object 3 (view: axis of xz). Left: initial pose, center: registration with Go-ICP, right: registration with ICP alone.

ρ (or $\rho = 0$) might lead to a very large running time since the outliers prevent BnB to converge to a reasonable RMS error. A very small ρ (or ρ close to 0) does not remove any outliers that may be at a large distance from the model and all points must be considered in the optimization. In other words, due to the presence of outliers, the RMS error cannot converge to the selected threshold (ε). For example, in Fig. 4, considering $\rho = 10\%$, $\rho = 20\%$, and $\varepsilon = 10$ for Deformed object 1, 2 results in inaccurate registration, while using $\rho = 10\%$ and $\varepsilon = 10$ registers Deformed objects 1, 2 more accurately. Therefore, to get the best results from BnB and ICP, one should tune ρ for the specific data point set in order to remove outliers properly. In all the experiments, we find the best ρ factor that produces the best registration in a reasonable running time e.g. less than 10 minutes for most experiments except the ones with very large point clouds (with millions of points). The curves of the effect of the trimming factor on RMS error and computation time are shown in Fig. 5.

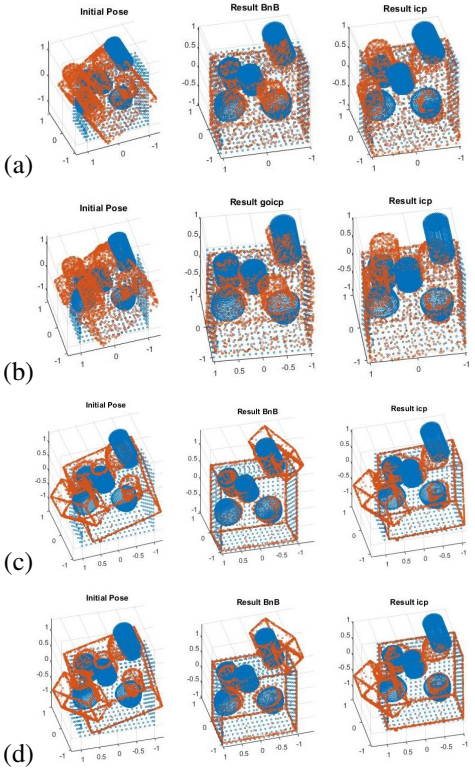


Figure 4: The effect of trimming on 3D registration of deformed objects 1 and 2. (a) is the result for $\rho = 10\%, \varepsilon = 10$ on deformed object 1, (b) is the result for $\rho = 20\%, \varepsilon = 10$ on deformed object 1, (c) is the result for $\rho = 10\%, \varepsilon = 10$ on deformed object 2, and (d) is the result for $\rho = 20\%, \varepsilon = 10$ on deformed object 2. Left: initial pose, center: registration with Go-ICP, right: registration with ICP alone.

The effect of registration error threshold ε

ISBN 978-80-86943-37-4

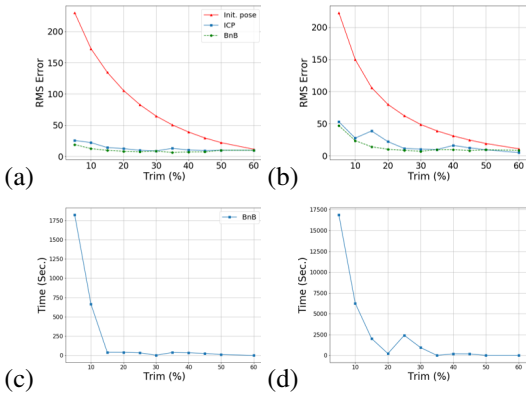


Figure 5: The curve of the impact of the trimming factor ρ on registration error and running time. The red curves are the initial RMS errors, the blue curves are the RMS errors of ICP, and the green curves are the RMS errors of Go-ICP. (a) is the curve of the effect of trimming on the 3D registration error of deformed object 1, (b) is the result for deformed object 2, (c) is the curve of the effect of trimming on execution time (seconds) of the 3D registration of deformed object 1, and (d) is execution time (seconds) for deformed object 2.

The accuracy (RMS error) and running time of BnB highly depend on the threshold (ε) as it defines the stopping criterion. In other words, BnB terminates its registration process when the optimal RMS error is close enough to the lower bound RMS error, i.e. $E^* - E_r < \varepsilon$ (E^* is the optimal RMS error and E_r is the lower bound RMS error of the domain of rotation space). Setting the threshold to a small value can register data points to model points more accurately but the running time is very high. On the other hand, a greater threshold can lead to a fast execution time, but the accuracy of registration decreases. It is worthwhile to note that the initial optimal RMS error of Go-ICP is computed by ICP. So, for large threshold values, Go-ICP can terminate very fast since its initial optimal RMS error can become lower than the specified threshold quickly. For example, the running time for Deformed objects 1, 2 with $\varepsilon = 7$ is only 1839.64 seconds for deformed object 1 and 4258.06 seconds for deformed object 2 since ICP as the first step of BnB achieves a RMS error of 0.685 which is lower than $\varepsilon = 5$ (Note that in this case the lower bound RMS error was zero) (Fig. 6). However, this RMS error achieved by ICP is not the global minimum. Therefore, we highlight that for reaching the global minimum of the objective function, one should set the threshold appropriately. The curve of the effect of the threshold ε on the RMS error and on the running time is shown in Fig. 7.

The effect of reducing the number of points (Nd) on the execution time of Go-ICP.

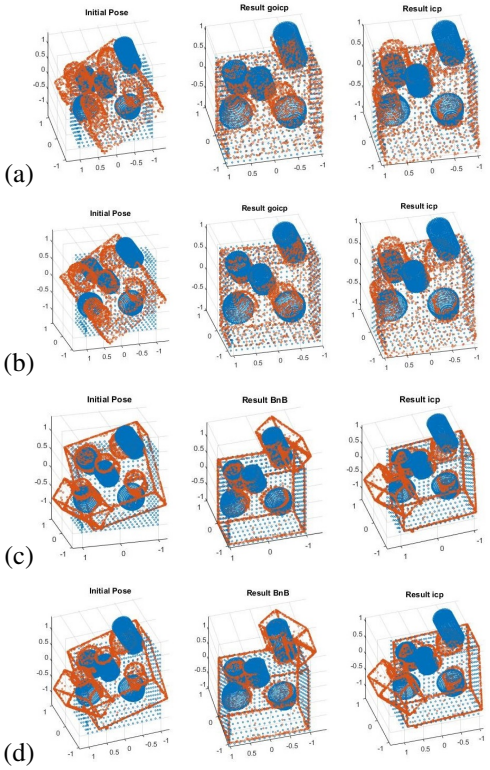


Figure 6: The effect of threshold ε on 3D registration for deformed objects 1 and 2. (a) is the result of $\varepsilon = 5, \rho = 10\%$ on deformed object 1, (b) is the result of $\varepsilon = 15, \rho = 10\%$ on deformed object 1, (c) is the result of $\varepsilon = 5, \rho = 10\%$ on deformed object 2, and (d) is the result of $\varepsilon = 15, \rho = 10\%$ on deformed object 2. Left: initial pose, center: registration with Go-ICP, right: registration with ICP alone.

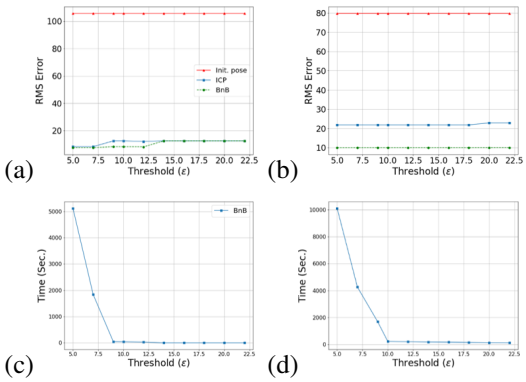


Figure 7: The curve of the effect of the threshold ε on registration error ((a) and (b)) and execution time in seconds ((c) and (d)). The red curves are the initial errors, the blue curves are the errors of ICP, and the green curves are the errors of Go-ICP. Effect of ε on the registration for deformed object 1 (a) and deformed object 2 (b). Effect of ε on execution time for deformed object 1 (c) and deformed object 2 (d).

We also studied how down-sampling the point clouds affect the registration achieved by go-ICP. We down-

ISBN 978-80-86943-37-4

sampled the original and model point clouds and repeated the registration experiments for deformed object 1, 2, and 3. Nd is the factor by which the number of points were reduced (i.e. $N_{down} = N_{total/Nd}$). As shown in fig. 8, the registration values (R, T) are the same for all values of Nd but the execution time varies. It is surprising to see that decreasing the number of points does not necessarily reduce the computation time specially for deformed object 3 in Fig. 8 (c).

