ISSN 1213-6972 *Volume 20, Number 1, 2012*

Journal of WSCG

An international journal of algorithms, data structures and techniques for computer graphics and visualization, surface meshing and modeling, global illumination, computer vision, image processing and pattern recognition, computational geometry, visual human interaction and virtual reality, animation, multimedia systems and applications in parallel, distributed and mobile environment.

Vaclav Skala – Union Agency

Journal of WSCG

Editor-in-Chief: Vaclav Skala
c/o University of West Bohemia
Faculty of Applied Sciences
Univerzitni 8
CZ 306 14 Plzen
Czech Republic
http://www.VaclavSkala.eu

Managing Editor: Vaclav Skala

Printed and Published by:
Vaclav Skala - Union Agency
Na Mazinach 9
CZ 322 00 Plzen
Czech Republic

Hardcopy: ***ISSN 1213 – 6972***
CD ROM: ***ISSN 1213 – 6980***
On-line: ***ISSN 1213 – 6964***

Journal of WSCG

Editor-in-Chief

Vaclav Skala

c/o University of West Bohemia
Centre for Computer Graphics and Visualization
Univerzitni 8
CZ 306 14 Plzen
Czech Republic

http://www.VaclavSkala.eu

Journal of WSCG URLs: http://www.wscg.eu or http://wscg.zcu.cz/jwscg

Editorial Advisory Board
MEMBERS

Journal of WSCG

Vol.20, No.1

Contents

Coupling between Meshless FEM Modeling and Rendering on GPU for Real-time Physically-based Volumetric Deformation

Muhammad Mobeen Movania
Nanyang Technological University, Singapore
mova0002@e.ntu.edu.sg

Feng Lin
Nanyang Technological University, Singapore
asflin@ntu.edu.sg

Kemao Qian
Nanyang Technological University, Singapore
mkmqian@ntu.edu.sg

Wei Ming Chiew
Nanyang Technological University, Singapore
chie0017@e.ntu.edu.sg

Hock Soon Seah
Nanyang Technological University, Singapore
ashsseah@ntu.edu.sg

ABSTRACT

For real-time rendering of physically-based volumetric deformation, a meshless finite element method (FEM) is proposed and implemented on the new-generation Graphics Processing Unit (GPU). A tightly coupled deformation and rendering pipeline is defined for seamless modeling and rendering: First, the meshless FEM model exploits the vertex shader stage and the transform feedback mechanism of the modern GPU; and secondly, the hardware-based projected tetrahedra (HAPT) algorithm is used for the volume rendering on the GPU. A remarkable feature of the new algorithm is that CPU readback is avoided in the entire deformation modeling and rendering pipeline. Convincing experimental results are presented.

Keywords

Volumetric deformation, physically based deformation, finite element method, meshless model, GPU transform feedback, volume rendering

1. INTRODUCTION

Interactive visualization of physically-based deformation has been long pursued as it plays a significant role in portraying complex interactions between deformable graphical objects. In many applications, such subtle movements are necessary, for example, surgical simulation systems in which a surgeon's training experience is directly based on the feedback he/she gets from the training system.

Prior to the advent of the Graphics Processing Unit (GPU), such interactions were only restricted to sophisticated hardware and costly workstations. Thanks to the massive processing capability of modern GPUs, such interactions can now be carried out on a consumer desktop or even a mobile device. However, even with such high processing capability, it is still difficult to simultaneously deform and visualize a volumetric dataset in realtime. Several promising volumetric deformation techniques have been proposed, but they have mostly favored a specific stage of the programmable graphics pipeline. Therefore, these approaches could not utilize the full potential of the hardware efficiently.

With new hardware releases, new and improved features have been introduced into the modern GPU. One such feature is transform feedback in which the GPU feedbacks the result from the geometry shader stage back to the vertex shader stage. While this method was usually used for dynamic tessellation and level-of-detail (LOD) rendering, we have proposed to use this mode for an efficient deformation pipeline. Since this deformation uses the vertex shader stage, we may streamline the fragment shader stage for volume rendering, forming a coupled graphics pipeline.

In Section 2, we report a comprehensive survey on deformation algorithms and GPU acceleration technologies. Then, we describe our new meshless FEM approach and the formulation of the physical model in Section 3. In Section 4, we present the techniques for coupling between the novel deformation pipeline and the GPU-based volume rendering. Experimental results and comparisons of the performance are given in Section 5. And finally, Section 6 concludes this paper.

2. PREVIOUS WORK

Up to now, physically-based deformation can be broadly classified into mesh-based and meshless methods. Mesh-based methods include finite element method (FEM), boundary element method (BEM), and mass spring system. Meshless methods include smoothed point hydrodynamics (SPH), shape matching and Lagrangian methods. We refer the reader for meshless methods to [ST99], [HF00] [BBO03], [NRBD08] and for physically-based deformation approaches in computer graphics to [NMK06].

One of the first mass spring methods for large deformation on the GPU for surgical simulators is attributed to Mosegaard et al. [MHSS04], in which Verlet integration is implemented in the fragment shader. Using the same technique, Georgii et al. [GEW05] implemented a mass spring system for soft bodies. The approach by Mosegaard et al. [MHSS04] requires transfer of positions in each iteration. Georgii et al. [GEW05] thus focused on how to minimize this transfer by exploiting the ATI Superbuffers extension. They described two approaches for implementation: an edge centric approach (ECA) and a point centric approach (PCA). A CUDA-based mass spring model has been proposed recently [ADLETG10].

All of the mass spring models and methods discussed earlier used explicit integration schemes which are only conditionally stable. For unconditional stability, implicit integration could be used as demonstrated for the GPU-based deformation by Tejada et al. [TE05].

The mass spring models are fast but inaccurate. FEM methods have been proposed for more accurate simulation and animation. The model assumes linear elasticity so the deformation model is limited to small displacements. In addition, the small strain assumption produces incorrect results unless the corotational formulation is used [MG04] which isolates the per-element rotation matrix when computing the strain.

With the increasing computational power, non-linear FEM has been explored, in which both material and geometric non-linearities are taken into consideration [ML03], and [ZWP05]. The fast numerical methods for solving FEM systems for deformable bodies are based on the multi-grid scheme. These approaches have been extended in animation [SB09] and medical applications for both the tetrahedral [GW05] [GW06] and hexahedral FEM [DGW10].

In addition to the above approaches, explicit non-linear methods have been proposed using the Lagrangian explicit dynamics [MJLW07] which are especially suitable for real-time simulations. A single stiffness matrix could be reused for the entire mesh. Especially with the introduction of the CUDA architecture, the Lagrangian explicit formulation has been applied for both the tetrahedral FEM [TCO08] as well as the hexahedral FEM [CTA08].

The problem with explicit integration is that it is only conditionally stable, that is, for convergence, the time step value has to be very small. In addition, such integration schemes may not be suitable during complex interactions as in surgery simulations and during topological changes (for example, cutting of tissues). Allard et al. [ACF11] circumvent these cons by proposing an implicitly integrated GPU-based non-linear corotational model for laparoscopic surgery simulator. They use the pre-conditioned Conjugate Gradient (CG) solver for solving the FEM. They solve the stiffness matrices directly on the mesh vertices rather than building the full stiffness assembly. Ill-conditioned elements may be generated in the case of cutting or tearing which may produce numerical instabilities.

3. THE MESHLESS FEM APPROACH

Although there have been significant achievements in deformable models, a few difficulties in the mesh-based models still exist in real-time volumetric deformation, as highlighted in the followings:

- Approximating a volumetric dataset requires a large number of finite tetrahedral elements. Numerical solution of such a large system would require a large stiffness matrix assembly. This makes the model unsuitable for real-time volumetric deformation. In addition, the corotated formulation is needed which further increases the computational burden.
- The solution of the tetrahedral FEM requires an iterative implicit solver for example Newton Raphson (Newton) or Conjugate Gradient (CG) method. These methods converge slowly. Moreover, the implicit integration solvers reduce the overall energy of the system causing the physical simulation to dampen excessively.
- Even though multi-grid schemes are fast, they have to update the deformation parameters across different grid hierarchy levels. This requires considerable computation. Moreover, the number

of grid levels required is subjective to the dataset at hand and there is no rule to follow for accurate results.

On the other hand, we have noticed that the meshless FEM approach has not been applied for volumetric deformation in the literature. Our preliminary study shows that the meshless formulation possesses a few advantages:

- It supports deformations without the need for stiffness warping (the corotated formulation).
- The solution of meshless FEM is based on a semi-implicit integration scheme which not only is stable but also converges faster as compared to the implicit integration required by the tetrahedral FEM solver. In addition, it does not introduce artificial damping.
- It does not require an iterative solver such as conjugate gradient (CG) method which is required for conventional FEM.

Therefore, in this study, we are interested in exploiting the meshless FEM approach for volumetric deformation, coupled with simultaneous GPU-based real-time visualization.

Formulation of the Physical Model

We base our deformation modeling and rendering on the continuum elasticity theory. Key parameters in the physical model are stress, strain and displacement. Strain (ε) is defined as the relative elongation of the element. Assuming an element undergoing a displacement (ΔL) having length (l), the strain may be given as:

$$\varepsilon = \frac{\Delta L}{l}$$

For a three-dimensional problem, the strain (ε) is represented as a symmetric 3×3 tensor. There are two popular choices for the strain tensor in computer graphics, the linear Cauchy strain tensor given as

$$\varepsilon_{Cauchy} = \frac{1}{2}(\nabla U + [\nabla U]^T) \quad (1)$$

and the non-linear Green strain tensor given as

$$\varepsilon_{Green} = \frac{1}{2}(\nabla U + [\nabla U]^T + [\nabla U]^T \nabla U) \quad (2)$$

In Eq. (1) and (2), the ∇U is the gradient of the displacement field U. Similar to the strain, in the three-dimensional problem, the stress tensor (σ) is also given as a 3×3 tensor. Assuming that the material under consideration is isotropic and it undergoes small deformations (geometric linearity), the stress and strain may be linearly related (material linearity) using Hooke's law, given as

$$\sigma = D\varepsilon \quad (3)$$

Since stress and strain are symmetric matrices, there are six independent elements in each of them. This reduces the isotropic elasticity matrix (D) to a 6×6 matrix as follows:

$$D = B\begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix}$$

$$B = \frac{E}{(1+\nu)(1-2\nu)}$$

where, E is the Young's modulus of the material which controls the material's resistance to stretching and ν is the Poisson's ratio which controls how much a material contracts in the direction transverse to the stretching.

In a finite element simulation, we try to estimate the amount of displacement due to the application of force. There are three forces to consider (see Fig. 1):

- Stress force (σ) which is an internal force,
- Nodal force (q) which is an external force applied to each finite element node, and
- Loading force (t) which is an external force applied to the boundary or surface of the finite element.

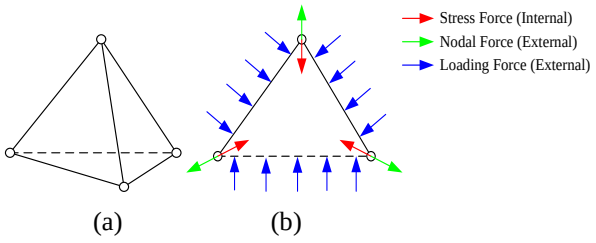


Figure 1. Different forces acting on a finite tetrahedral element (a), with (b) its cross sectional view highlighting the different internal and external forces acting on the finite element

For the finite element to be in static equilibrium, the amount of work done by the external forces must be equal to that of the internal forces, given as

$$\int_{V^e} W_\sigma dV = W_q + \int_{A^e} W_t dA \quad (4)$$

where, W_σ is the internal work done per unit volume by stress σ, W_q is the external work done by the nodal force q on the element's node and W_t is the external work done by the loading force t on the element per unit area. V^e is the volume and A^e is the area of the finite element e. W_σ is given as

$$W_\sigma = (\delta\varepsilon)^T \sigma$$

where, $\delta\varepsilon$ is the strain produced by the stress σ. Similarly, W_q is given as

$$W_q = (\delta u^e)^T q^e$$

where, δu^e is the displacement of the finite element e produced by the force q^e. W_t is given as

$$W_t = (\delta u)^T t$$

Substituting these in Eq. (4), we get

$$\int_{V^e} (\delta\varepsilon)^T \sigma dV = (\delta u^e)^T q^e + \int_{A^e} (\delta u)^T t dA \quad (5)$$

Since δu^e provides the displacement of node e at vertices only, to get the displacement at any point within the finite element, we can interpolate it with the shape function $\boldsymbol{N}$. After applying a differential operator $\boldsymbol{S}$ to the shape functions, we get the change in strain ($\delta\varepsilon$). The matrix product ($\boldsymbol{SN}$) can be replaced by B

$$\delta\varepsilon = B\delta u^e$$

Substituting ($\delta\varepsilon$) in Eq. (5), we get

$$\int_{V^e} (B\delta u^e)^T \sigma dV = (\delta u^e)^T q^e + \int_{A^e} (N\delta u^e)^T t dA \quad (6)$$

Simplifying Eq. (6), taking the constant terms out of the equation and solving integral (see Appendix) gives

$$B^T DBu^e V^e = q^e + \int_{A^e} (N^T t) dA \quad (7)$$

The left side is replaced by the element stiffness matrix ($K^e = B^T DBV^e$) and the right by the element surface force (f^e), which gives us the stiffness matrix assembly equation:

$$K^e u^e = q^e + f^e \quad (8)$$

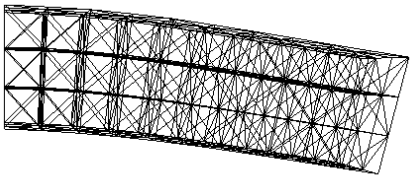
The Meshless FEM

The conventional FEM methods discretize the whole body into a set of finite elements. Calculation of the element stiffness matrix in Eq. (8) requires the volume of the body which is represented as the sum of the finite elements' volume. For instance, the widely used corotated linear FEM [MG04] has to construct the global stiffness matrix for each deformation frame, and the matrix is then solved using an iterative solver such as the conjugate gradients (CG). This makes the implementation inefficient for a large volume.

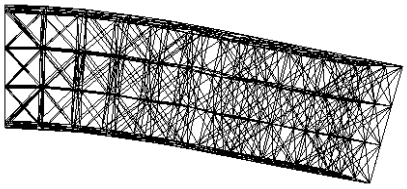
In our meshless FEM, the whole body is sampled at a finite number of points. The typical simulation quantities such as the position (x), velocity (v) and density (ρ) are all stored with the points, and the displacement field is estimated from the volume of the point and its mass distribution. The gradient of the displacement field is then estimated to obtain the Jacobian. Finally, the Jacobian is used to calculate the stresses and strains. These, in turn, allow us to obtain the internal forces. Since the meshless FEM uses the moving least square approximation, it does not require the stiffness matrix assembly, enabling a much better execution performance.

To ascertain that our proposed meshless FEM is able to produce the same deformation as that in the conventional FEM such as the corotated linear FEM, we conducted a computational experiment on a horizontal beam as shown in Fig. 2. The two results show a horizontal beam having Young's modulus of 500,000 psi and the Poisson ratio of 0.33. The dimensions of the two beams are the same. The beam in Fig. 2 (a) contains 450 tetrahedra for the corotated linear FEM whereas its equivalent one in Fig. 2 (b) contains 176 points for the meshless FEM.

While the two computations yield the same deformation under the given load, our meshless FEM has a significantly improved execution performance: 40 msecs per frame by the corotated linear FEM, compared to 1.25 msecs per frame with the meshless FEM. These timings include both the deformation as well as rendering time.



(a)



(b)

Figure 2. Comparison of deformation of a horizontal beam using (a) linear FEM and (b) meshless FEM

In a dynamic simulation, we are to solve the following system:

$$m\ddot{x} = -c\dot{x} + \sum (f_{\text{int}} + f_{ext}) \quad (9)$$

The first term on the right is the velocity damping term with c being the damping coefficient. For an infinitesimal element, the mass is approximated using density (ρ). This changes Eq. (9) to

$$\rho\ddot{x} = -c\dot{x} + \sum (f_{\text{int}} + f_{ext}) \quad (10)$$

The external forces (f_{ext}) are due to gravity, wind, collision and others. Since our system assumes geometric and material linearity, Eq. (10) becomes a linear PDE that may be solved by discretizing the domain of the input dataset using finite differences over finite elements. This system may be solved using either explicit or implicit integration schemes.

Smoothing Kernel

In the conventional FEM, the volume of the body is estimated from the volume of its constituent finite elements. Calculation of the element stiffness matrix requires the volume of the body which is usually represented as the sum of the finite elements volume. In the case of the meshless FEM, it is approximated from the point's neighborhood. For each point, its mass is distributed into its neighborhood by using a smoothing kernel (w)

$$w(r,h) = \begin{cases} \frac{313}{64\pi h^9}(h^2 - r^2)^3 & if\ (r < h) \\ 0 & else \end{cases} \quad (11)$$

where, r is the distance between the current particle and its neighbor, and h is the kernel support radius. The density is approximated by summing the product of the current point's mass with the kernel.