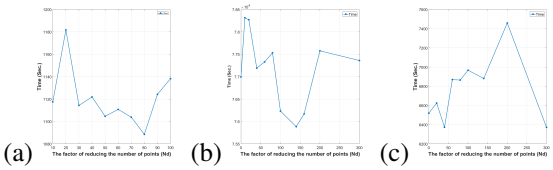


Figure 8: The effect of the down-sampling factor Nd on Go-ICP execution time. (a): deformed object 1. (b): deformed object 2. (c): deformed object 3.

3.2.3- Alignment of large-size point clouds using Go-ICP

In this section, we apply the Go-ICP approach to the alignment of large-scale point clouds and show that it can achieve good alignment.

We use the CAD model of a building with 387,564 data points (Fig. 9 (a)) and two real large-scale point clouds (inside point cloud with 8,497,325 data points and outside point cloud with 26,574,097 data points) captured by a Lidar (Fig. 9 (e, g)). As can be seen in Fig. 9 (b), the data points on the CAD model are not adequate for alignment because they are not distributed evenly which prevents Go-ICP to find the alignment of this CAD model with large-scale point clouds. Therefore, the CAD model is resampled using Blender to yield 330,327 data points evenly distributed on the inside and outside (Fig. 9 (c) and (d)). For the alignment, the large-scale point clouds were cropped to keep only the points belonging to interior or exterior of a building (Fig. 9 (f, h)). Consequently, the points on the ground were discarded.

Go-ICP was able to find the alignment between the resampled CAD model and the large-scale point cloud of the inside and outside of the building. For these 3D registration experiments, we considered $\rho = 20\%$ for trimming and $\varepsilon = 5$ as the registration error. The number of points of the cropped point cloud of the inside of the building is 6,297,579. The execution time of this registration is 7742.02 seconds for a RMS error of 6.7934. The ICP alone RMS error was 12.7098 (Fig. 10 (a)). Go-ICP estimates the rotation and translation parameters. The parameters are then used to align the original CAD model of the building with the cropped point cloud of the inside of the building. (Fig. 10 (c)).

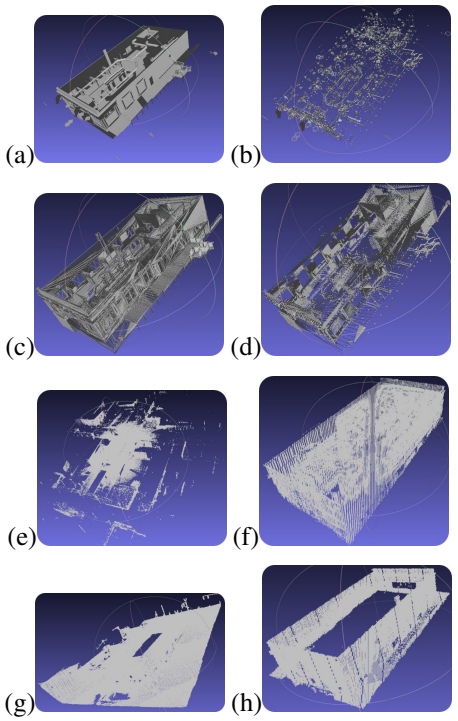


Figure 9: The point clouds of a building and its large-scale point clouds. (a, b) are the CAD model of the building (original CAD model) and its point cloud, (c,d) resampled of data points of the original CAD model with a more even spatial distribution of the points, (e, f) is the point cloud of the inside of the building and its cropped point cloud, (g, h) is the point cloud of the outside of the building and its cropped point cloud.

For the alignment of the resampled CAD model and the cropped point cloud of the outside of the building, we considered $\rho = 20\%$ for trimming and $\varepsilon = 10$ as the registration error. The number of points of the cropped point cloud of the outside of the building is 12,841,366. Go-ICP found the 3D registration shown in Fig. 11 (a) and (b). The execution time of this registration is 3559.15 seconds for a RMS error 13.9063. The ICP alone RMS error was 37.0139. The registration parameters were used to align the CAD model with the cropped point cloud of the outside of the building (Fig. 11 (c) and (d)) with Go-ICP. It is thus possible to compare the actual building ("as-built") with the CAD model of the design ("as-designed model").

In section 4, we use the Go-ICP for aligning the resampled CAD model of a building with a large-scale point cloud without any change (no cropping of the point cloud). Experiments on the alignment of man-made objects with large deformations are also presented.

ISBN 978-80-86943-37-4

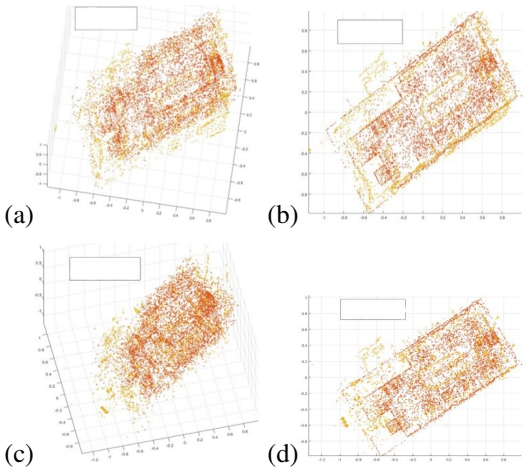


Figure 10: Result of the alignment of the resampled CAD model and the point cloud of the inside of the building: xyz view (a), xz view (b). Red: CAD model. Yellow: point cloud. The parameters of the registration are used to align the original CAD model with the point cloud: xyz view (c), xz view (d).

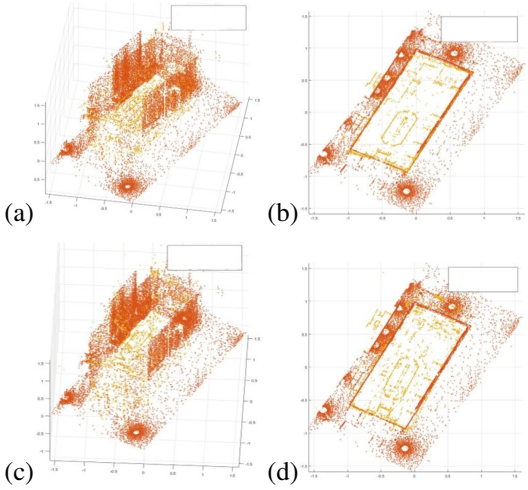


Figure 11: Result of the alignment of the resampled CAD model and the point cloud of the outside of the building: xyz view (a), xz view (b). Red: Cad model. Yellow: point cloud. The parameters of the registration are used to align the original CAD model with the point cloud: xyz view (c), xz view (d).

4 EXPERIMENTS ON THE ALIGNMENT OF THE POINT CLOUD OF DEFORMED OBJECTS WITH ITS ORIGINAL CAD MODEL

4.1 CAD model

In this section, we aim to align two deformed CAD model using Go-ICP. For this purpose, we consider the deformed CAD model 2 and the deformed CAD model that was described in Sec.3.2.1 (see Fig. 2 (d, e)). The 3D registration of these CAD models is difficult because of the large deformation. However, Go-ICP can find a good registration with $\rho = 20\%$ and $\varepsilon = 10$. Alignment results are shown in Fig. 12. In this case, the RMS error was 14.9752.

As an additional test, we used other CAD models with different shapes that represent alignment challenges (Fig. 13 (a, d)). In the first part of the test, we deformed the CAD model shown in Fig. 13 (a)(and called CAD model 1) with 3Ds Max and created a new CAD model (see Fig. 13 (b)). The challenge consists of the alignment of CAD model 1 and its deformed copy for quality control for instance. Go-ICP was able to find the 3D registration with $\rho = 10\%$ and $\varepsilon = 5$ with RMS error 4501.87 in 965.276 seconds. The result of registration is shown in Fig. 13 (d). Such an alignment can be used in metrology to compare objects with a deformed copy of themselves to assess whether the fabrication is reliable (the deformation is reflected by the large value of the RMS error).

In the second part of this test, we perform the 3D registration between CAD model 1 with the CAD model shown in Fig. 13 (e). This registration is difficult since the two objects are different but it may be interesting to compare them in a metrology application or for visualizing the objects in a virtual reality application. Go-ICP succeeded in aligning these different objects with $\rho = 10\%$, $\varepsilon = 5$, RMS error 2051.108, and an execution time of 324.502 seconds. The result is shown in Fig. 13 (g).