We analyzed the effect of varying the smoothing kernel [MCG03]. These kernels include the normal smoothing kernel (as given in Eq. (11)) the spiky kernel given as

$$w(r,h) = \begin{cases} \frac{15}{\pi h^6}(h - \|r\|)^3 & 0 \le \|r\| \le h \\ 0 & \|r\| > h \end{cases} \quad (12)$$

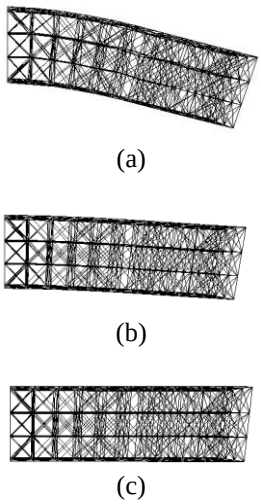
and the blobby kernel given as

$$w(r,h) = \begin{cases} \frac{15}{2\pi h^3}(-\frac{r^3}{2h^3} + \frac{r^2}{h^2} + \frac{h}{2r} - 1) & 0 \le \|r\| \le h \\ 0 & \|r\| > h \end{cases} \quad (13)$$

The deformation results on a horizontal beam containing 176 points are shown in Fig. 3. Note that for all the beams shown in Fig. 3, the Young's modulus of 500,000 psi and the Poisson ratio of 0.33 are used. As can be seen, changing the smoothing kernel alters the stiffness of the soft body. This is because each kernel has a distinct support radius which influences the neighboring points. Moreover, each of these kernels has a different falloff (or, different derivative) which gives a different deformation result even though the rest of the simulation parameters are the same.

Propagation of Deformation

For propagating the stress, strain and body forces in the meshless FEM, we compute the gradient of the displacement field (U) by a moving least square interpolation between the displacement values at the current point (u_i) and its neighbor (u_j) as given by



(a)

(b)

(c)

Figure 3. Effects of different smoothing kernels on the deformation: (a) the normal smoothing kernel (Eq. 11), (b) the spiky kernel (Eq. 12) , and (c) the blobby kernel (Eq. 13)

$$e = \sum_i (u_j - u_i)^2 w_{ij} \quad (14)$$

where, w_{ij} is the kernel function given in Eq. (11).

The displacement values (u_j) are given using the spatial derivatives approximated at point (i) as

$$u_j = u_i + \nabla u.(x_j - x_i)$$

We want to minimize the error (e) in Eq. (14) so we differentiate e with respect to X, Y and Z and set the derivatives equal to zero. This gives us three equations for three unknowns

$$\nabla u\,|_x = A^{-1}\left(\sum_i (u_j - u_i)(x_j - x_i)w_{ij}\right)$$

where, $A=\Sigma_i(x_j\text{-}x_i)(x_j\text{-}x_i)^T w_{ij}$ is the moment matrix that can be pre-calculated since it is independent of the current position and displacement. Once ∇u is obtained, the strain (ε) is obtained using Eq. (2). Using this strain, the stress (σ) may be obtained using Eq. (3). The internal forces (f_{int}) in Eq. (10) are calculated as the divergence of the strain energy which is a function of the particle's volume

$$U_i = v_i \frac{1}{2}(\varepsilon_i.\sigma_i)$$

where, v_i is the volume of the particle. The force acting on neighboring particle (j) due to particle (i) is given as

$$f_j = -\nabla U_i = -v_i \frac{1}{2}\nabla\varepsilon_i.\sigma_i$$

To sum up, the internal forces acting on the particles i and j may be given as

$$\begin{aligned} f_i &= -2v_i J\sigma_i d_i \\ f_j &= -2v_j J\sigma_j d_j \end{aligned} \quad (15)$$

where, $d_i=M^{-1}(\Sigma_i(x_j-x_i)w_{ij})$ and $d_j=M^{-1}(x_j-x_i)w_{ij}$, J is the Jacobian, v is the volume of the point and σ is the stress at the given point.

Note that in the case of point masses, the volume may be calculated from the mass density in the point's neighborhood. The mass (m_i) of the point (i) is calculated using

$$m_i = sr_i^3\rho$$

where, r_i is the average distance between the mass point and its neighbors, ρ is the material density and s is a scaling constant which is calculated as

$$\beta_j = \frac{1}{\sum_j r_j^3 w_j}$$

$$s = \frac{\sum_i \beta_i}{n}$$

where, n is the total number of points. Once the mass is obtained, the per-point density is then obtained by summing the product of the mass of its neighbor (m_j) with the kernel evaluated at the neighbor j (w_j), as follow:

$$\rho_i = \sum_j m_j w_j$$

This density can then be used to obtain the volume of the point which is given as

$$vol_i = \frac{m_i}{\rho_i}$$

During the force evaluation, the internal forces are scattered in the neighborhood of the point using Eq. (15). Since scattering cannot be implemented in a GPU shader program, therefore, for parallel processing, we convert the scatter operation into a gather operation by reformulation [Buc05]. First, the sum of force matrices (F_e and F_v) is obtained

$$sumF_i = (F_e + F_v)$$

Instead of calculating the force on the point i as given in Eq. (15), we multiply the matrix ($sumF_i$) with (d_i)

$$F_i = F_i + sumF_i d_i$$

where d_i is obtained as in Eq. (15). The internal force due to the neighbor points is then given as

$$F_i = F_i - sumF_j d_j$$

where, F_i is the net internal force, j loops for each neighbor of the current point i and d_j is obtained as in Eq. (15). This allows us to run the program in parallel on all points simultaneously.

4. COUPLING BETWEEN VOLUMETRIC DEFORMATION AND RENDERING

Prior rendering algorithms have resorted to the GPGPU-based techniques for evaluating the position and/or velocity integration on the fragment shader [GEW05], [GW06], [GW08], [TE05], [VR08]. This involves rendering a screen sized quad with the appropriate textures setup and then the fragment shader is invoked to solve the integration for each fragment. The output from the fragment shader is written to another texture. On the contrary, we adopt a different approach (see Fig. 4).

We implement the meshless deformation by using the transform feedback mechanism of the modern GPU. The original unstructured mesh vertices are used directly in our implementation. Our deformable pipeline is implemented in the vertex shader stage and it outputs to the buffer object registered to the transform feedback. We use a pair of buffer objects for both the positions and velocities to avoid the simultaneous read/write race condition [ML12a]. So when we are reading from a pair of position and velocity buffers, we write to another pair. In each iteration, the pair is swapped.

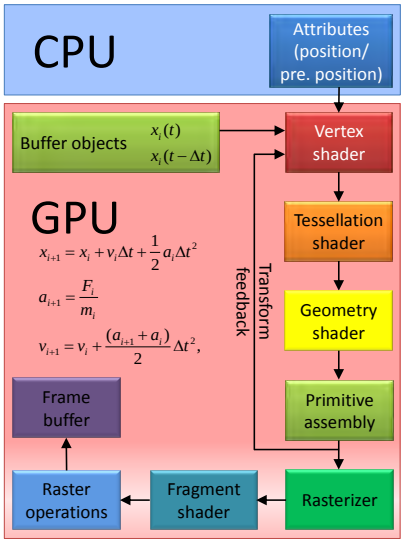


Figure 4. Proposed deformation pipeline using transform feedback

Attribute Setup for Transform Feedback

All the per-point attributes such as the current position (x_i), previous position (x^0_i), and velocity (v_i) are passed to the transform feedback vertex shader as per-vertex attributes.

The inverse mass matrices of individual nodes are stored in a texture ($texM_{inv}$) and the rest distances are stored in an attribute (r_{dist}); The point neighbor distance (r_i), support radius (h_i), the point mass (m_i) and volume (vol_i) are pre-computed and stored into a set of textures: the *neighborCountsTexture* and the *neighborListTexture*.

A pair of position and velocity buffer objects is bound as a transfer feedback buffer. This enables the vertex shader to output the results to a buffer object directly without CPU readback.

Dataflow

Referring to Fig. 5, for each rendering cycle, we swap between the two buffers to alternate the read/write pathways. Before the transform feedback can proceed, we need to bind the update array objects. Once the update array object is bound, we bind the appropriate buffer objects (for reading the current positions and velocities) to the transform feedback.

The draw point call is issued to allow the writing of vertices to the buffer object. The transform feedback is then disabled. Following the transform feedback, the rasterizer is enabled and then the points are drawn. This time, the render array objects are bound. This renders the deformed points on screen.

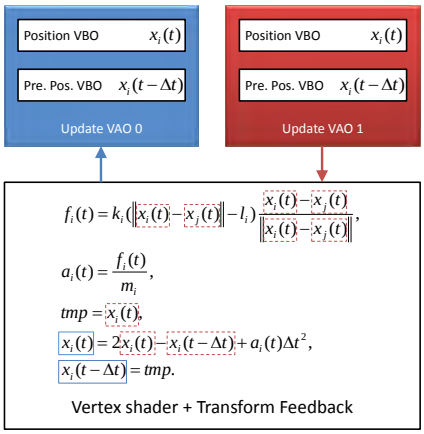


Figure 5. The vertex array object and vertex buffer object setup for transform feedback: the blue/solid rectangles show the attributes written to and the red/dotted rectangles show the attributes being read simultaneously from another vertex array object

Evaluation of Forces

The forces are evaluated per-vertex. Rather than storing the isotropic elasticity matrix (D) as a 6×6 matrix, we store it into a single vec3 attribute containing the three non-zero entries. The inverse mass matrix is pre-calculated at initialization on the CPU and then transferred to the GPU as a per-vertex attribute.

At initialization, the nearest K neighbors of the point are found using a neighborhood search. For the examples shown in this paper, the K is set as 10. This value was arrived at after some experiments. Large size of K generally makes the object stiffer. For fast and efficient search, we use a Kd-tree extracted from the given point set. The found points become the neighbors of the current point. Then, the mass, volume, weights and moment matrices are calculated for each point.

We first calculate the external forces such as the gravity force and the velocity damping force. We then calculate the Jacobians, the stresses and the internal forces using the neighbor node attributes.

Numerical Integration

Following the calculation of the forces, we perform the leap frog integration. The leap frog integration works by evaluating the velocities at 1/2 time step offset from the position. Mathematically, the leap frog integration is given as

$$x_{i+1} = x_i + v_i \Delta t + a_i \frac{\Delta t^2}{2}$$
$$v_{i+1} = v_i + \frac{a_i + a_{i+1}}{2} \Delta t \qquad (16)$$

The advantage that we obtain from this integration scheme is that it conserves the overall energy of the system. The expressions given in Eq. (16) may be converted directly into shader statements. The current position (x_i) and velocity (v_i) are passed in as per-vertex attributes whereas the next position (x_{i+1}) and velocity (v_{i+1}) are written using the transform feedback mechanism.

Cell Projection of Transformed Volume

In view of the various aspects of modeling and rendering requirements for fast rendering of transformed points, we use the hardware assisted projected tetrahedra (HAPT) algorithm [MMF10]. The deformation pipeline outputs a pair of buffer objects. These are used directly as positions for cell projection. Thus, we do not need to transfer the deformation results to CPU.

The nonrigid transformation is incorporated in the HAPT pipeline by streaming our deformed points directly. The HAPT algorithm stores positions in texture objects. In our implementation, we reuse the buffer objects output from our deformation pipeline directly. This involves no CPU readback, and thus, the data can be visualized directly.

User Interaction and Collision Detection with Deformed Volume

In a simulation system, it is often necessary to interact with the volume in realtime. In our proposed pipeline, we can interact with the volume in two ways: by modifying the vertex positions directly and by modifying the tetrahedra. In either case, we map the GPU memory to obtain the pointer to the GPU memory, index to the appropriate value and modify the value directly [ML12b].

Collision detection and response can be handled directly in the integration vertex shader by applying constraints to the calculated positions. For example,

if we have a mass point x_i, a sphere with a radius r and center C, the collision constraint may be given as

$$x_{i+1} = \begin{cases} C + \dfrac{(x_i - C).r}{|x_i - C|} & if\ |x_i - C| < r \\ x_i & else \end{cases}$$

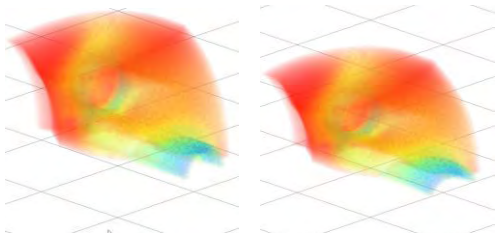
Likewise, other constraints may be integrated directly in the proposed pipeline using the vertex or geometry shader.

5. EXPERIMENTAL RESULTS AND PERFORMANCE ASSESSMENT

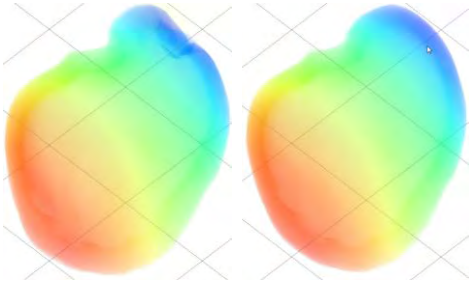
The coupled deformation and rendering pipeline has been implemented on a Dell Precision T7500 desktop with an Intel Xeon E5507 @ 2.27 MHz CPU. The machine is equipped with an NVIDIA Quadro FX 5800 graphics card. The viewport size for the renderings is 1024×1024 pixels.

The output results with deformation and rendering are shown in Fig. 6. We applied the meshless FEM to two volumetric datasets, the spx dataset (containing 2896 points and 12936 tetrahedra) and the liver dataset (1204 points and 3912 tetrahedra).

For all our experiments, the normal smoothing kernel Eq. (11) is used. We allowed the spx dataset to fall under gravity while the liver dataset was manipulated by the user. The time step value (*dt*) used for this experiment is 1/60. Thanks to the convenience of our proposed deformation pipeline, we can integrate our deformation pipeline directly into the HAPT algorithm.



(a)



(b)

Figure 6. Two frames of deformation of (a) the spx dataset falling due to gravity on the floor and (b) the liver dataset manipulated by the user

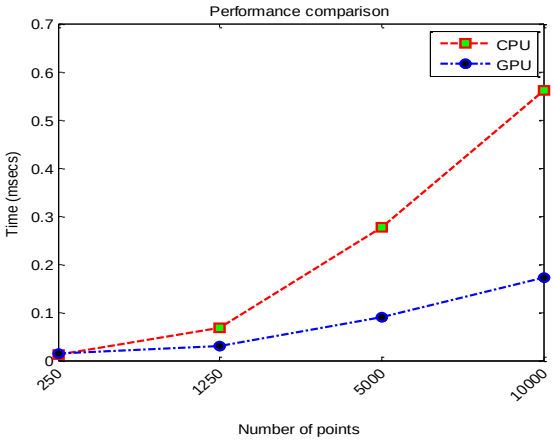


Figure 7. Performance of meshless FEM on GPU and CPU

In the second experiment, to assess the performance of the proposed method, we compare our GPU-based meshless FEM with an already optimized CPU implementation that utilized all available cores of our CPU platform. For this experiment, the bar model containing varying number of points (from 250 to 10000) was used with the time step value (*dt*) of 1/60. These timings only include the time for deformation using a single iteration and they do not include the time for rendering. These results are given in Fig. 7.
The results clearly show that the performance of the meshless FEM scales up well with the large datasets and we gain an acceleration of up to 3.3 times compared to an optimized CPU implementation. From this graph, it is clear that the runtime for CPU would be exponentially increased for larger datasets and the performance gap between the CPU and the GPU would be widened further.
It is the first time a meshless FEM model is applied to unstructured volumetric datasets. Our method uses the leap-frog integration which is semi-implicit whereas the existing mesh based FEM approaches use implicit integration schemes. The amount of time required for convergence in the case of implicit integration schemes is much more as compared to semi-implicit integration.

Moreover, the simulation system built using such formulation requires the stiffness matrix assembly which is then solved using an iterative solver such as Newton Raphson method or Conjugate Gradients (CG) method. Such stiffness assemblies are not required in meshless FEM. Therefore, ours converges much faster.

Nevertheless, for completeness, in the third experiment, we compared the performance of meshless FEM with an implicit tetrahedral FEM [ACF11]. These results are presented in Table 1.

Dataset	Tetrahedra	Frame rate (frames per second)	
		Implicit FEM	Meshless FEM
liver	3912	118.50-78.70	249.50-331.01
spx	12936	76.70-81.90	124.80-128.23
raptor	19409	40.30-43.80	71.22-71.71

Table 1. Comparison of meshless FEM against implicit tetrahedral FEM solver [ACF11]

As expected, the performance of meshless FEM is better as compared with the implicit tetrahedral FEM. Our meshless FEM is based on a semi-implicit integration scheme which does not require an iterative solver as is required for the implicit tetrahedral FEM.

6. DISCUSSION AND CONCLUSION

We have applied meshless FEM to nonrigid volumetric deformation. Using the proposed approach, interactive visualization of deformation on large volumetric dataset is made possible. We are confident of the results obtained from our experiments and would like to expand the model to address specific applications such as biomedical modeling [MCZLQS09], [MLQS09]; simulation [LSL96], [LSL97], [LSWM07]; fast ubiquitous visualization [YLS00], [ML12c]; and confocal imaging [TOTMLQS11].