A final test using Go-ICP for the alignment of an object and a deformed copy of itself is shown in Fig. 14. Fig 14 (a) shows a mechanical part and a deformed copy in Fig. 14 (b). Fig. 14 (c) shows the alignment obtained with Go-ICP with $\rho = 10\%$ and $\varepsilon = 5$ with a RMS error of 227.987 in 32.119 seconds.

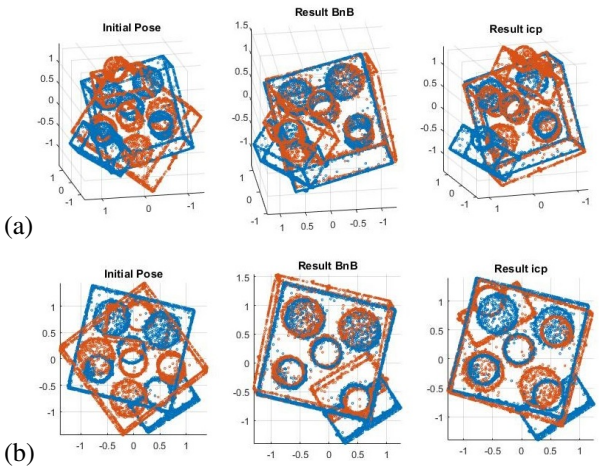


Figure 12: Result of 3D registration of deformed CAD model 2 and deformed CAD model 3. (a) XYZ axis, (b) XZ axis.

ISBN 978-80-86943-37-4

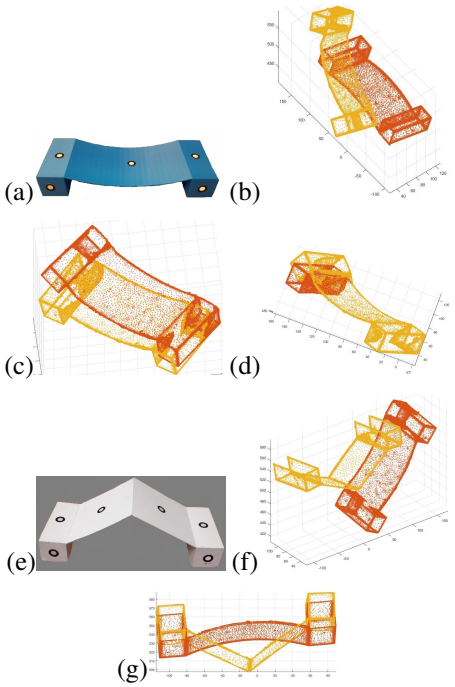


Figure 13: The results of 3D registration of the CAD models. (a) is the CAD model 1, (b) is the initial pose and point clouds of CAD model 1 and its deformed copy, (c) is the ICP alone result of the alignment of the CAD model 1 and its deformed copy, (d) is the result of the alignment of the CAD model 1 and its deformed copy, (e) is CAD model 2, (f) is the initial pose and point clouds of CAD model 1 and CAD model 2, (g) is the result of the alignment of CAD model 1 and CAD model 2.

4.2 Alignment of Large-Scale Point Clouds

The most challenging 3D registration experiment on which Go-ICP was tested is the alignment of very large-scale point clouds. For this experiment, we consider the CAD model of a building with 42,609 points and a very large-scale point cloud from the area of the original building with 1,236,922 points that was captured by a Lidar. Fig. 15 (a) shows the CAD model of the building and Fig. 15 (b, c) show the area of the building and its surroundings and the corresponding point cloud.

The goal is to align the CAD model of the building to the large-scale point cloud of the building and its surrounding area without any cropping. This makes the registration problem very difficult. We divided this experiment into two steps. In the first step, the points not belonging to the building were cropped and the CAD model was aligned with the remaining points (i.e. the belonging to the building). It is still a challenge to find a good alignment. Nevertheless, Go-ICP has found the 3D registration with $\rho = 20\%$, $\varepsilon = 20$, and RMS error

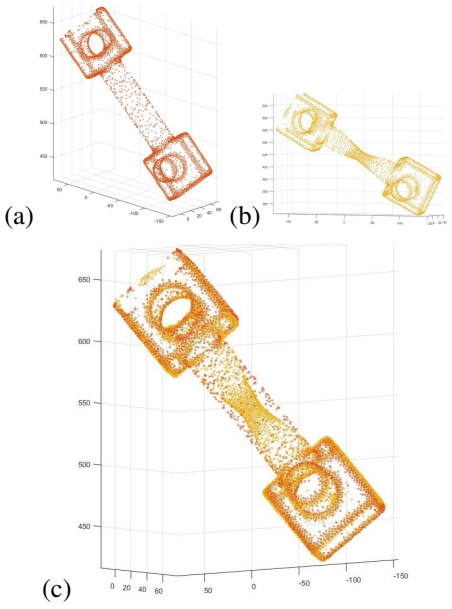


Figure 14: The results of 3D registration of mechanical parts. (a) point cloud of the part 1, (b) point cloud of deformed part 1, (c) is the result of the alignment of the part 1 and the deformed copy of itself 2.

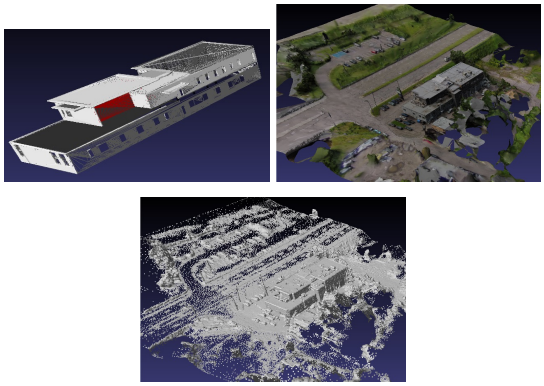


Figure 15: The CAD model of the building (a), the real area of the building (b), the large-scale point cloud of area (c).

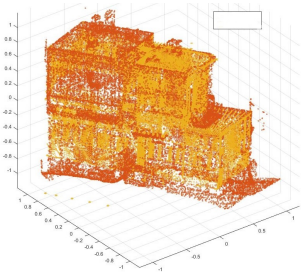
of 20.4129 in 48107 seconds. The result is shown in Fig. 16 (a): xyz view and (b): xz view.

In the second step, we aimed to align the CAD model to the point cloud of the building and its surrounding without any cropping. This is a significant challenge because of the number of points involved and the many local minimums of the cost function. Nevertheless Go-ICP is able to achieve the alignment with $\rho = 20\%$ and $\varepsilon = 60$ in 186,359 seconds and with a RMS error of 61.2573. Fig. 16 (c) shows the result of this 3D registration.

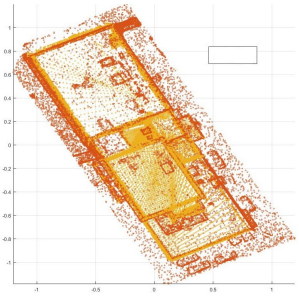
5 CONCLUSIONS

The Go-ICP algorithm was exploited to find the registration between a large point cloud and a deformed

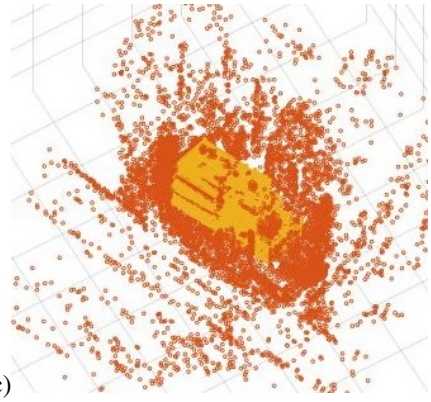
ISBN 978-80-86943-37-4



(a, view: xyz)



(b, view: xz)



(c)

Figure 16: The results of the alignment between the CAD model and very large-scale point cloud. Red points: CAD model, yellow point: building and its area. (a, b) are the 3D registration of the surrounding CAD model and the original building point cloud, (c) is the result of the registration of the CAD model and the point cloud of the building and its surrounding area without cropping.

copy itself or with a CAD model. Experiments show that with a proper selection of the algorithm's hyperparameters, good alignment can be achieved. For very large point clouds, the computation time needed for achieving a good alignment is important. We are currently using Go-ICP to find the the initial alignment between the CAD models of civil engineering infrastructures(building) and point clouds collected during its construction. This initial alignment is used in an augmented reality application providing architects ways of assessing whether or not what is being built ("as-built") corresponds with what has been designed by architects ("as-designed" model). The paper also demonstrates that Go-ICP can be exploited successfully for comparing man-made objects with deformed copies of themselves for the purpose of quality control in metrology applications.