We reiterate our main contributions. Firstly, we have applied meshless FEM for volumetric deformation. Secondly, we have integrated the meshless FEM into a novel deformation pipeline exploiting the transform feedback mechanism. And finally, we have integrated our novel deformation pipeline into the HAPT algorithm. There are some considerations on the meshless FEM. The deformation result is directly dependent on the number of mesh points used. More points generally give better approximation and vice versa. For better performance, we can think of two strategies. Firstly, we can use the image load-store extension in OpenGL 4.2 which allows shader programs to have access to arbitrary GPU memory. Since the hardware used in this study did not support OpenGL 4.2, this approach could not be verified. Secondly, we may use a CUDA kernel to do scattered writes, alongside a GLSL shader. This will possibly be a future research direction.

7. ACKNOWLEDGMENTS

This work is partially supported by a research grant (M408020000) from Nanyang Technological University and another (M4080634.B40) from Institute for Media Innovation, NTU. We would also like to thank the anonymous reviewers for their helpful suggestions and feedback.

8. REFERENCES

[ACF11] Allard J., Courtecuisse H., Faure F.: Implicit fem and fluid coupling on GPU for interactive multiphysics simulation. ACM SIGGRAPH 2011.

[ADLETG10] Andres D. L. C., Eliuk S., Trefftz G. H.: Simulating soft tissues using a GPU approach of the mass-spring model. Virtual Reality Conference (VR'2010), pp. 261–262, 2010.

[BBO03] Babuška I., Banerjee I., Osborn J. E., Survey of meshless and generalized finite element methods: A unified approach, Acta Numerica, 12, pp:1-125, 2003.

[Buc05] Buck I., Taking the plunge into GPU computing. Chapter 32 in GPU Gems 2, Matt Pharr and Randima Fernando (editors), 2005.

[CTA08] Comas O., Taylor Z., Allard J., Ourselin S., Cotin S., Passenger J., Efficient nonlinear fem for soft tissue modelling and its GPU implementation within the open source framework SOFA. International Symposium on Computational Models for Biomedical Simulation'08, pp. 28–39, 2008.

[DGW10] Dick C., Georgii J., Westermann R., A real-time multigrid finite hexahedra method for elasticity simulation using CUDA. In Simulation Modelling Practice and Theory'10, 19, No. 2, pp. 801–816, 2010.

[GEW05] Georgii J., Echtler F., Westermann R.: Interactive simulation of deformable bodies on GPUs. In Simulation and Visualization'05, 2005.

[GW05] Georgii J., Westermann R.: A multi-grid framework for real-time simulation of deformable volumes. Workshop on Virtual Reality Interactions and Physical Simulations'05, 2005.

[GW06] Georgii J., Westermann R., A generic and scalable pipeline for GPU tetrahedral grid rendering. IEEE Transaction on Visualization and Computer Graphics'06, 2006.

[HF00] Huerta A., Fernandez M. S., Enrichment and coupling of the finite element and meshless methods, International Journal for Numerical Methods in Engineering, 48, No. 11, pp:1615–1636, August 2000.

[LSL97] Lin F., Seah H. S., Lee Y. T., Structure Modeling and Context-Free-Grammar: Exploring a new approach for surface construction. Computers and Graphics (1997), 21, No. 6, pp. 777–785, 1997.

[LSL96] Lin F., Seah H. S., Lee Y. T., Deformable volumetric model and isosurface: Exploring a new approach for surface construction. Computers and Graphics (1996), 20, No. 1, pp. 33–40, 1996.

[LSWM07] Lin F., Seah H. S., Wu Z., Ma D., Voxelisation and fabrication of freeform models. Virtual and Physical Prototyping'07, 2 No. 2, pp. 65–73, 2007.

[MCG03] Mueller M., Charypar D., Gross M.: Particle based fluid simulation for interactive applications. In Proceedings of the ACM SIGGRAPH Symposium on Computer Animation (SCA'03), pp. 154–159, 2003.

[MCZLQS09] Movania M. M., Cheong L. S., Zhao F., Lin F., Qian K., Seah H. S., "GPU-based Surface Oriented Interslice Directional Interpolation for Volume Visualization," The 2nd International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL'09), Bratislava, Slovak Republic, November 24-27, 2009.

[MG04] Mueller M., Gross M., Interactive virtual materials. Proceedings of Graphics Interface (GI'04), pp. 239–246, 2004.

[MHSS04] Mosegaard J., Herborg P., Sangild Sorensen T.: A GPU accelerated spring mass system for surgical simulation. Health Technology and Informatics'04, pp. 342–348, 2004.

[MJLW07] Miller K., Joldes G., Lance D., Wittek A., Total Lagrangian explicit dynamics finite element algorithm for computing soft tissue deformation. Communications in Numerical Methods in Engineering'07, 23, No. 1, pp. 801–816, 2007.

[ML03] Mendoza C., Laugier C., Simulating soft tissue cutting using finite element models. IEEE International Conference on Robotics and Automation'03, pp. 1109–1114, 2003.

[ML12a] Movania M. M., Lin F., A novel GPU-based deformation pipeline. ISRN Computer Graphics, vol. 2012 (2012), p. 8.

[ML12b] Movania M. M., Lin F., Real-time physically-based deformation using transform feedback. Chapter 17 in The OpenGL Insights, Christophe, Riccio and Patrick, Cozzi (Ed.), AK Peters/CRC Press, 2012, pp. 233-248.

[ML12c] Movania M. M., Lin F., High-Performance Volume Rendering on the Ubiquitous WebGL Platform, 14th IEEE International Conference on High Performance Computing and Communications (HPCC'12), Liverpool, UK, 25-27 June, 2012.

[MLQS09] Movania M. M., Lin F., Qian K., Seah H. S., "Automated Local Adaptive Thresholding for Real-time Feature Detection and Rendering of 3D Endomicroscopic Images on GPU," The 2009 International Conference on Computer Graphics and Virtual Reality (CGVR'09), Las Vegas, US, July 13-16, 2009.

[MMF10] Maximo A., Marroquim R., Farias R., Hardware assisted projected tetrahedra. Computer Graphics Forum, 29, No. 3, pp: 903–912, 2010.

[NMK06] Nealen A., Mueller M., Keiser R., Boxermann E., Carlson M., Physically based deformable models in computer graphics. In STAR Report Eurographics 2006 vol. 25, pp. 809–836, 2006.

[NRBD08] Nguyena V. P., Rabczukb T., Bordasc S., Duflotd M., Meshless methods: A review and computer implementation aspects, Mathematics and Computers in Simulation, 79, No. 3, pp:763–813, December 2008.

[SB09] Sampath R., Biros G., A parallel geometric multigrid method for finite elements on octree meshes. In review, available online accessed in 2012 http://www.cc.gatech.edu/grads/r/rahulss/, 2009.

[ST99] Shapiro V., Tsukanov I., Meshfree Simulation of Deforming Domains," Computer Aided Design, 31, No. 7, pp: 459–471, 1999.

[TCO08] Taylor Z., Cheng M., Ourselin S., High-speed nonlinear finite element analysis for surgical simulation using graphics processing units. IEEE Trans. Medical Imaging, vol. 27, pp. 650–663, 2008.

[TE05] Tejada E., Ertl T., Large steps in GPU-based deformable bodies simulations. In Simulation Modeling Practice and Theory, 13 No. 8, Elsevier, pp. 703–715, 2005.

[TOTMLQS11] Thong P. S. P, Olivo M., Tandjung S. S., Movania M. M., Lin F., Qian K., Seah H. S., Soo K. C., "Review of Confocal Fluorescence Endomicroscopy for Cancer Detection," IEEE Photonics Society (IPS) Journal of Selected Topics in Quantum Electronics, Vol. PP, Issue 99, 2011. DOI: 10.1109/JSTQE.2011.2177447.

[VR08] Vassilev T., Rousev R., Algorithm and data structures for implementing a mass-spring deformable model on GPU. Research and Laboratory University Ruse 2008 (2008), pp. 102–109, 2008.

[YLS00] Yang Y. T., Lin F. and Seah H. S., "Fast Volume Rendering," Chapter 10 in Volume Graphics, Springer, January 2000.

[ZWP05] Zhong H., Wachowiak M., Peters T.: A real time finite element based tissue simulation method incorporating nonlinear elastic behavior. Computer Methods Biomechan. Biomed. Eng., 6, No. 5, pp. 177–189, 2005.

9. APPENDIX

$$\int_{V^e}(B\delta u^e)^T\sigma dV=(\delta u^e)^T q^e+\int_{A^e}(N\delta u^e)^T tdA$$

$$(\delta u^e)^T\int_{V^e}B^T\sigma dV=(\delta u^e)^T(q^e+\int_{A^e}N^T tdA)$$

$$\int_{V^e}B^T\sigma dV=q^e+\int_{A^e}N^T tdA$$

Substitution using Eq. (3)

$$\int_{V^e}B^T D\varepsilon dV=q^e+\int_{A^e}N^T tdA$$

$$\int_{V^e}B^T DBu^e dV=q^e+\int_{A^e}N^T tdA$$

$$B^T DBu^e\int_{V^e}dV=q^e+\int_{A^e}N^T tdA$$

$$B^T DBu^e V^e=q^e+\int_{A^e}N^T tdA$$

On Global MDL-based Multichannel Image Restoration and Partitioning

Tetyana Ivanovska
University of Greifswald, Germany
tetyana.ivanovska@uni-greifswald.de

Horst K. Hahn
Fraunhofer MEVIS, Germany
horst.hahn@mevis.fraunhofer.de

Lars Linsen
Jacobs University, Germany
l.linsen@jacobs-university.de

ABSTRACT

In this paper, we address the problem of multichannel image partitioning and restoration, which includes simultaneous denoising and segmentation processes. We consider a global approach for multichannel image partitioning using minimum description length (MDL). The studied model includes a piecewise constant image representation with uncorrelated Gaussian noise. We review existing single- and multichannel approaches and make an extension of the MDL-based grayscale image partitioning method for the multichannel case. We discuss the algorithm's behavior with several minimization procedures and compare the presented method to state-of-the-art approaches such as Graph cuts, greedy region merging, anisotropic diffusion, and active contours in terms of convergence, speed, and accuracy, parallelizability and applicability of the proposed method.

Keywords: Segmentation, Denoising, Minimum Description Length, Energy Minimization, Multichannel images.

1 INTRODUCTION

The goal of image partitioning is to detect and extract all regions of an image which can be distinguished with respect to certain image characteristics. A special form of image partitioning is image segmentation, where one or a few regions of given characteristics are separated from the rest of the image. If the underlying image is supposed to be piecewise constant, image partitioning is equivalent to the restoration of that image, which is often obtained using denoising algorithms. Although there exist plenty of methods for solving image partitioning, segmentation, and restoration problems, many of them are task-specific or require intensive user interaction and triggering of a large number of parameters.

In this paper, we restrict ourselves to generic solutions using automatic methods based on energy minimization. Hence, two tasks need to be tackled: First, one needs to formulate an energy functional based on some assumptions and, second, one needs to find an appropriate and computationally feasible minimization algorithm. The main emphasis of this work is put on analysis and discussion of an energy functional, based on the assumption that the processed image is multichannel, piecewise constant, and affected by white Gaussian noise. We analyse and compare several minimization procedures and draw analogies to other approaches. Color images are used as examples, as they document the algorithmic behavior in an intuitive manner. However, all steps work are applicable for any multichannel image data. The paper is organized as follows. In Section 2 the related work in this area is described. In Section 3 we formulate the energy functional and describe two minimization procedures. Our findings are presented and discussed in Section 4.

2 RELATED WORK

Global energy minimization approaches originate from such seminal works as the ones by Mumford and Shah [24] and Blake and Zisserman [2]. The Markov Random Fields (MRF) framework (see the seminal work of Geman and Geman [8]) is a stochastic branch of the energy minimization approaches. In the last years a great breakthrough has been done in the direction of Markov random fields methods [19]. Such methods as Graph Cuts [5] are mathematically well described and allow to find a solution that lies close to the global optimum.

The minimum description length (MDL) based approach [15] uses basic considerations from information theory [19] in the formulation of the energy term, in the sense that the best image partitioning is equivalent to obtaining the minimum description length of the image with respect to some specific description language.

There exist two main tendencies in further developement of this approach. The first one consists in extending the functional to be minimized. These extentions include either different noise models (not only white Gaussian noise), or elimination of the model parameters that must be manually tuned by a user.

Kanungo [12] et al. formulated the functional for multiband and polynomial images. Lee [17] considered correlated noise. Galland [7] et al. considered speckle, Poisson, and Bernoulli noise. Zhu and Yuille [31] proposed an algorithm combining region growing, merging, and region competition which permits one to segment complex images. In several approaches [20, 16], an extended version of the functional is used and all user-defined parameters are eliminated. For example, Luo and Khoshgoftaar [20] propose to use the well known Mean-Shift method [6] to obtain the initial segmentation and start the MDL-based region merging procedure from it.

These approaches utilize region growing, as the minimization of the extended functional is infeasible. However, such an iterative technique, generally, does not lead to a stable local minimum and can give much coarser results, when compared to the relaxation technique, if the procedure of region merging is non-reversible. Moreover, the authors state, for instance, in [20], that the initial region selection has a strong impact on the efficiency and effectiveness of the region merging.

The second direction is to use a simplified or limited model with some user-defined parameters, but apply a global minimization procedure, which allows one to find at least a stable local minimum. Kerfoot and Bresler [13] formulated a full MDL-based criterion for piecewise constant image partitioning, but the class of images is limited to the class of simply-connected ones. For minimization such methods as Graph Cuts [5] have gained in popularity in the last years.

3 METHODS

3.1 Multichannel Model Description

The fundamental idea behind the Minimum Description Length (MDL) [19, 23] principle is that any regularity in the given data can be used to compress the data.

The image partitioning problem with respect to the MDL principle can be formulated as follows: Using a specified descriptive language, construct the description of an image (code) that is simplest in the sense of being shortest (when coded, needs the least number of bits) [15]. Let $L(M)$ denote the language for describing a model M and $L(D|M)$ the language for describing data D given model M. Moreover, let $|.|$ denote the number of bits in the description. The goal is to find the model M that minimizes the code length $C_l = |L(M)| + |L(D|M)|$. This corresponds to the two-part MDL code [9]. If the a priori probabilities $P(M)$ of the described models are known, then the number of bits in the description equals the negative base-two logarithm of the probability of the described models [19]: $|L(M)| = -\log_2 P(M)$.

In terms of image partitioning and restoration the code length can be written as $C_l = |L(u)| + |L(z-u)|$,where the model we are looking for is the underlying image representation (or partitioning) u that minimizes the code length. The term z describes the initial (or given) image, and the difference $r = (z-u)$ between the given image z and the partitioning u corresponds to the noise in the image. The noise describes the data with respect to model u.

A simple implementation of the MDL principle for image partitioning was presented by Leclerc [15, 23]: he assumed a piecewise constant model and derived the functional (or energy term)

$$C_l = \frac{b}{2}\sum_{i\in I}\sum_{j\in N_i}\left(1-\delta\left(u_i-u_j\right)\right) + a\sum_{i\in I}\left(\frac{z_i-u_i}{\sigma}\right)^2, \quad (1)$$

where u denotes the underlying image, z the given image, and σ^2 the noise variance. Moreover, $\delta(u_i-u_j)$ denotes the Kronecker delta (1 if $(u_i = u_j)$, else 0), I denotes the range of the image, and N_i is the neighbourhood of the ith pixel, a and b are constants. The first term in Equation (1) encodes the boundaries of the regions, whereas the second term encodes the noise in form of uncorrelated white Gaussian noise.

One can observe the similarities between the functional in Equation (1) with constants a, b, σ and the energy considered in different MRF approaches, namely in the Graph Cut methods (see [5] for more details): $E(f) = \lambda \sum_{(p,q)\in N} V_{p,q}(f_p, f_q) + \sum_{p\in I} D_p(f_p)$, where the interaction potential V between pixels p, q having labels (colors) f_p, f_q is taken from the Potts model (which corresponds to the Kronecker deltas), D is the distance between the initial and current colors of the pixel p (which corresponds to the noise values), and λ is a constant.

We expand on this approach to derive a multichannel image description length. For encoding the model, i.e., deriving $L(u)$, we have to encode the boundaries of the regions. To do so, we calculate the number of pixels that contribute to the boundary. Hence, the codelength for the boundary encoding is given by $|L(u)| = \frac{b}{2}\sum_{i\in I}\sum_{j\in N_i}\left(1-\prod_{k\in Ch}\delta\left(u_i^k-u_j^k\right)\right)$, where k denotes the channel, Ch is the range of channels (e.g., RGB in the three-channel color case), and other notations are as above. To encode the data that do not fit the model, i.e., the noise, we derive $L(z-u)$ assuming that the values in each channel are subject to white Gaussian noise with parameters $(0, (\sigma^k)^2)$. This assumption implies that the noise between channels is not correlated. Such an assumption allows for better understanding of the underlying processes and is often sufficient for many applications, since the uncorrelated noise can appear during the transmission, storing, or manipulation of images [1]. Moreover, the assumption holds when a multichannel image is combined from different independent modalities for processing.

The codelength of the noise is derived as

$$|L(z-u)| = -\sum_{i\in I}\sum_{k\in Ch}\log_2 P(r_i^k) \qquad (2)$$
$$= -\sum_{i\in I}\sum_{k\in Ch}\log_2\left(\frac{q}{\sqrt{2\pi(\sigma^k)^2}}\exp\left(-\frac{(r_i^k)^2}{2(\sigma^k)^2}\right)\right)$$
$$= \frac{1}{2\ln 2}\sum_{i\in I}\sum_{k\in Ch}\left(\frac{r_i^k}{\sigma^k}\right)^2 + const$$

where $q=1$ is the pixel precision, *const* is an additive constant, which is discarded, if $\left(\sigma^k\right)^2$ is considered constant and equal for all channels.

The resulting codelength functional becomes $C_l = \frac{b}{2}\sum_{i\in I}\sum_{j\in N_i}\left(1-\prod_{k\in Ch}\delta\left(u_i^k-u_j^k\right)\right) + \frac{1}{2\ln 2}\sum_{i\in I}\sum_{k\in Ch}\left(\frac{r_i^k}{\sigma^k}\right)^2$. This functional corresponds to the one considered in the Graph cuts method up to the constant weights.

3.2 Minimization

Having formulated the energy functionals, one needs to minimize them for a given image in order to compute the image partitioning. As it has been shown that computing the global optimum even of the simplest functional is an NP-hard problem [5], in practice one has to look for efficient approximations for it.

In the current paper, we will deal with two optimization approaches for energy minimization: a discrete one and a continuous one, namely, the α-expansion Graph Cut algorithm, introduced by Boykov [5], and the GNC-like approach, introduced by Leclerc [15], with several modifications, which have been done to check the convergence.

Graph cuts is an efficient minimizing technique that allows for finding a local minimum within a known factor of the global one. This algorithm belongs to the class of discrete optimization. Here, we give the outline of the algorithm and refer the reader to the original paper by Boykov et al [5] for further details. Let S and $\mathscr{L}$ denote image pixels (lattice) and the palette (set of all possible colors), correspondingly. The labeling u is described as $\{\mathscr{S}_l | l\in\mathscr{L}\}$, where $\mathscr{S}_l = \{p\in S | u_p = l\}$ is a subset of pixels with assigned color l. Given a label α a move from a labeling u to a new labeling u' is called an α- expansion if $\mathscr{S}_\alpha \subset \mathscr{S}'_\alpha$ and $\mathscr{S}'_l \subset \mathscr{S}_l$ for any label $l\neq\alpha$. In other words, an α-expansion move allows any set of image pixels to change their labels to α [5]. The minimum of the energy E for each label α is found by constructing a graph and finding the minimum cut for it. It is efficiently done by the algorithm developed by Boykov and Kolmogorov [3].

Start with an arbitrary partitioning u
repeat
  Set *success* $\leftarrow 0$
  for all $\alpha\in\mathscr{L}$ **do**
    Find $\hat{u} = \arg\min E(u')$ among u' within one α-expansion of u
    if $E(\hat{u}) < E(u)$ **then**
      $u \leftarrow \hat{u}$
      success $\leftarrow 1$
    end if
  end for
until *success* $\neq 0$
Return u

A Relaxation method using ideas of Graduated Non Convexity (GNC) by Blake and Zisserman [2] was proposed by Leclerc [15]. This is a continuous optimization method, where the labeling u is not selected from the given palette, as in the Graph Cuts case, but is iteratively computed. Here, $u\in\mathbb{R}^n$. The basic concept of the minimization procedure is to replace the non-convex codelength functional $C_l(u)$ by an embedding in a family of continuous functions $C_l(u,s)$, where $s\in\mathbb{R}$ is a user-defined parameter, that converge towards the target functional $C_l(u)$ when s goes to zero. $\lim_{s\to 0} C_l(u,s) = C_l(u)$. For the starting value of s, the functional $C_l(u,s)$ is convex such that standard convex minimization procedures can compute the single minimum. When s approaches zero, number and positions of the local minima of $C_l(u,s)$ become those of C_l. The minimization procedure iterates over s, which steadily decreases, and minimizes $C_l(u,s)$ for the respective value of s in each iteration step.

To obtain a continuous embedding, the discontinuous parts in functional C_l need to be replaced by a continuous approximation. The discontinuity of C_l is due to the use of the function δ. Hence, function δ is replaced by a continuous approximation that converges to δ when s goes to zero [15]. We use the approximation $\delta\left(u_i^k-u_j^k\right)\approx\exp\left(-\frac{\left(u_i^k-u_j^k\right)^2}{\left(s\sigma^k\right)^2}\right) = e_{ij}^k$.

The minimization iteration starts with a sufficiently large value $s=s^0$ and computes the (global) minimum u^0 of the convex functional $C_l(u,s^0)$. In each iteration step $T+1$, we set $s^{T+1} = rs^T$, where $0<r<1$, and compute the local minimum of $C_l(u,s^{T+1})$ starting from minimum u^T of the previous iteration step. The iteration is repeated until s is sufficiently small, i.e., until $s<\varepsilon$ with ε being a small positive threshold.

To compute the local minimum u on each iteration we apply Jacobi iterations [26]. The functional $C_l(u,s)$ is convex, if we choose a value for s that satisfies $(x_i^k - x_j^k)^2 \le 0.5\left(s\sigma^k\right)^2$ for all pixels i and j with $i\neq j$) and all channels k. Hence, when this condition is met, the local minimum must be a global one. The condition needs to be fulfilled for the starting value $s=s^0$. Then, the condition $\frac{\partial C_l(u,s^T)}{\partial u_i^k} = 0$for the local minimum at iteration step T becomes

$$\frac{2a\left(u_i^k-z_i^k\right)}{\left(\sigma^k\right)^2} + \frac{b}{2}\sum_{j\in N_i}\left[\frac{2\left(u_i^k-u_j^k\right)}{\left(s^T\sigma^k\right)^2}\prod_{l\in Ch} e_{ij}^l\right] = 0, \qquad (3)$$

where constant $a = (2\ln 2)^{-1}$. As Equation (3) cannot be solved explicitly, we use an iterative approach, where at each iteration step $t+1$ we compute

$$u_i^{k,t+1} = \frac{z_i^k + \frac{b}{a(s^T)^2} \sum_{j \in N_i} u_j^{k,t} \prod_{l \in Ch} e_{ij}^{lt}}{1 + \frac{b}{a(s^T)^2} \sum_{j \in N_i} \prod_{l \in Ch} e_{ij}^{lt}}. \quad (4)$$

In total, we have two nested iterations. The outer iteration denoted by T iterates over s^T, while the inner iteration denoted by t iterates over $u_i^{k,t+1}$. Considering the behavior of the exponential function, the termination criterion for the inner loop is given by $\left| u_i^{k,t+1} - u_i^{k,t} \right| < s^T \sigma^k, \quad \forall i \in I$. Starting with $u = z$, the minimization procedure can be summarized by the following pseudocode:

while $s \geq \varepsilon$ **do**
  start with local minimum for u found in previous iteration
  while termination criterion for u is not met **do**
    recalculate u using Equation (4)
  end while
  update s
end while

Comparison to Anisotropic Diffusion. The derived iterative scheme for computing u_i^{t+1} in the single-channel case, cf. [15], is similar to the iteration in the well-known anisotropic diffusion approach. The continuous form of the Perona-Malik equation [25] for anisotropic diffusion is given by

$$\frac{\partial I}{\partial t} = div(g(\|\nabla I\|) \cdot \nabla I), \quad (5)$$

where I denotes the image and function g is defined by $g(\|\nabla I\|) = \exp\left(-\left(\frac{\|\nabla I\|}{K}\right)^2\right)$ with flow constant K. The discrete version of Equation (5) is given by $I_i^{t+1} - I_i^t + \lambda \sum_{j \in N_i} \left(I_i^t - I_j^t\right) \exp\left(-\left(\frac{I_i^t - I_j^t}{K}\right)^2\right) = 0$, where λ is a normalization factor.

The local minimum of the MDL-based energy C_l in single-channel notation (for a fixed s) is defined by solving $\frac{\partial C_l(u,s)}{\partial u_i} = 0$, which leads to

$$(u_i - z_i) + \alpha \sum_{j \in N_i} \left[(u_i - u_j) \exp\left(-\frac{(u_i - u_j)^2}{(s\sigma)^2}\right)\right] = 0.$$

The continuous version of this equation can be written as

$$\int_0^{t_{end}} \frac{\partial u}{\partial t} dt = div(g(\|\nabla u\|) \cdot \nabla u), \quad (6)$$

where $u_0 = z$ and $u_{t_{end}}$ describes the image values at the current step t_{end}. Comparing the continuous form of the Perona-Malik equation (5) with the continuous form of the MDL-based equation (6), one can immediately observe the similarity. The main difference is the integral on the left-hand side of Equation (6). The integral represents the changes between the current image and the initial image. Thus, this version of the MDL-based minimization algorithm can be considered as an "anisotropic diffusion algorithm with memory".

In general, the assumptions about the convexity of $C_l(u,s)$ do not hold anymore, when s is small. Hence, the relaxation method with Jacobi iterations can give an inadequate result. To check this we implemented the steepest descent minimization scheme [26] and compared the results.

4 RESULTS AND DISCUSSION

The presented algorithm belongs to the class of algorithms that are not purely for denoising or segmentation, but can be treated as an elegant combination of both approaches. Usually, some part of meaningful image data might be lost on the denoising step, which affects the subsequent segmentation results. Here, on the contrary, the complete information is used both for boundary preservation and noise exclusion, which is referred in the literature as image restoration [28] or reconstruction [25].

The algorithm (minimization procedures with Jacobi iterations, Gradient descent, and Graph cuts) was implemented using C/C++ programming language, optimized appropriately, and was compiled under gcc4.4. The Graph cuts code for grayscale images was kindly provided on the Vision.Middlebury web-site [22].

Figure 1 documents the general behavior of the Relaxation scheme. Starting with a synthetic 100×100 image with manually added noise, we apply the Jacobi iteration minimization procedure. The three rows in Figure 1 show (intermediate) results at iteration $T = 4$, $T = 600$, and $T = 1484$, respectively. The first column shows the currently detected underlying image and the second column the currently removed noise, i.e., the difference between the initial image and the currently detected underlying image (first column of Figure 1). For the generation of the images in column three, we picked one row of the image, i.e., a horizontal cut through the image, and show the current values of u for that single row. The third column in Figure 1 documents nicely, how individual values (first row) start assimilating and grouping together (second row) to eventually form a piecewise constant representation (third row).

To qualitatively describe the restoration results, we evaluated the difference between the resulting and initial (not noisy) images using two measures. The first one is the mean squared error (MSE) estimator [18], which is suitable to show whether the colors were reconstructed as close to

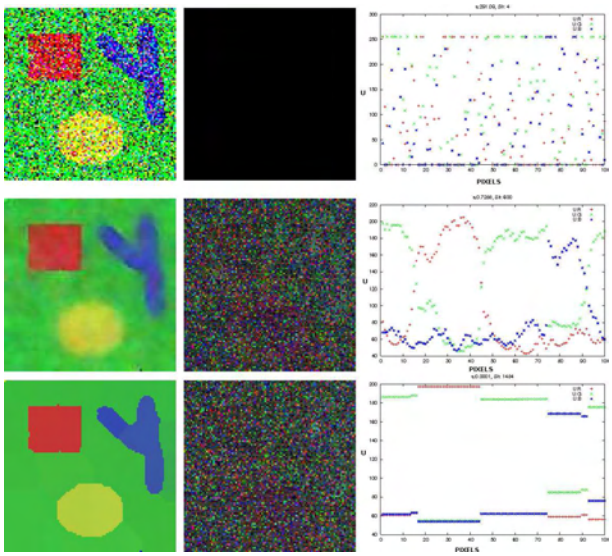


Figure 1: Different phases of MDL minimization on timesteps 4, 600, and 1484. The images from left to right are: the detected underlying image, the removed noise, image values u of one row (The pixel coordinates are given in x axis, and u values are in y axis).

the initial ones as possible, i.e., for the denoising part. For the multichannel case, MSE is defined as $MSE = \frac{1}{cwh}\sum_{c_1\in c}\sum_{y\in h}\sum_{x\in w}(In(x,y,c_1)-Res(x,y,c_1))^2$, where c is the number of color channels, w is the image width, h is the image height, and In and Res define the initial and resulting image, respectively.

The second metric is utilized to evaluate the segmentation results, i. e., oversegmentation and inaccurate boundary localization. It is based on the one proposed by Mezaris et al [21]. This metric is defined as follows. Let $S = s_1, s_2, ..., s_K$ be the segmentation mask to be evaluated, comprising K regions s_k, $k = [1,K]$, and let $R = r_1, r_2, ..., r_Q$ be the reference mask, comprising Q reference regions r_q, $q = [1,Q]$. Each region r_q is associated with a different region s_k, i. e. s_k is chosen such that the overlap $r_q \cap s_k$ is maximized. Let $A(r_q, s_k)$ denote the set of region pairs and let N_S denote the set of non-associated regions of mask S. Energy E_b is used for the evaluation, values closer to zero indicate better segmentation: $E_b = \sum_{q=1}^{Q} E_q + \sum_{s_k \in N_S} F_k$, where E_q and F_k are defined are follows.

$$E_q = \sum_{p\in(r_q - r_q\cap s_k)} f_1(p,r_q) + \sum_{p\in(s_k - s_k\cap r_q)} f_2(p,r_q) \quad (7)$$

$$F_k = \alpha \sum_{p\in s_k} f_1(p,r_q) \quad (8)$$

E_q is a weighted sum of misclassified pixels for region pair (r_q, s_k) $\forall (r_q, s_k) \in A$. F_k is a weighted sum of misclassified pixels $\forall s_k \in N_S$. f_1 and f_2 are weight functions, proposed by Villegas et al [30] to deal with the fact that the distance of a misclassified pixel from the boundary of the reference region to which it belongs affects the visual relevance of the error [21]. f_1 is used for false negatives and f_2 is used for false positives.

$$f_1(p,r_q) = d(p,r_q) * 10^{-4},$$
$$f_2 = \begin{cases} d(p,r_q) * 10^{-4} & d(p,r_q) < 10 \\ 10^{-3} & otherwise \end{cases}.$$

Moreover, d is the Euclidean distance between the pixel p and the boundary of the region r_q. α is a weight parameter which was heuristically set to 100 in our experiments, since we would like to penalize oversegmentations.

First, we run a series of tests on synthetic images with artificially added Gaussian noise applying three algorithms: Graph Cuts, Relaxation with Jacobi iterations, and Relaxation with steepest descent. Figures 2 and 3 demonstrate the high-quality results produced by both Graph Cuts and Relaxation with gradient descent methods. The exact regions are reconstructed with few misclassifications. Graph Cuts restore the closest colors to the initial image ($MSE = 830.65$).

However, the computational costs of these methods are rather high. For the Graph cuts method the execution time is dependent on the palette size, i. e., the number of colors (labels) which would be considered for expansion. Ideally, the palette should include the whole color space. In this case, the optimum will be found accurately. Boykov et al. proposed to reduce the space of labels for this case [4], taking, for instance, only unique colors of the image and their closest neighbors as labels. Such a choice allows for a massive label space reduction, but leaves enough variability for the label expansion. In our experiments the computation for a 100×100 color image with $50,000$ labels (unique colors and their closest neighbours of a noisy synthetic image) took around 10 minutes per iteration. As usually several iterations are needed, this approach appears to be infeasible for bigger datasets.

The relaxation scheme with the Gradient Descent method for a 100×100 color image takes in several hours, which makes this method inapplicable for real datasets.

The relaxation scheme with Jacobi iterations restores more regions than required, however, they have very close colors and the "weak" region boundaries can not be distinguished by a human eye. Due to such color closeness, these regions can be merged into one using a simple region-merging procedure. Starting at any pixel (seed point), the algorithm subsequently adds neighbouring pixels to the region if the distance in color space between them and the region color is lower than a certain threshold. The region color is the average color of the pixels that belong to it. The algorithm stops when all pixels belong to some regions. Computational costs of the region merging procedure are negligible

when compared to the ones of the Relaxation method. When the simple region merging is applied, we obtain $E_b = 0.0021$, which is slightly better than the other results.

The main advantage of the Jacobi iteration approach is its independence of the palette size and straightforward parallelizability, such that computations only take seconds [11], which makes this method attractive for real applications.

Next, we would like to compare the presented method to other (similar) approaches, namely, anisotropic diffusion, region merging, and a combination of these two procedures. Although anisotropic diffusion is primarily used to remove noise from digital images without blurring edges, it can be used in edge detection algorithms. By running the diffusion with an edge seeking diffusion coefficient for a certain number of iterations, the image can be evolved towards a piecewise constant image with the boundaries between the constant components being detected as edges [25]. We compared our approach to the anisotropic diffusion algorithm for multichannel images, presented by Sapiro and Ringach [27]. This method similarly to ours considers the influence of all image channels at once, which allows for better boundary preservation when compared to the methods treating each image channel separately. However, our approach has a "memory", as it always refers to the initial image instead of iterating from the result obtained on the previous time step. Such a behavior should guarantee a better boundary preservation and initial color restoration if the parameters are properly chosen. Figure 4 demonstrates that Sapiro's method successfully eliminates the noise (except from some mistakes on the image boundaries) and preserves most of the object boundaries. Our approach appears to be stable to noise, preserves object boundaries, restores the colors close to the initial ones, and produces the most uniformly colored background.