6 ACKNOWLEDGMENTS

The authors acknowledge the support of NSERC-Bentley CRD grant number 474640-14.

ISBN 978-80-86943-37-4

7 REFERENCES

[Aiger et al., 2008] Aiger, D., Mitra, N. J., and Cohen-Or, D. (2008). 4-points congruent sets for robust pairwise surface registration. In *ACM Transactions on Graphics (TOG)*, volume 27, page 85. Acm.

[Belongie et al., 2002] Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (4):509–522.

[Besl and McKay, 1992] Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics.

[Blais and Levine, 1995] Blais, G. and Levine, M. D. (1995). Registering multiview range data to create 3d computer objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):820–824.

[Breuel, 2003] Breuel, T. M. (2003). Implementation techniques for geometric branch-and-bound matching methods. *Computer Vision and Image Understanding*, 90(3):258–294.

[Gelfand et al., 2005] Gelfand, N., Mitra, N. J., Guibas, L. J., and Pottmann, H. (2005). Robust global registration. In *Symposium on geometry processing*, volume 2, page 5. Vienna, Austria.

[Irani and Raghavan, 1999] Irani, S. and Raghavan, P. (1999). Combinatorial and experimental results for randomized point matching algorithms. *Computational Geometry*, 12(1-2):17–31.

[Johnson and Hebert, 1999] Johnson, A. E. and Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449.

[Li and Hartley, 2007] Li, H. and Hartley, R. (2007). The 3d-3d registration problem revisited. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE.

[Makadia et al., 2006] Makadia, A., Patterson, A., and Daniilidis, K. (2006). Fully automatic registration of 3d point clouds. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 1297–1304. IEEE.

[Mellado et al., 2014] Mellado, N., Aiger, D., and Mitra, N. J. (2014). Super 4pcs fast global pointcloud registration via smart indexing. In *Computer Graphics Forum*, volume 33, pages 205–215. Wiley Online Library.

[Olsson et al., 2009] Olsson, C., Kahl, F., and Oskarsson, M. (2009). Branch-and-bound methods for euclidean registration problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):783–794.

[Papazov and Burschka, 2011] Papazov, C. and Burschka, D. (2011). Stochastic global optimization for robust point set registration. *Computer Vision and Image Understanding*, 115(12):1598–1609.

[Parra Bustos et al., 2014] Parra Bustos, A., Chin, T.-J., and Suter, D. (2014). Fast rotation search with stereographic projections for 3d registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3930–3937.

[Pfeuffer et al., 2012] Pfeuffer, F., Stiglmayr, M., and Klamroth, K. (2012). Discrete and geometric branch and bound algorithms for medical image registration. *Annals of Operations Research*, 196(1):737–765.

[Rusu et al., 2009] Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217. IEEE.

[Sandhu et al., 2010] Sandhu, R., Dambreville, S., and Tannenbaum, A. (2010). Point set registration via particle filtering and stochastic dynamics. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1459–1473.

[Silva et al., 2005] Silva, L., Bellon, O. R. P., and Boyer, K. L. (2005). Precision range image registration using a robust surface interpenetration measure and enhanced genetic algorithms. *IEEE transactions on pattern analysis and machine intelligence*, 27(5):762–776.

[Yang et al., 2016] Yang, J., Li, H., Campbell, D., and Jia, Y. (2016). Go-icp: A globally optimal solution to 3d icp point-set registration. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2241–2254.

[Zhou et al., 2016] Zhou, Q.-Y., Park, J., and Koltun, V. (2016). Fast global registration. In *European Conference on Computer Vision*, pages 766–782. Springer.

 ISBN 978-80-86943-37-4

ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

3D Gaussian Descriptor for Video-based Person Re-Identification

Chirine Riachy[1], Noor Al-Maadeed[2], Daniel Organisciak[1], Fouad Khelifi[1], Ahmed Bouridane[1]
[1]Northumbria University, Newcastle Upon Tyne, UK
[2]Qatar University, Doha, Qatar
{chirine.riachy, daniel.organisciak, fouad.khelifi,
ahmed.bouridane}@northumbria.ac.uk
n.alali@qu.edu.qa

ABSTRACT

Despite being often considered less challenging than image-based person re-identification (re-id), video-based person re-id is still appealing as it mimics a more realistic scenario owing to the availability of pedestrian sequences from surveillance cameras. In order to exploit the temporal information provided, a number of feature extraction methods have been proposed. Although the features could be equally learned at a significantly higher computational cost, the scarce nature of labelled re-id datasets encourages the development of robust hand-crafted feature representations as an efficient alternative, especially when novel distance metrics or multi-shot ranking algorithms are to be validated. This paper presents a novel hand-crafted feature representation for video-based person re-id based on a 3-dimensional hierarchical Gaussian descriptor. Compared to similar approaches, the proposed descriptor (i) does not require any walking cycle extraction, hence avoiding the complexity of this task, (ii) can be easily fed into off-shelf learned distance metrics, (iii) and consistently achieves superior performance regardless of the matching method adopted. The performance of the proposed method was validated on PRID2011 and iLIDS-VID datasets outperforming similar methods on both benchmarks.

Keywords

Person Re-identification, Spatio-temporal Descriptor, Feature Extraction, Gaussian Distribution, Surveillance.

1 INTRODUCTION

With the rise in the need for smart surveillance applications, person re-identification (re-id) has attracted the attention of many researchers within the computer vision community. The problem entails finding a match for a given person image or sequence of images (the probe) among a set of gallery person instances captured using a different non-overlapping camera view. The aim is to track the person under a multi-camera setting. The challenges associated reside mainly in the large intra-class variations caused by significant changes in viewpoint angle, pose and illumination, added to the presence of background clutter and occlusions [1, 2, 3]. Furthermore, small inter-class variations caused by clothes similarities between different people render the task even more challenging.

To mitigate the effect of these challenging attributes on re-id performance, one could exploit the rich visual cues and temporal information provided by person video sequences available from surveillance cameras. For this purpose, the past few years have witnessed the development of several video person re-id algorithms. As few of the descriptors were specially designed for the video-based problem, most of them rely on image-based low-level representations to develop multi-shot ranking algorithms based on set-to-set distance measures [4, 5, 6], or to perform frame selection and weighting [7, 8]. When it comes to descriptors utilising spatio-temporal information, they are often accompanied by walking cycle extraction [9, 10], which is not trivial especially under severe noise and occlusions.

A robust hand-crafted spatio-temporal descriptor that can be efficiently computed and used for matching with common distance metrics has not been yet proposed, which motivates the current work. Such a descriptor presents the advantage of considerably boosting metric learning or multi-shot ranking accuracy, while being equally suited for use in unsupervised settings as no learning is required.

Benefitting from the advances achieved in image-based person re-id feature design, the state-of-the-art Gaussian of Gaussian (GOG) [11] feature is here extended to 3 dimensions integrating temporal information. The suggested extension coupled with existing metric learning approaches achieved significant accuracy improve-

ment on two widely tested video-based person re-id benchmarks, PRID2011 and iLIDS-VID.

Briefly, this paper proposes a robust person descriptor for video-based re-id leveraging both spatial and temporal cues. The proposed method is based on local Gaussian distributions of 3-dimensional pixel features that are subsequently projected into the Euclidean space. This allows the extracted feature to be learning-free and flexible to use with any matching method.

The remainder of the paper is organised as follows. Section 2 highlights recent related work. Section 3 explains the proposed approach. The experiments conducted are described in detail in Section 4. Finally, the paper is concluded in Section 5.

2 RELATED WORK

Tracking a person in a multi-camera setting involves three main steps: person detection, tracking, and retrieval. The latter consists of searching a person captured in one view, in a different non-overlapping camera view. It is commonly known as person re-id. For that purpose, two tasks should be fulfilled: pedestrian description and matching. The former involves representing persons by a set of features describing their physical appearance. The most popular have been colour and texture features [11, 12, 13, 14, 15, 16, 17]. On the other hand, distance metric learning has emerged as the most prevalent matching method [15, 6, 14, 18, 19] due to its efficiency and promising accuracy.

Early research in person re-id focused mostly on the single-shot scenario where each person is represented by a pair of images that need to be matched [17, 20, 21]. Other scenarios were also investigated including multi-shot re-id and video-based re-id [4, 5, 6, 9, 10, 22]. In that case, each subject is represented by multiple images or video sequences in both probe and gallery views.