Due to the energy functional complexity, which arises for the models with no manually adjustable parameters or for more complicated noise models, it is often difficult to find a feasible minimization procedure. Thus, many authors follow a "greedy" region merging strategy [12, 20, 14, 16]. The idea of the algorithm is as follows. It starts from some initial oversegmentation. Then at each timestep it chooses two neighbouring regions and merges them to form a new region. These two regions are chosen in such a way that, when they are merged, it provides the largest energy reduction amongst all other possible merges. Although this procedure is very fast, it has several drawbacks. First, it does not guarantee convergence to a stable local minimum, especially, in the presence of noise. Second, the way the initial oversegmentation is chosen also affects the results.

Although Kanungo et al. [12], Luo and Khoshgoftaar [20], Lee [16], and Koepfler et al. [14] propose to use different functionals for color image segmentation, the region merging procedures are similar to each other. We assume that a more complicated functional could lead to a more adequate result. However, there is no guarantee about the convergence to a stable local minimum, and this issue stays unresolved.

We compared our approach to the algorithm proposed by Koepfler et al. [14]. They use the Mumford-Shah functional for the piecewise constant case, which coincides with the MDL-based functional C_l. Here, we computed both metrics to evaluate the results. As it can be observed in Figure 5, the iterative region merging produces results with some misclassified boundaries, which appears due the fact that the merging algorithm did not land close to the minimum, and not completely erased noise when compared to Figure 4. The relaxation approach can be approximated with a pipeline that utilizes several algorithms, namely, denoising, boundaries sharpening, and region merging. We constructed a pipeline from Sapiro's anisotropic diffusion and Koepfler's region merging. As it can be observed in Figure 5, the pipeline produces the best result in terms of boundary preservation and noise elimination when compared to the independent application of these methods. However, in general, each method in the pipeline requires additional adjustment of the parameters for each image.

Our comparison shows that the proposed approach produces the best results in terms of color (the lowest $MSE = 377.628$) and region restoration (the lowest $E_b = 0.0008$) for the test image.

We also compared our results to the results obtained with the Active contours without edges [29] method introduced by Chan and Vese. This is a variational approach based on energy minimization, and the energy is formulated using the Mumford-Shah functional for the piecewise constant case. For our tests, we applied the 4-phase version of the algorithm for color piecewise constant images with the parameters for each phase:$\lambda_{1,2} = 1$, $\nu = 0$, as it is recommended by the authors. We experimented with different values of parameter μ and initial contour locations. We executed the algorithm with max. 2000 iterations with the $timestep = 0.5$. Figure 6 documents that it is problematic to obtain decent results for images with low signal-to-noise ratio for the reasonable amount of time and the prior denoising is needed.

When applied to real images, our approach allows for obtaining the results with different levels of detail, which reminds one of the multi-scale theory of piecewise image modeling, introduced by Guigues et al. [10]. In Figure 7 the results illustrate this effect. Increasing the value of b allows us to reduce the

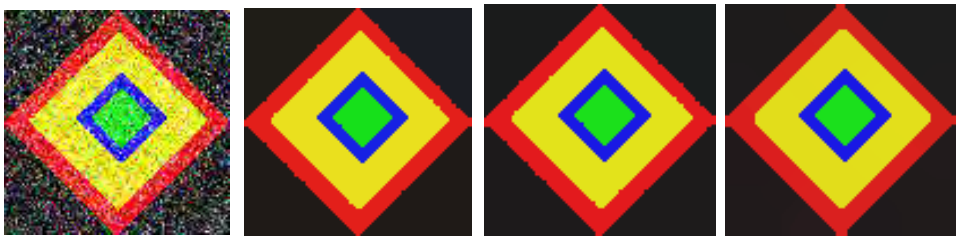


Figure 2: *First*: Input diamand image with manually added noise $N(0,70^2)$. *Second*: Graph cuts result. $MSE = 830.65$. $E_b = 0.0027$ *Third*: Relaxation (Steepest descent on inner iterations) result. $MSE = 2273.8096$. $E_b = 0.0031$ *Fourth*: Relaxation (Jacobi iterations) result. $MSE = 2418.262$. No region merging $E_b = 651.287$. With region merging $E_b = 0.0021$

Figure 3: *First*: True image with 8 regions (5 unique colors). *Second*: Piecewise-constant regions are marked with random different colors on the Graph cuts result. The image consists of 8 regions (7 unique colors). *Third*: Regions are marked with random different colors on the Relaxation (Descent) result. The image consists of 8 regions. *Fourth and Fifth*: Piecewise-constant regions are marked with random different colors on the Relaxation (Jacobi) result. The image consists of 49 regions (before region merging) and 8 regions (after region merging).

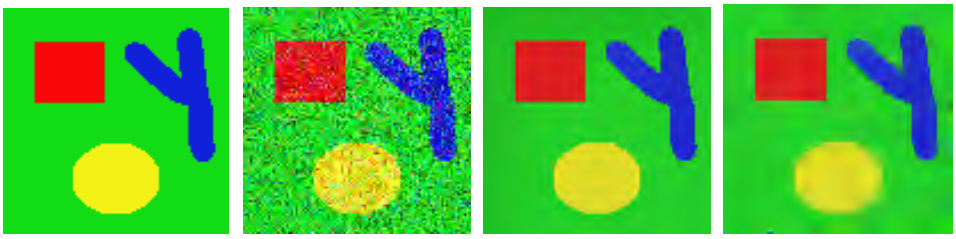


Figure 4: *First*: Initial synthetic image. *Second*: Synthetic image with added noise $N(0,70^2)$. *Third*: Result for our approach with subsequent region merging. $MSE = 377.628$. $E_b = 0.0008$. *Fourth*: Result for Sapiro's anisotropic diffusion with parameters: $edge = 49$, $numStep = 45$. $MSE = 447.83$.

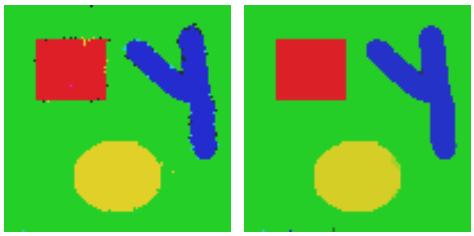


Figure 5: *First*: Result for iterative region merging with $\lambda = 10^6$. Noise is not erased completely. Further increasing the parameter v_0 does not improve the result. $MSE = 436.538$. $E_b = 6.5369$ *Second*: Result for the pipeline consisting of the anisotropic diffusion and region merging based on the Mumford-Shah functional. $MSE = 419.7864$. $E_b = 0.26$

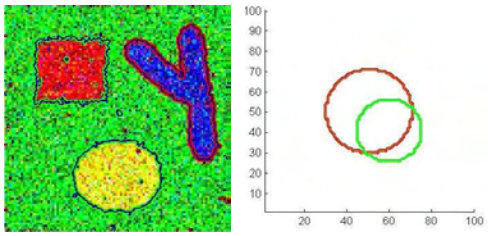


Figure 6: *First*: Result for Active contours. The resulting contours are still noisy. *Second*: Initial contours.

number of details, i.e., the number of regions, and leave only the "strongest" edges.

5 CONCLUSION AND FUTURE WORK

We generalized the existing single-channel MDL-based method to a multichannel one and applied it to color image partitioning and restoration. For synthetic color images, the results were compared to the ones obtained with Graph cuts, anisotropic diffusion, greedy region merging with energy, and active contours approaches. Our method produces best results in terms of color

Figure 7: Results with different *b* for the "woman" image. The initial image is the leftmost one. The parameter *b* is selected: 2, 5, 20 for the images from left to right, correspondingly.

restoration and boundary accuracy for our test cases. Moreover, the relaxation scheme with Jacobi iterations combined with a simple region merging procedure allows for fast computations due to straightforward parallelizability, which is important when applied to real-world problems.

We believe that this approach has potential in solving problems of image restoration. The future work directions include studying different noise models (e. g. correlated noise) for piecewise constant and piecewise smooth images as well as the models based on other MDL codes.

6 REFERENCES

[1] D. Baez-Lopez, F. H. Mendoza, and J. M. Ramirez. Noise in color digital images. *Circuits and Systems, Midwest Symposium on*, 0:403, 1998.

[2] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, 1987.

[3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[4] Y. Boykov, O. Veksler, and R. Zabih. Efficient restoration of multicolor image with independent noise. Technical report, 1998.

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[6] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[7] F. Galland, N. Bertaux, and P. Refregier. Multi-component image segmentation in homogeneous regions based on description length minimization: Application to speckle, poisson and bernoulli noise. *Pattern Recognition*, 38(11):1926 – 1936, 2005.

[8] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):721–741, 1984.

[9] P. Grunwald. *A tutorial introduction to the minimum description length principle*. 2004.

[10] L. Guigues, J. Cocquerez, and H. Le Men. Scale-sets image analysis. *International Journal of Computer Vision*, 68:289–317, 2006.

[11] T. Ivanovska, L. Linsen, H. K. Hahn, and H. Voelzke. Gpu implementations of a relaxation scheme for image partitioning: Glsl vs. cuda. *Computing and Visualization in Science*, 14(5):217–226, 2012.

[12] T. Kanungo and B. Dom et al. A fast algorithm for mdl-based multi-band image segmentation. *Computer Vision and Pattern Recognition*, pages 609–616, 1994.

[13] I.B. Kerfoot and Y. Bresler. Theoretical analysis of multispectral image segmentation criteria. *Image Processing, IEEE Transactions*, 8(6):798–820, 1999.

[14] G. Koepfler, C. Lopez, and M. Morel. A multiscale algorithm for image segmentation by variational method. *SIAM Journal of Numerical Analysis*, 31:282–299, 1994.

[15] Y.G. Leclerc. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision*, 3(1):73–102, 1989.

[16] T. C. M. Lee. A minimum description length-based image segmentation procedure, and its comparison with a cross-calidation-based segmentation procedure. *Journal of the American Statistical Association*, 95(449):259–270, 2000.

[17] T. C. M. Lee and T. Lee. Segmenting images corrupted by correlated noise. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:481–492, 1998.

[18] E. L. Lehmann and George Casella. *Theory of Point Estimation (Springer Texts in Statistics)*. Springer, 2003.

[19] S. Z. Li. *Markov random field modeling in image analysis*. Springer-Verlag New York, Inc., 2001.

[20] Q. Luo and T.M. Khoshgoftaar. Unsupervised multiscale color image segmentation based on mdl principle. *IEEE Transactions on Image Processing*, 15(9):2755–2761, 2006.

[21] V. Mezaris, I. Kompatsiaris, and M. G. Strintzis. Still image objective segmentation evaluation using ground truth. In *In Proceedings of the Fifth COST 276 Workshop on Information and Knowl-*

edge Management for Integrated Media Communication, 2003.

[22] http://vision.middlebury.edu/MRF/ Middlebury Vision.

[23] A. Mitiche and I. B. Ayed. *Variational and Level Set Methods in Image Segmentation*. Springer Berlin Heidelberg, 2011.

[24] D. Mumford and J. Shah. Boundary detection by minimizing functionals. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1985.

[25] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(7):629–639, 1990.

[26] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, 2002.

[27] G. Sapiro and D. L. Ringach. Anisotropic diffusion of multivalued images with applications to color filtering. *Image Processing, IEEE Transactions on*, 5(11):1582–1586, 1996.

[28] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(6):1068–1080, 2008.

[29] L. A. Vese and T. F. Chan. A multiphase level set framework for image segmentation using the Mumford and Shah model. *International Journal of Computer Vision*, 50(3):271–293, December 2002.

[30] P. Villegas, X. Marichal, and A. Salcedo. Objective evaluation of segmentation masks in video sequences. In *In Proceedings of Workshop om Image Analysis for Multimedia Interactive Services*, 1999.

[31] S.C. Zhu and A.L. Yuille. Region competition: unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE Trans. on PAMI*, 1996.

A Study on the Animations of Swaying and Breaking Trees based on a Particle-based Simulation

Yasuhiro AKAGI
Tokyo University of Agriculture and Technology
Naka-cho 2-24-16
184-8588, Koganei, Tokyo
scout@cc.tuat.ac.jp

Katsuhiro KITAJIMA
Tokyo University of Agriculture and Technology
Naka-cho 2-24-16
184-8588, Koganei, Tokyo
Kitajima@cc.tuat.ac.jp

ABSTRACT

In this paper, we propose a particle-based simulation method to create the animations of swaying and breaking trees. Since the shapes of trees and their realistic motions are usually complex, it is difficult for us to create an animation of a swaying tree manually. Therefore, to produce films and image contents which contain a natural scene of swaying and breaking trees, it takes a lot of work to create the animation. To solve this problem, it is important that how to calculate interactions between a tree and wind to automatically generate the swaying and breaking motions of a tree by using a physical simulation. We model both a tree and wind as particles to simulate the interactions. The advantage of the particle-based method is that the method is robust for changing of the topology of wind and the branching structure of a tree. Our results show that the proposed method can naturally represent the breaking behavior of a tree and the wind flow around the tree by using the particle-based simulation of the wind.

Keywords

Tree animation, Breaking branches, Fallen leaves, Particle-based simulation, SPH method, Elastic deformation

1. INTRODUCTION

The 3D modeling and creation of animations of natural scenes is one of the most time-consuming works of producing films and videos[Chn11]. It is difficult to manually create an animation of swaying trees that has realistic movements of branches and leaves in an environment having a complex wind flow. Therefore, there are many studies on the animation of swaying trees on the basis of tree dynamics and wind simulations. We propose a practical method of generating tree animations that simulate the swaying motions and breaking behaviors of trees by using a particle-based model. We represent both the trees and wind by particles and links to be able to simulate the breaking behaviors of trees. The most advantage of the method using the particles is that the simulation method can freely divide the tree structure into broken branches and fallen leaves. This method also represents the change of the movement which is caused by the breaking and changing the shape of the tree.

Previous work

Most of these studies propose a method of simulating the dynamics of branches by using beam structures[AWZ09]. These methods simulate the bending motions of the beam structures (branches) on the basis of external forces (wind). For instance, Habel [HKW09] and Yang[YHY+10] proposed a method of automatically animating swaying trees using beam structures. Diener [DRB+09] proposed a simpler method of simulating bending branches. This method divides a tree into the sections and simulates the movements of branches only by the rotation of these sections. The method enables us to generate tree animations that contain thousands of trees in real time. Then, our method simulates the deformation of the beam structure of a branch by using the particles and connection of the particles.

In recent years, there are several studies on the simulation of the breaking behaviors of branches and falling leaves for generating tree animations [BCN+10]. Saleem[SCZ+07] proposed a method that simulates the breaking behaviors of branches. In this method, the breaking positions of a branch are decided on the basis of the threshold velocity, which is the velocity of the wind around the branch. When the breaking position of a branch is detected, the method detaches the broken branch from the tree and simulates the falling motion of the branch. Since the threshold velocities are set by the users beforehand, it takes a considerable amount of time to create the tree animations[Chn11]. Moreover, this method cannot

detect the broken positions of branches and cannot generate the cross-sectional shape of a broken branch automatically.

On the other hand, to generate fluid animations, many studies on the simulation of complex movements such as broken waves, splashing, and pouring have been conducted [AT11][CBP05]. These studies are based on the SPH method[Luc77], which is a particle-based fluid dynamics method. One of the features of the SPH method is that it represents a fluid as a set of particles that makes it easy to simulate the separation and fusion of the fluid. Another feature of the SPH method is that it is easy to parallelize. With the use of this method, Goswami[GSS+10] succeeded in accelerating the generation of fluid animations by using GPUs. Stava et.al.[SBB+08] proposed "Surface particles" to simulate the erosion of a terrain by defining the relationship between particles and terrain.

Moreover, there are several studies based on the particle method for simulating the deformations of elastic objects. Muller[MC11] proposed a particle-based simulation method for elastic objects. One of the features of this method is that it effectively represents elastic objects by using ellipsoid particles. Gerszewski[GBB09] also succeeded in deforming elastic objects robustly by using particle-based models. Since the particle-based approach has the advantage of geometric changes, there are several studies on the destruction simulation of solid objects. Pauly[PKA+05] succeeded in simulating the destruction of solid objects by replacing the finite element meshes of an object with particles. [WRK+10] proposed a method for simulating deformation and fracturing of elastic objects. This method efficiently simulates the elastic objects with remeshing technique. Since the trees have complexly branched structures and intersections, we use the particle-based method to simulate the deformation and fracturing of trees to prevent the miss generation of meshes.

2. PHYSICAL SIMULATION OF SWAYING AND BREAKING TREES

In this section, we propose a simulation method to create an animation of swaying and breaking trees. This method is based on a particle-based simulation of wind and elastic deformations. We model both wind and trees as particles that have the same physical parameters in order to handle the interactions between the wind and the trees independently. The advantages of using the particle-based method are as follows:

1. If the physical parameters of the particles, such as position and velocity, are discrete, the interactions between the particles can be simply simulated as a particle-to-particle problem.
2. The particle-based structure has an advantage with respect to a breaking behavior that contains a topology change.
3. As the particles are discrete, it is easy to speed-up the simulation by using a parallel processing technique.