Hand-crafted features have been predominant in person re-id until very recently when deep-learning popularity started growing [22, 23, 24, 25]. This was mainly triggered by the release of large-scale datasets such as CUHK03 [24], Market-1501 [26], and MARS [25]. A common practice for hand-crafted methods is to divide the person image into small patches and several horizontal stripes on which features are subsequently extracted. The most prominent methods in this category are ELF [20], gBiCov [12], HistLBP [14], DenseColorSIFT [27], and LOMO [15]. They all leverage colour and texture information to describe the person's appearance.

More recently, a high-performing image person descriptor called GOG [11] has been proposed. It divides the image into small overlapping patches and a number of horizontal stripes. It then leverages both mean and covariance information by encoding each patch using a Gaussian distribution. Towards the goal, each pixel i is initially described by a feature vector p_i given by $p_i = [y, M_{0^\circ}, M_{90^\circ}, M_{180^\circ}, M_{270^\circ}, R, G, B]^T$ where y is the y-coordinate of pixel i, M_{0° to M_{270° are the orientations along which the gradient is quantised and multiplied by the gradient magnitude, and R, G, B are the RGB colour channels. Patches belonging to the same horizontal stripe are in turn summarised using another Gaussian distribution that is flattened into the Euclidean space. The final image representation is the concatenation of all stripes' features.

As for the matching process, distance metric learning attempts to find a subspace that brings positive samples (feature vectors of the same person) closer to each other while pushing negative samples apart. Various methods were proposed in this category of which the most prominent are KISSME [18], Cross-view Quadratic Discriminant Analysis (XQDA) [15], and kernelised versions of Local Fisher Discriminant Analysis (kLFDA) and Marginal Fisher Analysis (kMFA) [14]. More recently, Zhang *et al.* [19] exploited the kernel Null Foley-Sammon Transform (kNFST) to learn a subspace where the within-scatter is zero and the between-scatter is positive. By enforcing such a strict condition, the learned subspace is more discriminative and exhibits better separability of the data. Distance metric learning methods have been a great success in person re-id given their efficiency and accuracy especially on small datasets where deep learning usually fails. These reasons motivate their use in this work to additionally boost the performance of the proposed person descriptor.

Compared to image-based re-id, video-based re-id represents a more intuitive scenario owing to the availability of pedestrian videos from surveillance cameras. The early trend was to treat the task as a multi-shot matching problem. That is, features were extracted from still images on which the person appearance was built, and temporal cues were completely ignored [5, 21]. This was compensated by exploiting the multiple person instances in the matching process. More recently, Wang *et al.* [9] leveraged temporal information by proposing a spatio-temporal descriptor based on HOG3D features [28], they later combined it with mean colour values [29]. These features were computed on fragments extracted from the person sequence using the Flow Energy Profile (FEP) signal as an approximation of walking cycles. A ranking function, DVR, was learned for matching. Liu *et al.* [10] subsequently proposed a spatio-temporal descriptor based on Fisher vectors extracted from video-fragments representing body-action units after performing walking cycle extraction similarly to [9].

 ISBN 978-80-86943-37-4

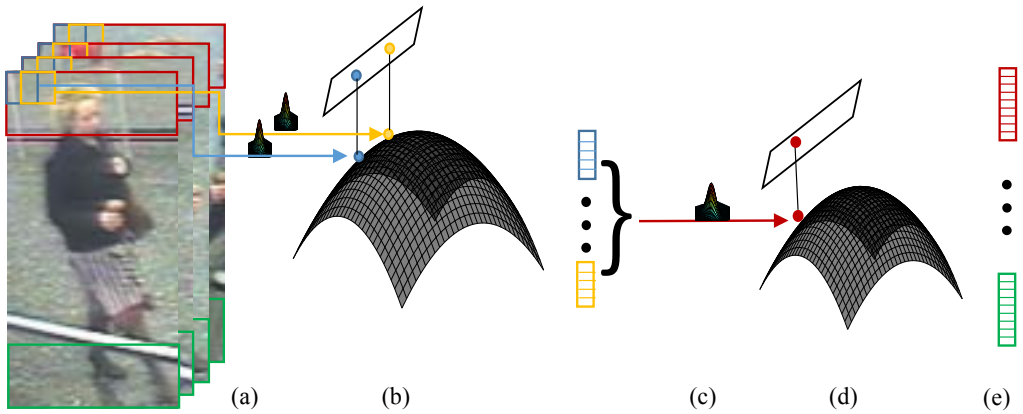


Figure 1: Representative diagram of GOG3D feature. (a) Patch Gaussians are computed. (b) They are then flattened into the Euclidean space. (c) Region Gaussians are formed using patches in the same horizontal region. (d) They are also projected into the Euclidean space. (e) Region feature vectors are concatenated to form the final image feature. Finally, average-pooling is performed over image features of the same person.

Following these works, temporal information was mainly exploited in deep learning methods such as RNN [22] or ASTPN [23] where spatial and temporal pooling layers were added for that purpose. The remaining video-based re-id systems either design a set-to-set distance metric such as DRAH [4] that models the distance between two sequences as the minimum distance between their respective affine hulls, or assign a signature to each person using the corresponding frames by fitting a GMM [30] for instance.

The proposed method is a spatio-temporal descriptor that benefits from temporal correlation to give a richer representation for each person. However, each person sequence is eventually described by a single feature vector which is in essence similar to the single-shot scenario. This is very convenient as it speeds up the matching process significantly and broadens the applicability of the descriptor with any off-shelf distance metric. It is worth noting here that extracting the proposed feature from sampled walking cycles is also possible. However, in this work we highlight the advantages presented by using it in its simplest setting.

3 PROPOSED METHOD

This section details the proposed spatio-temporal descriptor for video-based person re-id which we call GOG3D. Similar to GOG [11], a part-based model is adopted where each person image is divided into R overlapping horizontal stripes, roughly representing different human body-parts. Each stripe is also divided into small overlapping patches, and each patch is modelled by a Gaussian distribution using its pixel feature information. Patch Gaussians are subsequently embedded into the space of Symmetric Positive Definite (SPD) matrices and flattened into the Euclidean space forming the patch feature vector. Patches belonging to the same horizontal region are in turn summarised by a Gaussian that is projected into the Euclidean space forming the region feature vector. Region feature vectors are concatenated to form the final image feature. Finally, image features of the same person are averaged to form the final representation of a person sequence. A diagram summarising GOG3D is shown in Fig. 1.

3.1 Pixel Features

By first considering each of the patches, local information is described using a 10-dimensional pixel feature vector summarising the spatial position of the pixel, its gradient magnitudes along four directions, and the intensity values of some colour channels. Namely, for a pixel i, the pixel feature vector p_i is given by:

$$p_i = [x, y, M_{0^\circ}, M_{90^\circ}, M_{180^\circ}, M_{270^\circ}, |I_t|, L, A, B]^T, \quad (1)$$

where x and y are the x- and y-coordinates of pixel i taken from the top-left of the image, M_{0° through M_{270° are the orientations along which the gradient is quantised and multiplied by its magnitude, $|I_t|$ is the gradient magnitude in the temporal direction, and finally L, A, B correspond to the Lab colour channels. Each dimension of p_i is scaled to the range [0, 1] before further processing can take place. The computation of these pixel features is detailed in the following.

For simplicity, let us consider one person sequence of images given by $S = \{Q_k | k = 1, ..., N\}$, where N is the number of frames in this video sequence. By taking a pixel $i(x, y, t)$ in frame Q_t, the gradients I_x, I_y and I_t in the horizontal, vertical and temporal directions can be computed as:

$$I_x(x, y, t) = i(x+1, y, t) - i(x-1, y, t), \quad (2)$$

ISBN 978-80-86943-37-4

Figure 2: The top row represents example images sampled from person sequences of iLIDS-VID dataset, each 2 adjacent images represent a correct match. The bottom row includes the temporal gradient computed for these sequences and averaged over the frames involved.

$$I_y(x,y,t) = i(x,y+1,t) - i(x,y-1,t), \quad (3)$$

$$I_t(x,y,t) = i(x,y,t+1) - i(x,y,t-1), \quad (4)$$

where x, y and t are the x-, y- and t-coordinates of pixel $i(x,y,t)$, respectively.

Although, it is possible to compute the gradient orientation in 3D and quantise it using a regular polyhedron in a manner similar to HOG3D in spirit [28], this is suboptimal in our case due to the following reasons. Firstly, binning in 3D while preserving the distinctive power of the descriptor requires the use of a dodecahedron (12 bins) or icosahedron (20 bins) [28], thus raising significantly the dimensionality of the pixel feature vector p_i. A high-dimensional p_i will definitely cause numerical problems upon the computation of covariance matrices in small patches. Moreover, it is favourable to use spatial gradients separately as a texture descriptor, while motion information is encoded via the temporal gradient. For this purpose, hereby the orientation of the spatial gradient given by I_x and I_y is exploited separately to the temporal gradient I_t.