	Parameter	Symbol
Particle Prt_i	Position	x_i
	Velocity	u_i
	Acceleration	a_i
	Interaction Radius	r_i
	Mass	m_i

Table 1. Physical parameters of a particle

In the following sections, we describe the equations of our simulation by using the symbols given in Table 1.

Types of interaction models

When we perform a particle-based physical simulation, it is important to define the interactions between particles. In order to distinguish between the physical characteristics and interactions of the wind and the trees, we define the following four types of particles:

Wind : wind particle

Branch: particle for a part of a branch

Leaf : leaf particle

Bone : particle for the bone structure of a tree

We define three of the abovementioned types of particles for representing a tree. Two of these three are the types that help distinguish between the branches and the leaves. The fourth one represents the structure of a tree, such as the bone structure for a CG character. This bone particle is used for calculating the restoring force. We will describe the effect of this type of particles in Section 2.

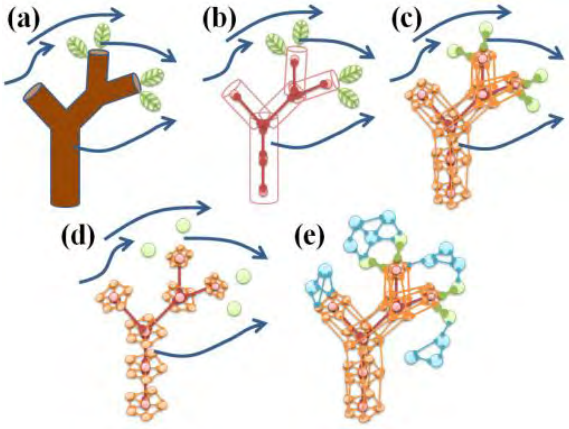


Figure 1. Generation process of the particle and link-structure based tree model form a polygon-based tree model

Particle-based tree modeling

In this section, we describe a method of modeling trees on the basis of the abovementioned four types of particles. To simulate the swaying motions and breaking behaviors of a tree, we define a link structure of the particles that are used for replacing the 3D tree model. The link structure represents the relationships between the particles for detecting an interaction force between two particles. In the proposed method, we require two types of information for the 3D tree model. One is the information regarding the differences between branches and leaves, and the other is the information regarding the tree structure and the positions of the branching points. By using the information of the tree, the proposed method generates the particles and link structures by following steps:

		Destination			
		Wind	Branch	Leaf	Bone
Source	Wind	f_{wind}	f_{wind}	f_{wind}	f_{wind}
	Branch	f_{wind}	f_{spr}	f_{spr}	f_{spr}
	Leaf	f_{wind}	f_{spr}	-	-
	Bone	f_{wind}	$f_{spr} + f_{shr}$	-	f_{bnd}

Table 2. Interaction model

(a) Define central axis

We use the links that are named the "central axis" [TMW02]. The central axis connects two particles to form the bone structure of a tree. We generate these links on the basis of the information regarding the positions of the branching points of the 3D tree model. Then, the proposed method generates subdivision particles at regular intervals in order to calculate the bending of a branch.

(b) Replace branches and leaves with particles

The particles for representing branches are generated on the circumference of a circle whose center is formed by the particles of the central axis generated in step (a). The radius of the circle is the same as that of the branch, and the normal vector of the circle is the same as the direction of the central axis. The particles for representing the branches are generated on the central position of the leaf.

(c) Generate link structure

We generate the link structure of a tree on the basis of the particles that are generated in the previous step. Particles for various parts of a branch are linked side by side on the circumference of the circle and are linked to the particles of the central axis, which are the center particles of the circle. These particles are also linked to particles that are linked to the neighboring circles. The leaf particles are linked to the nearest particles of the parent branch.

(d) Define particles and links for wind

We define the cubic area for simulating the wind, and the wind particles are generated in this area at regular intervals. The links from a wind particle are generated on the basis of the influence radius of the particle, and the links are updated in each time step of the simulation.

Next, we define the four types of interaction forces as follows:

f_{wind} : Force of the wind from the fluid simulation
f_{spr} : Internal force of a tree based on a mass-spring system
f_{bnd} : Restoring force from the central axis
f_{shr} : Shear stress from the structure of a branch

Table 2 shows the relationships between the four types of particles and the four types of interaction forces. The names of the types in the first column are the source of the interaction force, and those in the second row from the top are the destination.

The wind particles affect all types of interactions. We calculate the interaction force f_{wind} using the Navier-Stokes equations. Further, we simulate the elastic deformations of a tree by using three types of interaction forces. We will describe the method for simulating the elastic deformations in fifth Section.

Wind simulations

The interactions from the wind particles are calculated using the Navier-Stokes equations, which are one of the fundamental equations in the field of fluid dynamics. We apply the SPH method, which is a numerical solution of fluid dynamics, to simulate the airflow around a tree. Since the SPH method handles both fluids and objects as particles, it can be used for simulating the interactions between the wind and the trees. Equation (1) gives the interaction force $f_{wind\,i}$, which is the force exerted by all types of particles on particle Prt_i.

$$f_{wind\,i} = \mu\nabla^2 u_i - \nabla p_i + \rho_i f \quad (1)$$

μ: viscosity, p_i: pressure,

ρ_i: density, f: external force

The first factor on the right side of equation 1 is the viscosity factor and the second factor is the pressure factor. The following equations are the differential equations for each factor.

$$\mu\nabla^2 u_i = \mu \sum_j m_j \frac{u_j - u_i}{\rho_j} \nabla W(x_j - x_i) \quad (2)$$

$$\nabla p_i = \sum_j m_j \frac{p_j + p_i}{2\rho_j} \nabla W(x_j - x_i) \quad (3)$$

The variable W denotes the kernel function of the SPH method[MC11] between Prt_i and Prt_j.

By using these equations, we simulate the interactions between the wind and the trees on the basis of the particle definitions. Then all of the relationships between wind and trees are detected based on the radius of the particles. In this process, since we deal with the particles of trees as same as the particles of wind, the two-way effect are calculated by the SPH method. When we implement the SPH method, we set the natural density of the air for the parameter ρ_i and the mass of the tree m_j is calculated with $0.5g/cm^3$.

Interaction models for branches and leaves

We described the link structures of branches and leaves for simulating elastic deformations of a tree in the second section. In this section, we describe the three types of interaction forces that affect the particles for representing a tree. Figure 2 shows the link structure and the interaction forces of a branch.

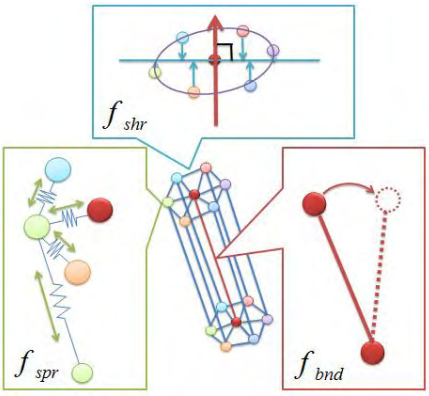


Figure 2. Three types of interaction forces for simulation of the elastic deformation of a branch

The first interaction force $f_{spr\,i}$ is the internal force of a tree based on a mass-spring system. The value of $f_{spr\,i}$ is given by equation 4.

$$f_{spr\,i} = \sum_j k_{i.j}\left(\left(x_j - x_i\right) - L_{i,j}\right) \quad (4)$$

$k_{i.j}$: Spring constant between Prt_i and Prt_j

$L_{i,j}$: Initial length between Prt_i and Prt_j

The second interaction force $f_{bnd\,i}$ is the restoring force from the central axis. Now, there are two particles for representing the central axis, Prt_i (top) and Prt_j (bottom). Then, we define the direction vector of the central axis $V_{center\,i}\left(= x_i - x_j\right)$. When we determine the restoring force, we require the relative rotation angle of the central axis to be against the initial rotation. To determine the rotation angle, we calculate the quaternion Q_j, which represents the rotation from $V_{center\,j}$ to $V_{center\,i}$ on the basis of the initial position of the tree beforehand.

$$Q_j(V_{center\,j}) = V_{center\,i} \quad (5)$$

Then, we define the interaction force $f_{bnd\,i}$ as follows:

$$f_{bnd\,i} = G_{b\,i}\left(Q_j\left(L_{i,j}|V_{center\,j}|\right) - V_{center\,i}\right) \quad (6)$$

$G_{b\,i}$: Elastic modulus

The third interaction force f_{shr} is the shear stress from the structure of a branch. This force moves Prt_i for representing branches on the circle whose center is formed by the particles of the central axis Prt_j. The normal vector of the circle is given by equation 7:

$$V_{normal\,i} = \frac{V_{center\,i}}{|V_{center\,i}|} \quad (7)$$

By using the normal vector, we can determine the distance and the direction from Prt_i to the circle. Equation 8 gives the relationship between both the distance and the direction and the interaction force f_{shr}.

$$f_{shr\,i} = G_{s\,i}\left(V_{normal\,i} \cdot x_j - V_{normal\,i} \cdot x_i\right)V_{normal\,i} \quad (8)$$

$G_{s\,i}$: Modulus of rigidity

We simulate the elastic deformations of trees by using these three types of interaction forces.

Breaking conditions of a tree

In this section, we describe the determination of the broken locations of a tree. We represent the breaking behavior of a tree by erasing the links between the particles that compose the particle-based model of the tree. We assign the limitation length of a stretch $e_{i,j}$ and a compress $c_{i,j}$ to each link of the tree model. Usually the breaking conditions are detected based on the stress of an object. In our method, the stress of a link can be calculated with $f_{spr\,i}$. Since the value of $f_{spr\,i}$ simply proportion the length between particles, we use the length to detect the broken position to be easy to set the threshold parameters by users. To determine the broken link between Prt_i and Prt_j, we use the strain measure of the link $d_{i,j}$.

$$d_{i,j} = \left|\frac{(x_j - x_i)}{L_{i,j}}\right| \quad (9)$$

If the strain rate of the link $d_{i,j}$ is greater than $e_{i,j}$ or lesser than $c_{i,j}$, the link is erased from the tree model.

$$IsBroken = \begin{cases} true & d_{i,j} \geq e_{i,j} \ \ or \ \ d_{i,j} \leq c_{i,j} \\ false & e_{i,j} > d_{i,j} > c_{i,j} \end{cases} \quad (10)$$

The threshold parameters $e_{i,j}$ and $c_{i,j}$ are given by the users. Then, if half of the links that are generated along the central axis are erased, the link of the central axis is erased.

3. GENERATION OF BROKEN BRANCH

In this section, we describe a method of modeling a broken shape of a branch. This method automatically generates a polygon model that represents a cross-sectional shape of a broken branch on the basis of the link structure of a tree described in Section 2.

Internal model of a branch

A branch is constructed by a bundle of fibers, and the cross-sectional shape of a broken branch is uneven. We define the internal model of a branch that represents the difference in the density, radius, and color of fibers in order to generate the unevenness. This internal model consists of three layers: core, wood, and bark. By controlling the thickness of each layer, this model can handle a branch that has an internal cavity, such as a bamboo.

Polygon model for broken branch

In this section, we describe the generation of a polygon model for a broken branch on the basis of the internal model of a branch (Figure 3). The generation procedure is as follows:

(a) The layered structure of a branch.

(b) The vertices that represent the fibers are placed randomly into the area of each layer according to its density.

(c) By using the Delaunay triangulation, we connect the vertices to form triangles.

(d) The triangles generated in step (c) are expanded to form triangle poles.

In our implementation, the broken shape is dynamically generated when the branch is broken. In our preliminary experiment, the computation time of the generation process is match faster than other simulations (< 0.005sec.).

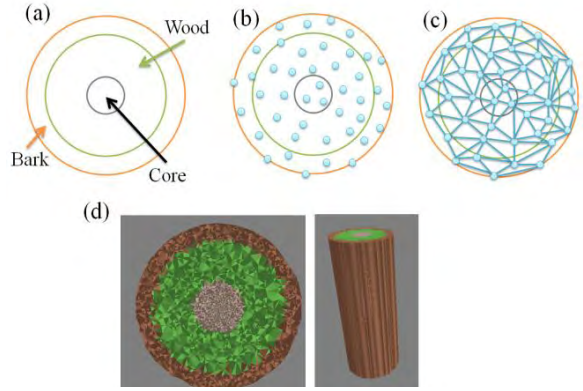


Figure 3. Generating process of a polygon model for broken branch.

Cross-section of a broken branch

In this section, we describe the division of the polygon model generated in Section 3. When a natural branch is broken at an arbitrary position on the surface of the branch, the breaking process proceeds to the other side. Then, the cross-sectional shape of the branch forms a curve that goes down to the bottom gradually. Therefore, we define the side curve of the cross-sectional shape on the basis of equation 11. This equation gives the drop rate F of the side curve (Figure 4 upper-side).

$$F(x) = L * \alpha \left(\frac{(C-x_0)\cdot(x-x_0)+1}{2r^2}\right)^2 \qquad (11)$$

L: Length of central axis, α: Brittleness,

C: Position of central axis, x_0: Starting position of the breaking.

The bottom side of the figure 4 shows the cross-sectional shape of the branch.

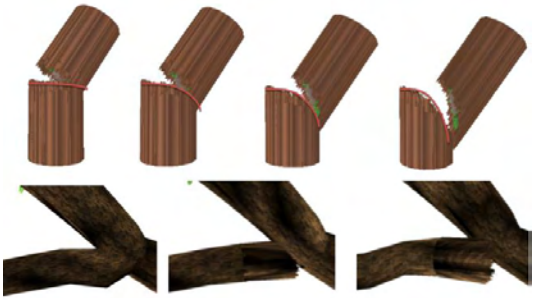


Figure 4. Cross-sectional shapes of a branch.
Upper-side: Braking goes down to the bottom.
Bottom-side: Braking behavior in a scene.

4. Results and Discussions

Particle-based tree modeling

We generate a particle- and link-based tree model by using the method described in Section 2. Figure 5 shows the polygon-based model and the particle-based model of a tree. One of the advantages of this method is that since the intervals between the particles can be changed, the users can freely change the details of the simulation of the tree.

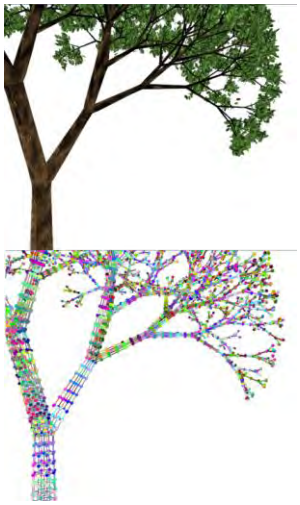


Figure 5. Result of generating particle-based model from a 3D model of a tree.

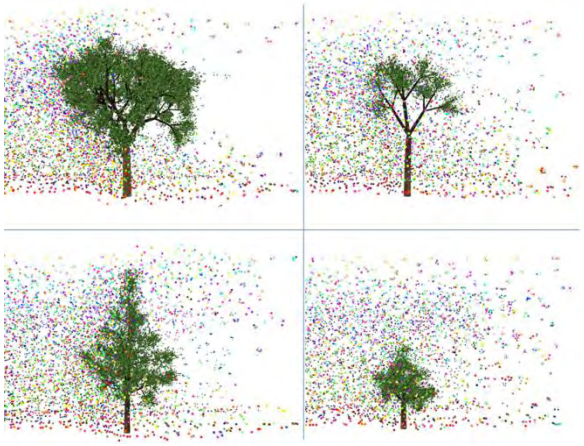


Figure 6. Comparison of the interaction between a tree and wind flow according to the differences in the shapes of trees.

Interactions between wind and tree

In this section, we present the results of the tree animations that contain the breaking behaviors of branches and falling leaves.

4.1.1 Swaying branches and leaves

Figure 6 shows the comparison of the wind flow according to the differences in the shapes of trees. The color dots on the images are the particles used for representing the wind. Since the branches and leaves of the trees block the wind particles, the larger and denser tree the more wind flows around the tree. These results show that the proposed method can represent the natural interactions between the tree and the wind.

4.1.2 Breaking branches and falling leaves

Figure 7 shows the Comparison of the broken conditions of branches according to the differences in the velocity of the wind. The red circles are the signs of the broken branches. In the previous study, Saleem et.al. [SCZ+07] enable to visualize the flying effect of the branches. However, this method calculates the broken conditions by using the angle of the branch and velocity of wind. Therefore the Salem's method could not visualize the difference of the broken position of a tree according with the velocity of wind. In our result, the image on the top of figure 7 is the result of the lowest velocity of the wind. This results show that there are any broken branches and a few fallen leaves. The faster the velocity of the wind causes the more number of the broken branches and fallen leaves. This result shows that the proposed method can visualize the difference of the broken position according with the velocity of the wind and the structure of the tree.