The gradient orientation O is given by $O = \arctan(I_y/I_x)$ and its magnitude is defined as $M = \sqrt{(I_x^2 + I_y^2)}$. It has been proved in [31] that quantisation into vector angles rather than using magnitude and orientation raw values is essential to enhance the discriminative power of the descriptor. Therefore, soft voting is used to quantise the values of O into two neighbouring bins to account for the loss of information caused by quantisation while maintaining some rotation invariance. Since four bins are considered in this case, the reference points are 0°, 90°, 180°, and 270°. Given $\alpha \leq O < \beta$ where $\alpha, \beta \in \{0°, 90°, 180°, 270°\}$ are the boundaries of the bin containing O, the distances (positive differences) $d_\alpha = |O - \alpha|$ and $d_\beta = |O - \beta|$ from O to the bin boundaries are computed, and the voting weights are assigned as $w_\alpha = d_\beta/(d_\alpha + d_\beta)$, $w_\beta = d_\alpha/(d_\alpha + d_\beta)$ and $w_{(\theta \neq \alpha,\beta)} = 0$. These weights are finally multiplied by the gradient magnitude M to obtain $M_\theta, \theta \in \{0°, 90°, 180°, 270°\}$. As for the temporal gradient I_t, the magnitude of I_t is found by taking its absolute value $|I_t|$. Some examples of the information provided by computing $|I_t|$ for all sequence images and taking their average can be seen in Fig. 2. This highlights the type of information added by computing the gradient in the temporal direction.

The choice of the colour channels is crucial for any appearance descriptor, especially for person re-id where individuals are mainly distinguished by their clothes' colours and texture. Performing feature fusion by extracting the features four times, each with different colour channels such as RGB, HSV, Lab and normalised RGB (nRGB), and concatenating them similarly to $\text{GOG}_{\text{fusion}}$ [11] and moM [16], is highly inefficient. The reason is that the features have to be extracted four times for each person, and the resulting vector has a high dimension which in turn slows down the matching process. For this purpose, it is convenient to select the most discriminative colour channels that can better deal with illumination changes. In this work, these are found to be the Lab colour channels.

It has been previously argued that person images are aligned vertically but not horizontally [13], therefore only the y-coordinate has been used in GOG algorithm. However, as it will be detailed thereafter, we find that the orderless representation of patches in the same horizontal stripe as a mean to deal with viewpoint angle variations may cause the loss of some important spatial information that could be useful for re-identification.

 ISBN 978-80-86943-37-4

This could be avoided by including the horizontal location of the pixel represented by its x-coordinate.

3.2 Patch and Region Gaussians

As both mean and covariance features have proved successful in person re-id [9, 12], a promising way to leverage both types of information is to summarise them using a Gaussian distribution. As discussed in [11], it is definitely possible to use a Gaussian Mixture Model (GMM) instead for a more accurate representation. However, given the small patch size, a simple Gaussian model should be sufficient to describe the patches. More importantly, unimodal Gaussians can be efficiently projected into the Euclidean space which renders the matching process with the resulting feature much easier, as any off-shelf distance metric can thus be exploited. Therefore, for each patch H, the mean μ_H and covariance Σ_H are estimated as $\mu_H = \frac{1}{n_H}\sum_{i\in H} p_i$ and $\Sigma_H = \frac{1}{n_H-1}\sum_{i\in H}(p_i-\mu_H)(p_i-\mu_H)^T$ where n_H is the number of pixels in patch H and p_i is the feature vector of pixel i defined in Section 3.1. Subsequently, the patch Gaussian $\mathcal{N}(p;\mu_H,\Sigma_H)$ is given by:

$$\mathcal{N}(p;\mu_H,\Sigma_H) = \frac{\exp\left(-\frac{1}{2}(p-\mu_H)^T\Sigma_H^{-1}(p-\mu_H)\right)}{(2\pi)^{d/2}|\Sigma_H|}, \tag{5}$$

where $|\cdot|$ is the matrix determinant operator and d is the dimension of pixel feature vector p.

Once all patch Gaussians are computed, an algorithm (see Section 3.3) is used to project these Gaussians into the Euclidean space transforming them into patch feature vectors f_H. To account for background clutter, patches are weighted similarly to SDALF algorithm [17] according to their distance from the central vertical axis of the image such that $w_H = \exp\left(-\frac{(x_H-x_C)^2}{2\sigma^2}\right)$ where $x_C = W/2$, $\sigma = W/4$ and x_H is the x-coordinate of the central pixel in patch H. By W we denote the image width. According to this definition, it is easy to see that more weight is assigned to patches closer to the central vertical axis of the image where the person is expected to be centred.

In a similar manner to patch Gaussian computation, the patches belonging to the same horizontal region are in turn summarised into a region Gaussian using their mean and covariance information. Based on the above defined weights, the region mean μ_R and covariance Σ_R are defined for region R as:

$$\mu_R = \frac{1}{\sum_{H\in R} w_H}\sum_{H\in R} w_h f_h, \tag{6}$$

$$\Sigma_R = \frac{1}{\sum_{H\in R} w_H}\sum_{H\in R} w_H(f_H-\mu_R)(f_H-\mu_R)^T, \tag{7}$$

where f_H is the patch feature vector for patch H. The region Gaussians are consequently computed according to 5 and projected into the Euclidean space before concatenation, to form the final representation of an image. Finally, frame-wise features are averaged over a person's sequence to form the final representation of that sequence. It is also worth noting that covariance matrices are regularised by adding a small value ε to diagonal entries to prevent them from becoming singular.

3.3 Euclidean Space Projection

Projecting patch and region Gaussians into the Euclidean space is essential for GOG3D, primarily to obtain a final descriptor that can be used with off-shelf distance metrics. Towards the goal, the following two steps need to be applied.

It is well known that multivariate Gaussian distributions lie on a Riemannian manifold that can be embedded into the space of SPD matrices [32]. Such embedding is favoured as the SPD space endowed with the log-Euclidean metric can be locally flattened into the tangent Euclidean space through matrix logarithm. More specifically, consider a d-dimensional multivariate Gaussian $\mathcal{N}(\mu_H,\Sigma_H)$, this can be embedded into a $(d+1)$-dimensional SPD matrix P_H as follows:

$$\mathcal{N}(p;\mu_H,\Sigma_H) \sim P_H = |\Sigma_H|^{-\frac{1}{d+1}}\begin{bmatrix}\Sigma_H+\mu_H\mu_H^T & \mu_H\\ \mu_H^T & 1\end{bmatrix}. \tag{8}$$

P_H can subsequently be mapped into the Euclidean tangent space by computing $\Gamma_H = \log(P_H)$ where $\log(\cdot)$ is the matrix logarithm operator. Noting that Γ_H is symmetric, only the upper triangular part needs to be stored resulting in the final vector f_H being $m = (d^2+3d)/2+1$ dimensional. Thus, $f_H = \text{vec}(\Gamma_H) = [\Gamma(1,1),\sqrt{2}\Gamma(1,2),...,\sqrt{2}\Gamma(1,d+1),\Gamma(2,2),\sqrt{2}\Gamma(2,3),...,\Gamma(d+1,d+1)]$. Note that off-diagonal entries are multiplied by $\sqrt{2}$ upon half-vectorisation to ensure that the Frobenius norm of Γ_H remains equal to the ℓ_2-norm of f_H, that is $||\Gamma_H||_F = ||f_H||_2$.

4 EXPERIMENTS

4.1 Datasets

The proposed GOG3D feature is evaluated on the two most widely tested benchmarks for video-based person re-id, PRID2011 and iLIDS-VID.

iLIDS-VID [9] includes 600 person sequences created from two non-overlapping camera views captured by a CCTV network in an airport arrival hall. 300 different subjects are sampled in this dataset with two sequences per person. Sequences have variable lengths ranging from 23 to 192 frames with an average number of 73. iLIDS-VID is a very challenging dataset in terms of illumination changes, viewpoint angle variations and occlusions.

ISBN 978-80-86943-37-4

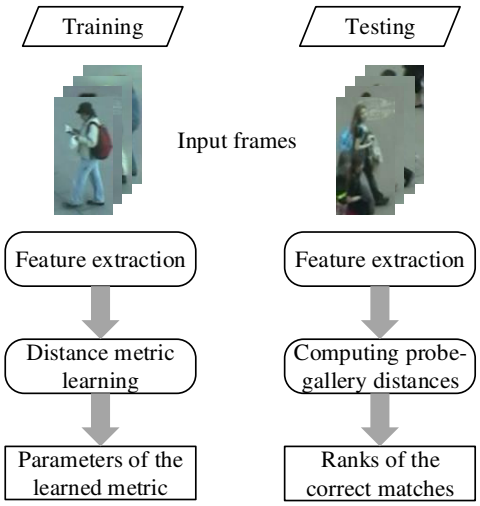


Figure 3: General pipeline of the re-id system employed in this work.

PRID2011 [33] consists of 400 image sequences of 200 different subjects captured by two adjacent cameras. Sequences lengths vary from 5 to 675 frames with an average number of 100 frames per sequence. PRID2011 is less challenging than iLIDS-VID as no occlusions or background clutter are involved. However, significant changes in illumination are exhibited.

4.2 Implementation Details

The original GOG algorithm divides the image into 7 overlapping horizontal regions and employs a patch size of 5×5 pixels. Patches are extracted at 2 pixels interval, and the regularisation parameter is $\varepsilon = 0.001$. For the first experiment reported in Section 4.3, and for fair comparison with GOG, a similar setting is used for GOG3D. This results in the patch feature vector being of size $(102+3\times10)/2+1 = 66$, and the region feature vector being $(662+3\times66)/2+1 = 2,278$ dimensional, finally the person descriptor obtained is $7 \times 2,278 = 15,945$ dimensional. When comparing to state of the art (Section 4.4), the parameters of GOG3D are further tuned through extensive experimentation which results in a better performance for a patch size of 9×9 pixels and $\varepsilon = 0.0001$. The patch extraction interval is kept at 2 pixels, and the results obtained on both datasets are presented using 10 and 15 overlapping horizontal regions denoted GOG3D^{10} and GOG3D^{15}, respectively. In that case, the resulting feature vector is 22,780 and 34,170 dimensional for $R = 10$ and $R = 15$, respectively.

Similar to GOG, mean removal and ℓ_2-normalisation are applied. Dimension reduction using PCA is performed with KISSME metric [18], and the dimension of the reduced feature is set to 100. A linear kernel is used with kNFST [19], kLFDA [14], and kMFA [14] in all experiments. The code provided by the authors is employed for GOG, and frame-wise features are also averaged for a person sequence in a similar manner to proposed GOG3D. The performance is evaluated using the Cumulative Matching Characteristic (CMC) curve as follows. Given a probe (query) instance, the gallery elements are ranked according to their distance from the probe, and at each given rank of the CMC curve, the probability of the correct match appearing at a similar or higher rank is computed. The general pipeline of the re-id system employed in this work can be seen in Fig. 3.

4.3 Components Analysis and Comparison to GOG

To highlight the consistent performance gain by employing GOG3D over GOG algorithm for video-based re-id, we compare the results in top-matching rates of the CMC curve using five state-of-the-art metric learning methods: XQDA [15], KISSME [18], NFST [19], kLFDA and kMFA [14]. For this purpose, each dataset was divided into two subsets, half for training and half for testing. The experiments were repeated over 10 trials and the average results are reported.

Table 1 shows the performance obtained by adding different pixel features to baseline GOG. Specifically, GOG_I$_t$ is obtained by adding the $|I_t|$ component, GOG_I$_t$_x also involves the x-coordinate, and GOG3D is the final feature obtained by replacing RGB colour channels by the Lab channels. A detailed evaluation of these components is performed using XQDA distance metric since the latter was initially used with GOG in [11]. Moreover, results comparing GOG directly to GOG3D with four other metrics can be seen in Table 2 for further validation.

It is clear from these results that GOG3D presents a remarkable advantage over GOG for video-based person re-id. Using XQDA, the performance gain was gradual and consistent by adding different pixel feature components to reach a maximum of almost 7% for iLIDS-VID and 3% for PRID2011 from baseline GOG to GOG3D. This margin undergoes some fluctuations when employing different metrics. It reaches a maximum of almost 13% with kLFDA on iLIDS-VID and 2% on PRID2011 with most other metrics.

4.4 Comparison to State of the Art

Since available spatio-temporal descriptors are mostly designed to be applied to extracted walking cycles or fragments sampled from person sequences [9, 10], employing them directly with a common distance metric in the same evaluation protocol to ours is unfair. When possible, it will in fact cause their performance to downgrade. Therefore, video-based re-id systems were

 ISBN 978-80-86943-37-4

Dataset	iLIDS-VID			PRID2011		
Rank	1	5	20	1	5	20
GOG + XQDA	66.6	87.2	96.9	86.6	97.6	99.6
GOG_I_t + XQDA	70.6	90.6	98	87.2	97.2	99.8
GOG_I_t_x + XQDA	72.9	91.3	**98.7**	87.7	97.8	99.9
GOG3D + XQDA	**73.7**	**92**	98.3	**89.9**	**97.9**	**100**

Table 1: Components analysis of GOG3D. Best results in top-matching rates are in bold.

Dataset	iLIDS-VID			PRID2011		
Rank	1	5	20	1	5	20
GOG + KISSME	49.5	73.4	88.1	81	94.2	99.2
GOG3D + KISSME	**55.1**	**78.9**	**92.7**	**83.3**	**94.9**	**99.2**
GOG + kNFST	63.9	85.9	95	87.8	97.4	99.9
GOG3D + kNFST	**74.3**	**92.2**	**98.9**	**89.6**	**97.8**	**100**
GOG + kLFDA	54.6	85.5	97.5	82.8	**96.8**	99.6
GOG3D + kLFDA	**67.4**	**90.7**	**98.8**	**85.1**	96.5	**99.9**
GOG + kMFA	56.3	85.3	97.6	83.3	**96.7**	99.5
GOG3D + kMFA	**66.5**	**90.9**	**98.9**	**85.4**	96.6	**99.9**

Table 2: Comparison to GOG. Best results in top-matching rates for each distance metric are in bold.

compared to our method in their original setting. For this purpose, the same evaluation protocol employed by most algorithms was adopted. More specifically, for PRID2011 dataset, only sequences from 178 persons consisting of more than 27 frames were retained. Half of each dataset was used for training and the remaining half for testing. Experiments were repeated over 10 trials and average results in CMC top-matching rates are reported in Table 3. The distance metric used in this experiment was kNFST due to its superior performance. It is worth noting that the results here are different from those of the previous subsection because the evaluation protocol of PRID2011 and the parameters (patch size, ε, and number of stripes) have been changed as previously detailed in Section 4.2.

ColHOG3D [9] and STFV3D [10] are state-of-the-art spatio-temporal descriptors that fall in the same category with GOG3D. The difference in performance between GOG3D and these descriptors is very obvious, even when using the same distance metric KISSME as shown in the second row of Table 2. The gap in rank1 matching-rate with the better performing STFV3D is over 10% on iLIDS-VID and around 20% on PRID2011.

When compared to deep learning techniques, GOG3D + kNFST outperforms RNN [22], CNN + XQDA [25] and ASTPN [23] by at least 18% on iLIDS-VID and 17% on PRID2011 in terms of rank1 accuracy. It also outperforms multi-shot ranking methods DRAH [4] and SPW [8] by around 10% on iLIDS-VID for SPW and around 5% on PRID2011 for DRAH. The only method that exhibits comparable or slightly worse performance than GOG3D on both datasets is PAM+KISSME [30]. However, while not being in the same category with GOG3D, PAM requires (i) extracting low-level features, (ii) fitting GMMs to person sequences with many parameters to learn, and (iii) the final person representation does not fall in a Euclidean space. Hence, special care needs to be taken for matching. This also means that the final person signature does not exhibit the flexibility of use with other distance metrics.

It is finally noteworthy that GOG3D not only achieves outstanding results on two challenging benchmarks, it is also simple, flexible and computationally efficient. The low computational complexity is derived from omitting additional tasks like walking cycle extraction and fragment selection used by similar space-time descriptors, or feature clustering and frame weighting required for multi-shot ranking methods that may involve further learning. Moreover, the high computational cost needed to train deep neural networks is also avoided. Finally, the flexibility of GOG3D feature is granted by the possibility of its use with any matching method in both supervised and unsupervised settings since it does not involve any learning.