4.1.3 Implementation and performance

In this section we discuss the simulation performance of our method. We measured the processing time on a 2.66Ghz Xeon PC with 8GB RAM. We set the radius of the particles for wind 0.05m and place the particles for the trees every 0.1m. The time-step of our simulation is 0.01sec. Table3 shows the results to compare the time scaling according to the number of the particles for trees. This result shows that the increase of the number of the particles causes the increase of both the time for simulating wind and the tree. Since our method calculates the relationship between a tree and wind in the process of SPH method, the processing time for SPH method also increase according to the particles of a tree. The table 4 shows the results to compare the time scaling according to the number of the particles for wind. This result shows that since the relationship between the tree and wind are considered in the process of SPH method, the processing time for computing the tree deformation is not changed according to the number of the particle of wind. Since the processing times are not real-time,

Number of particles		Calculation time per fame (sec.)		
Wind	Tree	SPH	Tree Deformation	Total
7680	16877	0.225	0.134	0.360
	22319	0.356	0.178	0.535
	31812	0.565	0.264	0.829
	44222	0.958	0.367	1.325
	72759	2.412	0.617	3.028
	102282	4.350	0.869	5.219

Table 3. The time scaling according to the number of the particles for trees.

Number of particles		Calculation time per fame (sec.)		
Wind	Tree	SPH	Tree Deformation	Total
2560	21900	0.262	0.177	0.439
5120		0.310	0.174	0.484
9984		0.395	0.174	0.569
17664		0.610	0.176	0.787
25088		1.096	0.175	1.270
35072		2.312	0.181	2.492

Table 4. The time scaling according to the number of the particles for wind.

5. CONCLUSIONS

We proposed the particle-based simulation method to create the animations of swaying and breaking trees. We obtained the following results:

(1) We defined the four types of particles (wind, branch, leaf and bone) and the four types of interaction forces (wind, internal force, restoring

force and shear stress) between the particles. By using the models, we can simulate the swaying and breaking behaviors of trees.

(2) We proposed the generation method of a polygon model for a broken branch. This method is able to generate the cross-sectional shape of a broken branch on the basis of the link structure of a tree.

(3) Our results show that the animation of swaying and breaking trees is automatically generated by the proposed method. By using the particle-based simulation of the wind, the proposed method can represent that the wind flow naturally around the tree.

In a future work, we will accelerate our simulation method to be able to generate animations in real-time by using parallel processing on GPU.

6. ACKNOWLEDGMENTS

This research was supported by Adaptable and Seamless Technology Transfer Program through Target-driven R&D (A-Step), Japan Science and Technology Agency.

7. REFERENCES

[AT11]Ando, R., Tsuruno, R., A particle-based method for preserving fluid sheets, In Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '11), pp.7-16, 2011.

[AWZ09]Ao, X., Wu, Y., Zhou, M., Real time animation of trees based on BBSC in computer games, International Journal of Computer Games Technology, Article 5, 2009.

[BCN+10]Baxter, R., Crumley, Z., Neeser, R., Gain, J., Automatic addition of physics components to procedural content, In Proceedings of the 7th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa (AFRIGRAPH '10), pp.101-110, 2010.

[CBP05]Clavet, S., Beaudoin, P., Poulin, P., Particle-based viscoelastic fluid simulation, In Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '05), pp.219-228, 2005.

[Chn11]Chng. E., Realistic Placement of Plants for Virtual Environments. IEEE Comput. Graph. Appl. 31, 4, pp.66-77, 2011.

[DRB+09]Diener, J., Rodriguez, M., Baboud, L., Reveret, L., Wind projection basis for real-time animation of trees, Computer Graphics Forum (Proceedings EUROGRAPHICS 2009), 28, 2, pp. 533-540, 2009.

[GSS+10]Goswami, P., Schlegel, P., Solenthaler, B., Pajarola, R., Interactive SPH simulation and rendering on the GPU, In Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '10), pp.55-64, 2010.

[GBB09]Gerszewski, D., Bhattacharya, H., Bargteil, W.A., A point-based method for animating elastoplastic solids, In Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '09), pp.133-138, 2009.

[HKW09]Habel, R., Kusternig, A., Wimmer, M., Physically Guided Animation of Trees, Computer Graphics Forum (Proceedings EUROGRAPHICS 2009), 28, 2, pp.523-532, 2009.

[Luc77]Lucy, L., A numerical approach to the testing of the fission hypothesis, 82, pp.1013-1024, 1977.

[MC11]Muller, M., Chentanez, N., Solid simulation with oriented particles, ACM Trans. Graph., 30, 4, Article 92, 2011.

[PKA+05]Pauly, M., Keiser, R., Adams, B., Dutr, P., Gross, M., Guibas, J.L., Meshless animation of fracturing solids, ACM Trans. Graph. 24, 3, pp.957-964, 2005.

[SBB+08]Stava, O., Benes, B., Brisbin, M., Krivánek, J., Interactive Terrain Modeling Using Hydraulic Erosion. In Proceedings of Symposium on Computer Animation. pp.201-210, 2008.

[SCZ+07]Saleem, K., Chen, S., Zhang, K., Animating Tree Branch Breaking and Flying Effects for a 3D Interactive Visualization System for Hurricanes and Storm Surge Flooding, In Proceedings of the Ninth IEEE International Symposium on Multimedia Workshops (ISMW '07), pp.335-341, 2007.

[TMW02]Tobler, F. R., Maierhofer, S., Wilkie, A., A multiresolution mesh generation approach for procedural definition of complex geometry. In Shape Modeling International 2002, 11, 15, pp.35-43, 2002.

[WRK+10]Wicke, M., Ritchie, D., Klingner,M. B., Burke, S., Shewchuk, R. J., O'Brien, F. J, Dynamic Local Remeshing for Elastoplastic Simulation, In Proceedings of ACM SIGGRAPH 2010, pp. 49:1–11, 2010.

[YHY+10]Yang, M., Huang, M., Yang, G., Wu, E., Physically-based animation for realistic interactions between tree branches and raindrops, In Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology (VRST '10), pp.83-86, 2010.

Figure 7. Comparison of the broken conditions of branches according to the differences in the velocity of the wind.

A Statistical Framework for Estimation of Cell Migration Velocity

Guangming Huang Jiwoong Kim Xinyu Huang Gaolin Zheng Alade Tokuta
Department of Mathematics and Computer Science
North Carolina Central University
1801 Fayetteville Street
Durham, NC 27707, USA
{ghuang, jkim7, huangx, gzheng, atokuta}@nccu.edu

ABSTRACT

Migration velocity of cell populations *in vitro* is one of important measurements of cell behaviors. As there are massive amount of cells in one image that share similar characteristics and are highly deformable, it is often computational expensive to track every individual cell. It is also difficult to track cells over a long period of time due to propagation of segmentation and tracking errors. This paper presents an algorithm to estimate migration velocity of cell populations observed by time-lapse microscopy. Instead of tracking cells individually, our proposed algorithm computes mutual information between image blocks of consecutive frames. The migration velocity is then estimated by a linear regression, with mutual information and foreground area ratio as input. Experiments on a variety of image sequences verified that our algorithm can give accurate and robust estimation under different situations in real-time.

Keywords

Cell Migration, Mutual Information, Linear Regression.

1. INTRODUCTION

It is important to measure cell migration velocity in many biomedical applications, such as wound healing assay of cell monolayers [YPW+04] and analysis of red blood cell in microcirculation [WZH+09]. For cell populations, there are mainly two obstacles to estimate the migration velocity accurately and robustly. First, all the cells in a population have very similar characteristic, such as shape and intensity. Second, cells are often highly deformable. For example, two cells could merge into one cell, and one cell could divide to two or more cells. As a result, it could be difficult and computational expensive to track every cell in image sequences. Figure 1 shows three examples of cell populations. Cells in some types of images even have very similar intensities to the background as shown in figure 1(b). Thus, the segmentation algorithms based on intensity values would easily fail.

In this paper, we present an efficient and novel algorithm to estimate cell migration velocity. Our algorithm first computes mutual information and foreground ratio between image blocks of two consecutive frames. A linear regressor is then trained and applied to estimate migration velocity.

Mutual information has been widely used to align two images in many medical applications [PMV03] in order to reduce the error during the image acquisition (*e.g.,* finger jigging). However, to our knowledge, there is no work that uses mutual information as an input variable of regression for the estimation of cell migration velocity. As there are no individual cells involved in the estimation process, our algorithm contains no accumulated segmentation and tracking errors.

Therefore, the proposed algorithm has several advantages. First, accurate segmentation and data association of cell contours are not required. Second, it can be performed in real-time without using motion trackers for all the cells. Third, since tracking accuracy is not an issue (*e.g.,* there are no accumulation errors), it can be used to estimate migration velocity for a long period.

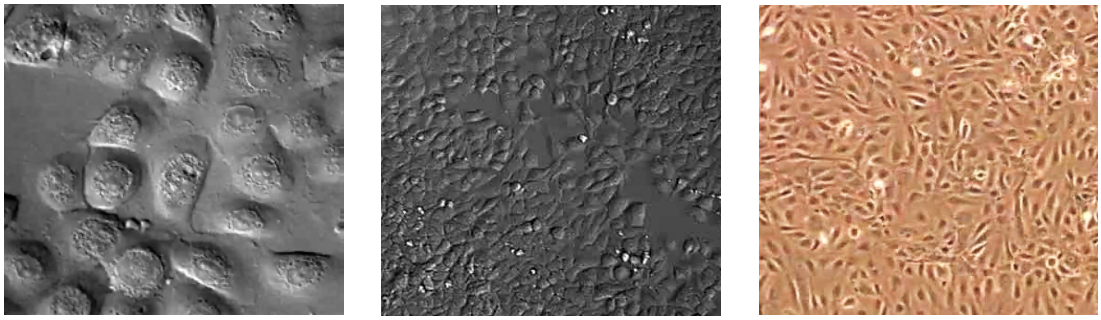


Figure 1. Examples of microcopy images of cell populations: (*left*) primary keratinocytes [Cel09]. (*middle*) cancer cells captured spinning disc confocal microscope [Mar09]. (*right*) human umbilical vein endothelial cells (HUVEC) [Yam09].

The remainder of this paper is organized as follows. Section 2 describes related work. Our proposed algorithm is given in section 3. Section 4 shows the experiments using three different datasets. The conclusion is given in section 5.

2. RELATED WORK

Wu *et al.* proposed an approach to measure velocity of red blood cell from capillary video using the optical flow technique [WZH+09]. In order to apply the optical flow technique, the skeleton of vessel needs to be extracted first based on a set of pre-processing steps, such as connected component labeling, thinning, and length pruning. These pre-processing processes may not be applied to other types of cells in general due to occlusions, deformations, and even varying illuminations. More importantly, the velocity determination in this approach is based on two assumptions: a) intensity of each cell does not change over time; b) the surrounding area of the cell move in a similar manner. These two assumptions are fundamental to apply the optical flow on the skeletons. They however, are too restricted and cannot be extended in general. Other approaches [DSA+08, LGM04] that based on detection and graph extraction of vessel shapes also have similar problems.

In [LMC+08], Li *et al.* proposed an automated tracking algorithm to track hundreds to thousands of cells and construct lineages simultaneously. This tracking system first segments candidate cell regions and tracks them over frames by forming a minimization problem with a topological constraint. Then this system predicts and filters the cell motion dynamics using interacting multiple model filters, and construct lineages by checking the entire tracking history. The proposed system can be used to analyze a number of cell behaviors including migration, division, death, and so on.

According to [MDI+09], tracking in cell can be divided into two stages, segmenting individual cells and connecting cells over time. However, since each possible candidate cell needs to be considered, the whole process could be computational expensive. For the purpose of estimating migration velocity, the algorithms based on tracking individual cells could be complicated and hence may not be the best choice. Moreover, common used segmentation algorithms based on intensity values could easily fail to distinguish between background areas and candidate cells.

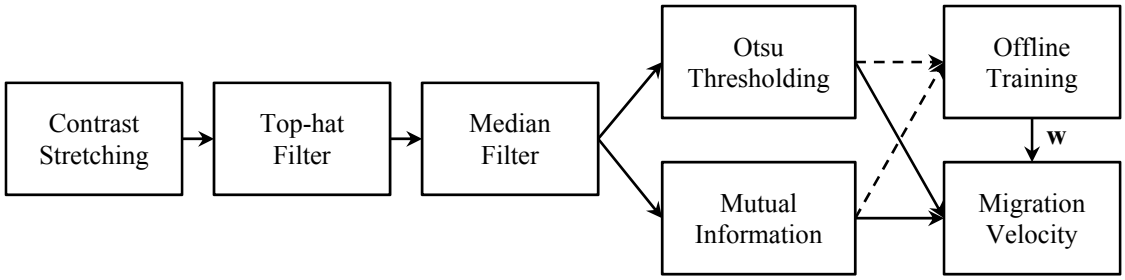


Figure 2. Overview of our methodology

3. METHODOLOGY

Our algorithm can be divided into three modules, 1) image enhancement and foreground detection; 2) computation of mutual information between image blocks; 3) velocity estimation using linear regression. Figure 2 gives an overview of the algorithm.

3.1 Image Enhancement and Foreground Detection

The purpose of image enhancement is to reduce noise and enhance the image contrast. For simplicity, we assume two image frames has been aligned. This can be achieved by an affine transformation based on an estimated homography [HZ04], or a non-linear transformation using mutual information [PMV03].

First, we enhance the image by a linear contrast stretch that enlarges the range of intensity values to the entire available range. If the cells appear darker than the background, the images are inverted so that the cell interior areas are brighter. Next, the enhanced image is convolved with a 7×7 top-hat filter, which is often used to detect bright features on a dark background. A 3×3 median filter is then applied twice to remove small grey objects that could be noise or artifacts caused by the top-hat filter. The output image after the median filter is the input image for the computation of mutual information. We further detect the foreground by using Otsu thresholding [Otu79] to obtain a binary image where the white pixels indicate the foreground. Figure 3 visualizes these image processing steps.

Unlike many algorithms that require fairly accurate foreground and cell detections, we only need to detect a rough foreground as shown in Figure 3(f). Moreover, there are no accumulated segmentation errors.

3.2 Computation of Mutual Information

Mutual information is usually used to test the independence between two random variables **x** and **y**. If two random variables are independent, the joint probability $p(\mathbf{x}, \mathbf{y})$ can be factorized into the product of their marginals $p(\mathbf{x})p(\mathbf{y})$. In our application, the random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are two same image blocks from two consecutive image frames.

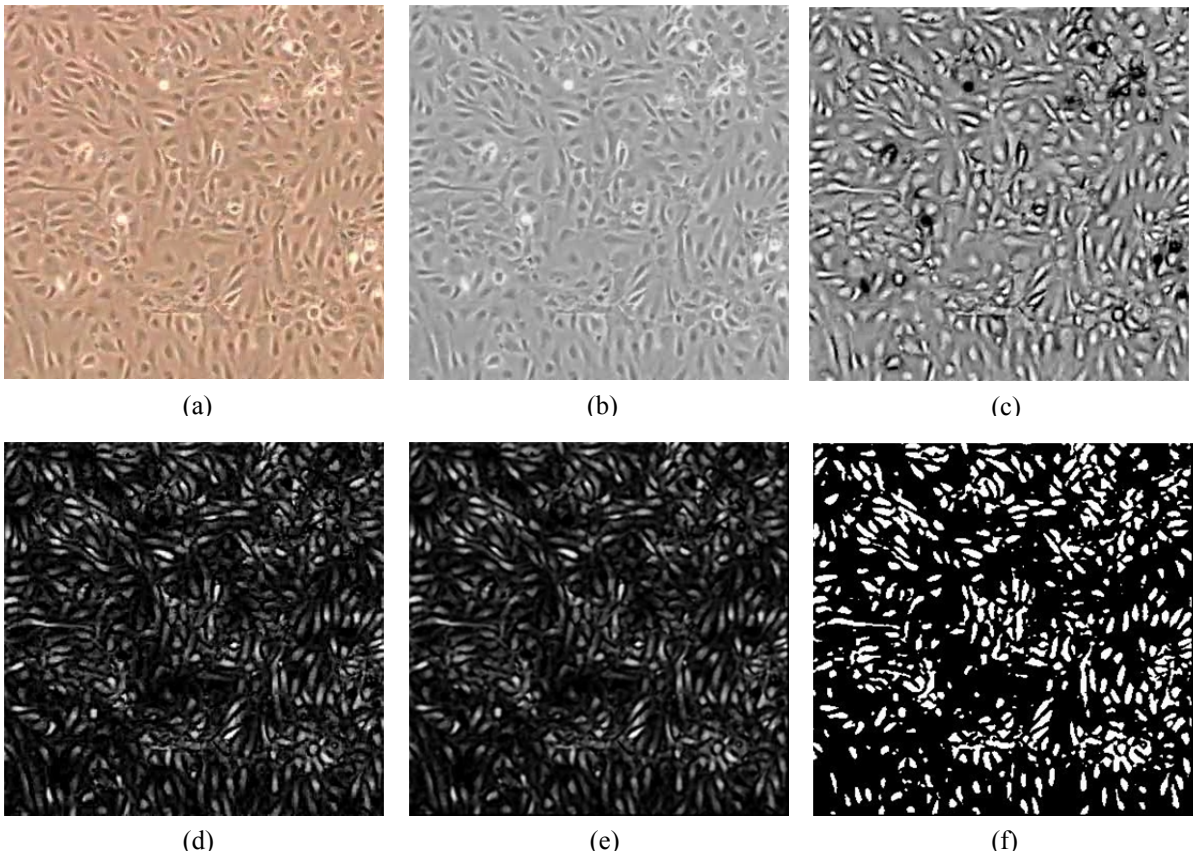


Figure 3. Image enhancement and foreground detection. (a) Original color image. (b) Grayscale image. (c) Result after intensity inversion and contrast stretching. (d) Result of top-hat filtering. (e) Output of median filtering. (f) Foreground detection by Otsu thresholding.