4.5 Computational Cost

GOG3D is implemented in MATLAB and experiments are run on a desktop PC equipped with Intel Xeon X5550 @2.67GHz CPU. The average time to extract GOG3D features per frame is 0.44 seconds. It is computed on 10 video sequences from PRID2011 dataset

 ISBN 978-80-86943-37-4

Dataset	**iLIDS-VID**			**PRID2011**		
Rank	**1**	**5**	**20**	**1**	**5**	**20**
ColHOG3D+DVR [9]	39.5	61.1	81	40	71.7	92.2
STFV3D+KISSME [10]	44.3	71.7	91.7	64.1	87.3	92
RNN [22]	58	84	96	70	90	97
CNN+XQDA [25]	53	81.4	95.1	77.3	93.5	99.3
ASTPN [23]	62	86	98	77	95	99
DRAH [4]	64	86	96.3	88.7	97.9	99.7
PAM+KISSME [30]	**79.5**	95.1	99.1	92.5	**99.3**	**100**
SPW [8]	69.3	89.6	98.2	83.5	96.3	**100**
GOG3D^{10}+kNFST (proposed)	**80**	**95.3**	**99.5**	**93.6**	**99.4**	**100**
GOG3D^{15}+kNFST (proposed)	**79.5**	**95.4**	**99.5**	**94**	99.1	**100**

Table 3: Comparison to state-of-the-art methods. Best and second best results in top-matching rates are in bold.

and averaged over the number of frames constituting these sequences. Under the same setting, the time taken to compute GOG features is 0.35 seconds per frame. It is intuitive for GOG3D to be slightly slower than GOG since it uses additional pixel features and more horizontal stripes. However, compared to other video person re-id descriptors [9, 10], GOG3D is evidently more efficient since it omits the walking cycle extraction step and any further post-processing. Moreover, frame-wise feature pooling employed with GOG3D renders the matching process very efficient. For instance, the average time taken to train the kNFST metric on PRID2011 dataset over 10 trials is 0.034 seconds, and the testing time on the same dataset is 0.008 seconds which is exceptionally fast for video-based re-id methods.

5 CONCLUSION

A novel spatio-temporal descriptor for video-based person re-id based on hierarchical Gaussian distributions was presented in this paper. The proposed algorithm leverages the gradient in the temporal direction to describe the temporal correlation between consecutive frames yielding significant improvement in accuracy. Unlike available spatio-temporal re-id descriptors, the proposed method does not require any complex walking cycle extraction of frame selection and weighting. It also does not involve any learning. By simply averaging the frame-wise computed features over a person sequence, robust representation can be achieved and consequently fed to most off-shelf distance metrics.

A thorough analysis of the proposed descriptor was conducted on 2 widely used benchmarks using 5 distance metrics, highlighting the advantages brought by exploiting temporal cues. Extensive experiments showed that the performance achieved surpasses similar methods by a large margin. It also outperforms a number of existing deep learning and multi-shot ranking techniques.

ACKNOWLEDGEMENT

This publication was made possible NPRP grant #NPRP 8-140-2-065 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

6 REFERENCES

[1] S. Karanam, M. Gou, Z. Wu, A. Rates-Borras, O. Camps, R. Radke, A systematic evaluation and benchmark for person re-identification: Features, metrics, and datasets., IEEE Transactions on Pattern Analysis and Machine Intelligence 41 (3) (2019) 523–536.

[2] L. Zheng, Y. Yang, A. G. Hauptmann, Person re-identification: Past, present and future, arXiv preprint arXiv:1610.02984.

[3] C. Riachy, A. Bouridane, Person re-identification: Attribute-based feature evaluation, in: IEEE World Symposium on Applied Machine Intelligence and Informatics (SAMI), 2018.

[4] S. Karanam, Z. Wu, R. J. Radke, Learning affine hull representations for multi-shot person re-identification, IEEE Transactions on Circuits and Systems for Video Technology.

[5] S. Karanam, Y. Li, R. J. Radke, Sparse re-id: Block sparsity for person re-identification, in: IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2015.

[6] X. Zhu, X.-Y. Jing, X. You, X. Zhang, T. Zhang, Video-based person re-identification by simultaneously learning intra-video and inter-video distance metrics, IEEE Transactions on Image Processing 27 (11) (2018) 5683–5695.

[7] Y.-J. Cho, K.-J. Yoon, Improving person re-identification via pose-aware multi-shot matching, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[8] W. Huang, C. Liang, Y. Yu, Z. Wang, W. Ruan, R. Hu, Video-based person re-identification via self paced weighting, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[9] T. Wang, S. Gong, X. Zhu, S. Wang, Person re-identification by video ranking, in: European Conference on Computer Vision (ECCV), 2014.

[10] K. Liu, B. Ma, W. Zhang, R. Huang, A spatio-temporal appearance representation for video-based pedestrian re-identification, in: International Conference on Computer Vision, 2015.

[11] T. Matsukawa, T. Okabe, E. Suzuki, Y. Sato, Hierarchical gaussian descriptor for person re-identification, in: Computer Vision and Pattern Recognition (CVPR), 2016.

[12] B. Ma, Y. Su, F. Jurie, Covariance descriptor based on bio-inspired features for person re-identification and face verification, Image and Vision Computing 32 (6-7) (2014) 379–390.

[13] B. Ma, Q. Li, H. Chang, Gaussian descriptor based on local features for person re-identification, in: Asian Conference on Computer Vision (ACCV), 2014.

[14] F. Xiong, M. Gou, O. Camps, M. Sznaier, Person re-identification using kernel-based metric learning methods, in: European Conference on Computer Vision (ECCV), 2014.

[15] S. Liao, Y. Hu, X. Zhu, S. Z. Li, Person re-identification by local maximal occurrence representation and metric learning, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[16] M. Gou, O. Camps, M. Sznaier, mom: Mean of moments feature for person re-identification, in: International Conference on Computer Vision (ICCV), 2017.

[17] L. Bazzani, M. Cristani, V. Murino, Symmetry-driven accumulation of local features for human characterization and re-identification, Computer Vision and Image Understanding 117 (2) (2013) 130–144.

[18] M. Koestinger, M. Hirzer, P. Wohlhart, P. M. Roth, H. Bischof, Large scale metric learning from equivalence constraints, in: Computer Vision and Pattern Recognition (CVPR), 2012.

[19] L. Zhang, T. Xiang, S. Gong, Learning a discriminative null space for person re-identification, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[20] D. Gray, H. Tao, Viewpoint invariant pedestrian recognition with an ensemble of localized features, in: European conference on computer vision (ECCV), 2008.

[21] D. S. Cheng, M. Cristani, M. Stoppa, L. Bazzani, V. Murino, Custom pictorial structures for re-identification., in: British Machine Vision Conference (BMVC), 2011.

[22] N. McLaughlin, J. Martinez del Rincon, P. Miller, Recurrent convolutional network for video-based person re-identification, in: Computer Vision and Pattern Recognition (CVPR), 2016.

[23] S. Xu, Y. Cheng, K. Gu, Y. Yang, S. Chang, P. Zhou, Jointly attentive spatial-temporal pooling networks for video-based person re-identification, in: International Conference on Computer Vision (ICCV), 2017.

[24] W. Li, R. Zhao, T. Xiao, X. Wang, Deepreid: Deep filter pairing neural network for person re-identification, in: Computer Vision and Pattern Recognition (CVPR), 2014.

[25] L. Zheng, Z. Bie, Y. Sun, J. Wang, C. Su, S. Wang, Q. Tian, Mars: A video benchmark for large-scale person re-identification, in: European Conference on Computer Vision (ECCV), 2016.

[26] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, Q. Tian, Scalable person re-identification: A benchmark, in: International Conference on Computer Vision (ICCV), 2015.

[27] R. Zhao, W. Ouyang, X. Wang, Unsupervised salience learning for person re-identification, in: Computer Vision and Pattern Recognition (CVPR), 2013.

[28] A. Klaser, M. Marszałek, C. Schmid, A spatio-temporal descriptor based on 3d-gradients, in: British Machine Vision Conference (BMVC), 2008.

[29] T. Wang, S. Gong, X. Zhu, S. Wang, Person re-identification by discriminative selection in video ranking, IEEE transactions on pattern analysis and machine intelligence 38 (12) (2016) 2501–2514.

[30] F. M. Khan, F. Brèmond, Multi-shot person re-identification using part appearance mixture, in: Winter Conference on Applications of Computer Vision (WACV), 2017.

[31] T. Kobayashi, N. Otsu, Image feature extraction using gradient local auto-correlations, in: European Conference on Computer Vision (ECCV), 2008.

[32] M. Lovrić, M. Min-Oo, E. A. Ruh, Multivariate normal distributions parametrized as a riemannian symmetric space, Journal of Multivariate Analysis 74 (1) (2000) 36–48.

[33] M. Hirzer, C. Beleznai, P. M. Roth, H. Bischof, Person re-identification by descriptive and discriminative classification, in: Scandinavian Conference on Image Analysis (SCIA), 2011.

 ISBN 978-80-86943-37-4