We do not use the entire image frames as the random variables. This is mainly because that the majority of an image could be the background, in which case, when the entire images are used, the mutual information between them could mainly reflect the differences between backgrounds. Furthermore, when the captured image has a high resolution, it also could be inefficient since the memory storage is increased. Therefore, we divide each frame into a set of image blocks with overlapping areas. The size of the image block is determined by the maximum of cell migration velocity, which can be easily estimated by the visual inspection. For many types of cells, the size of an image block is often around a few times that of a single cell.

Let us denote an image block as **x** and the same image block in the next image frame as **y**. The mutual information between the variables **x** and **y** is defined as

$$I(\mathbf{x},\mathbf{y}) = \mathrm{KL}(p(\boldsymbol{x},\boldsymbol{y})||p(\mathbf{x})p(\mathbf{y})) = \sum_{i=1}^{N}\sum_{j=1}^{N} p(x_i,y_j)\ln\left(\frac{p(x_i,y_j)}{p(x_i)p(y_j)}\right)$$

where KL(·) is known as the Kullback-Leibler divergence, and $p(\mathbf{x})$ and $p(\mathbf{y})$ are histogram distributions of two image blocks. We can see that $I(\mathbf{x},\mathbf{y}) = 0$ if and only if two image blocks are independent, which indicates a very large migration. The larger the $I(\mathbf{x},\mathbf{y})$, the more similar two image blocks are, which indicate a small migration.

3.3 Velocity Estimation using Linear Regression

For each pair of image blocks **x** and **y**, we describe the cell migration using two features: the mutual information *I* and the difference between foregrounds of **x** and **y**. The mutual information *I* measures independence between two image blocks including both foreground and background. The difference between two foregrounds is measured by the ratio of non-overlapping foreground to the union of the two foregrounds, which is defined by

$$D = \frac{|\mathbf{x}_F \cup \mathbf{y}_F - \mathbf{x}_F \cap \mathbf{y}_F|}{|\mathbf{x}_F \cup \mathbf{y}_F|}$$

where subscript *F* indicates the foreground. In general, the smaller the ratio, the more similar two image blocks are. Thus, the inputs for the regression are the mutual information *I* and area ratio *D*, and the output is the cell migration velocity *v*. Since there could exist multiple moving cells in one image block, the maximum velocity is chosen as output.

In order to avoid over-fitting problem, we only consider the second order of the inputs. Therefore, the possible variables include I, D, I^2, D^2, and ID. We adopt the *forward selection* to select a suitable model for the data. In this model selection approach, we add one variable that results in the largest reduction in sum squared errors (SSE), and then carry out a hypothesis test to determine whether this reduction in SSE is significant. If the reduction is significant, we continue the adding process and stop otherwise. The results show that the full model is the most suitable regression model when the area ratio can be robustly estimated. The regression model is given by

$$v(I,D,\mathbf{w}) = [1 \;\; I \;\; D \;\; I^2 \;\; D^2 \;\; ID]^T\mathbf{w}$$

where $\mathbf{w} = (w_0,\dots,w_5)^T$.

In order to estimate the parameters **w**, we first developed an interactive user interface to measure the velocities of a set of sample cells ($N \approx 30$) by manually clicking centroids of the same cells in two image frames. Then the parameters **w** can be estimated by the normal equations

$$\mathbf{w} = (\boldsymbol{\phi}^T\boldsymbol{\phi})^{-1}\boldsymbol{\phi}^T\boldsymbol{v}$$

where $\boldsymbol{\phi}$ is $N \times 6$ design matrix given by

$$\boldsymbol{\phi} = \begin{pmatrix} 1 & I_1 & D_1 & I_1^2 & D_1^2 & I_1D_1 \\ 1 & I_2 & D_2 & I_2^2 & D_2^2 & I_2D_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & I_N & D_N & I_N^2 & D_N^2 & I_ND_N \end{pmatrix}$$

It could be very difficult to estimate the area ratio *D* for some types of microcopy images. For example, for the image type shown in figure 1(b), the intensities of both foreground and background are very similar and illumination conditions also change when cells are moving. Therefore, the most exiting cell segmentation algorithms based on intensity values would fail to detect foreground and moving cells accurately.

For these types of images, we discard the area ratio *D* in our regression model and only use the mutual information. Thus, the model could be changed to a polynomial regression with order 3, which is given by

$$v(I,\mathbf{w}) = [1 \;\; I \;\; I^2 \;\; I^3]^T\mathbf{w}$$

where $\mathbf{w} = (w_0,\dots,w_3)^T$. The normal equations remain same and the $N \times 4$ design matrix is defined by

$$\boldsymbol{\phi} = \begin{pmatrix} 1 & I_1 & I_1^2 & I_1^3 \\ 1 & I_2 & I_2^2 & I_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & I_N & I_N^2 & I_N^3 \end{pmatrix}$$

During the prediction stage, we divide each image frame into $n \times m$ blocks and estimate the migration velocity for each block using the linear regression. We can then plot the velocity distribution over time. The expectation of the migration velocity can also be estimated for each frame by

$$E(v) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} v_{ij}$$

where v_{ij} is the velocity in each block.

4. EXPERIEMNTS

4.1 Datasets

Our proposed algorithm is tested on three microcopy image sequences. A few frames of the same cell type are used for estimation of parameters **w**.

Dataset A has one sequence of human umbilical vein endothelial cells (HUVEC) that are isolated from normal umbilical vein [Cel09]. The images are cropped to 474 × 364.

Dataset B includes image sequences that show cell migration of primary keratinocytes before and after calcium switch [Yam09]. These images have a dimension cropped to 642 × 449 pixels.

Dataset C contains one image sequence taken overnight by using a spinning disc confocal microscope [Mar09]. It shows motility of cancer cells. This image sequence has 111 frames with 472 × 360 pixels/frame.

4.2 Estimation of Parameters

The parameters **w** for the linear regression are learned from the training data. For each dataset, the parameters are learned using a set of samples marked by our interactive graphic interface. The image block size is set to 64 × 64, which is around 2 times larger than a typical cell length The training sets include 20~40 samples from each dataset. Figure 4-7 show the scatterplot matrix of the 34 training samples from dataset A, and 25 training samples from dataset C. It is easy to see the strong correlation among mutual information, area ratio, and migration. The training samples from the dataset B have the distribution similar to the figure 4 and 5 from the dataset A.

The area ratio distribution in figure 7 is more dispersed than the distribution shown in figure 5. This is mainly because that the cell intensity is very close to the background intensity as shown figure 1(b). The segmentation based on intensity tends to fail. In this case, the regression with only mutual information can provide a more stable result. Table 1 gives a summary of parameters for dataset A and C.

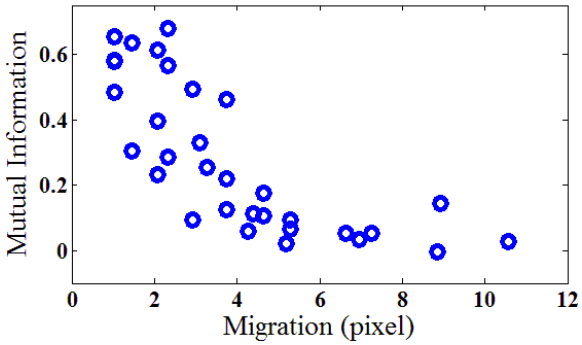


Figure 4. Mutual information *I* versus migration using 34 training samples from dataset A.

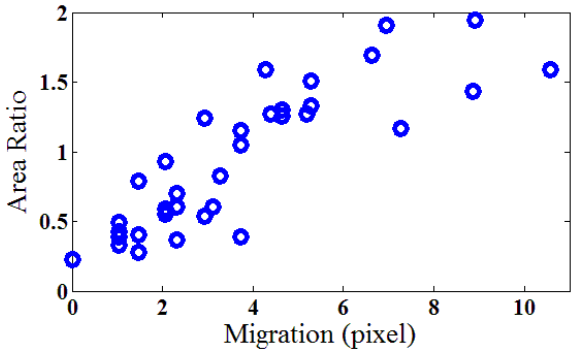


Figure 5. Area ratio *D* versus migration using 34 training samples from dataset A.

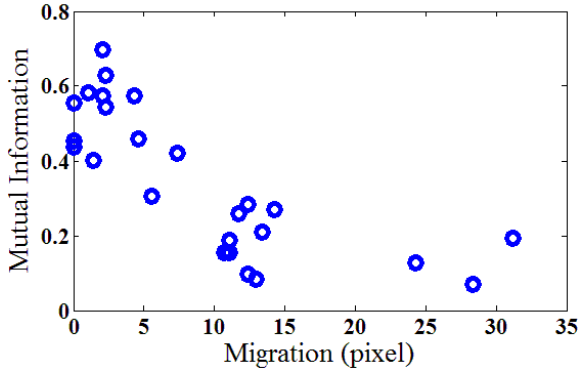


Figure 6. Mutual information *I* versus migration using 25 training samples from dataset C.

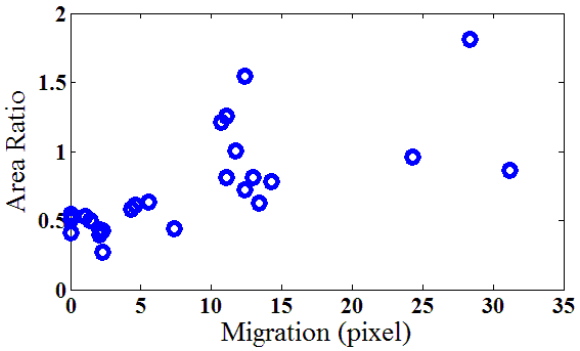


Figure 7. Area ratio D versus migration using 25 training samples from dataset C.

Table 1. Summary of parameters for dataset A and database C.

DB	w_0	w_1	w_2	w_3	w_4	w_5	Block Size
A	0.202	0.246	0.039	0.349	0.016	0.031	64×64
C	0.114	0.041	0.019	0.010	N/A	N/A	64×64

4.3 Estimation of Migration Velocity

After the training stages, we compute the migration velocity over the whole range of each dataset.

Figure 8 shows the result from the dataset A. We computed migration velocity using our algorithm for every 50 frames over 4271 image frames captured over 70 hours and 37 minutes. This velocity distribution with the bell shape is same as our expectation. At the beginning of the captured image sequences, the number of cells is relatively small and the cell growth rate is high. After the growth rate reaches its peak, the cells are very crowded and touch each other in the limited space. As a result, the growth rate decreases. Figure 9 shows two image frames at the beginning and at the end of the dataset A.

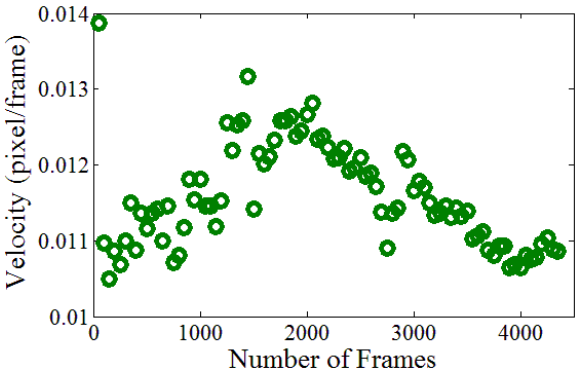
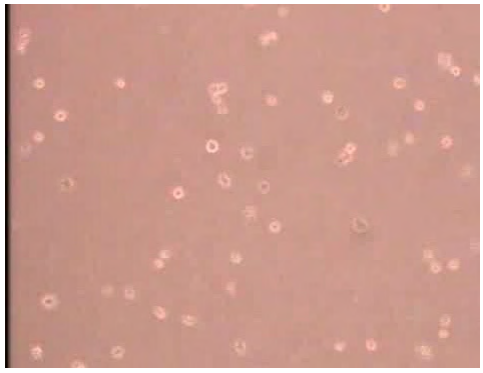
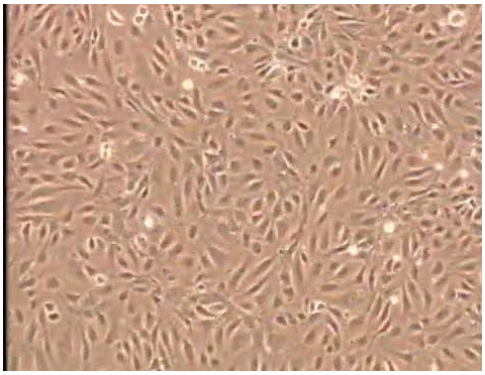


Figure 8. Velocity distribution for the dataset A.



(a) 92th Frame at 01m00s



(b) 4,250th Frame at 68h46m30s

Figure 9. Two image frames from the dataset A that could be used to further verify the cell growth rate.

For the dataset B, as the concentration of calcium was increased from low to high, cell migration velocity is prevented by the maturation of cell-cell adhesion after the calcium switch. Therefore, the velocity decreases shown in figure 10 is also same as our expectation. Here we take every 3 frames to compute velocity over 73 frames that are captured using 6 hours.

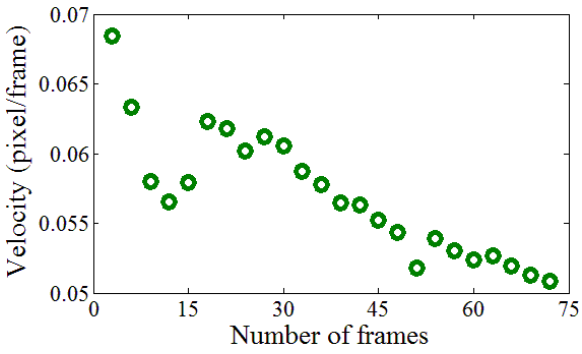


Figure 10. Velocity distribution for the dataset B.

Figure 11 shows the velocity distribution for the dataset C over 111 frames using more than 18 hours.

We sample every 10 frame. As the population of cancer cells gradually reaches its peak in the limited space, the velocity also decreases gradually. We further plot the mutual information distribution for every frame in order to verify our results. Figure 12 shows the distributions and the corresponding image frames. It is clear to see the distributions of mutual information are very similar to the distribution of the migration velocity.

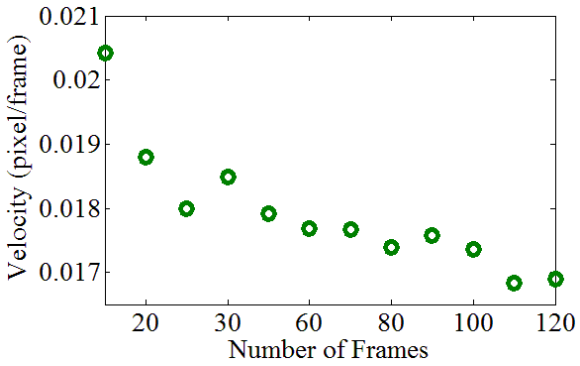
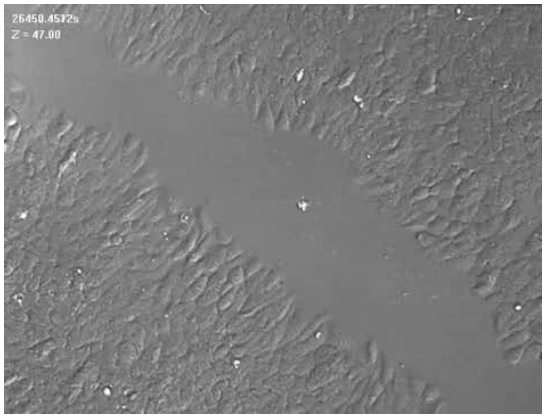
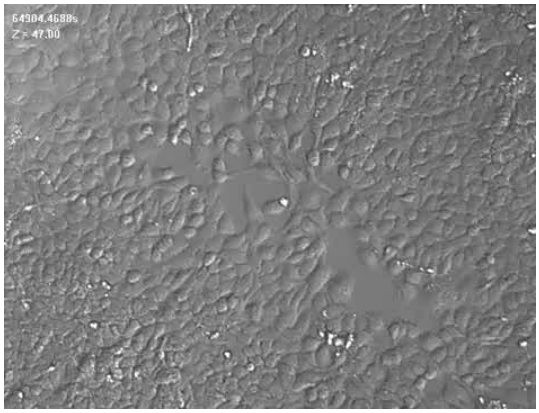


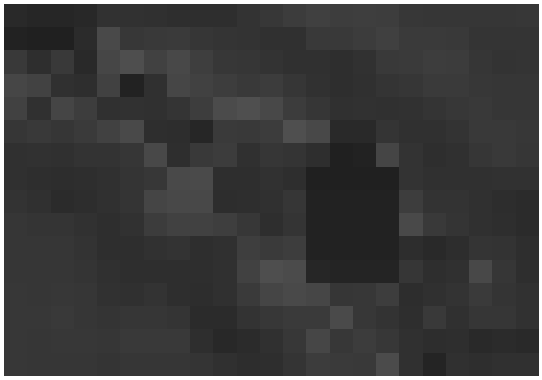
Figure 11. Velocity distribution for the dataset C.



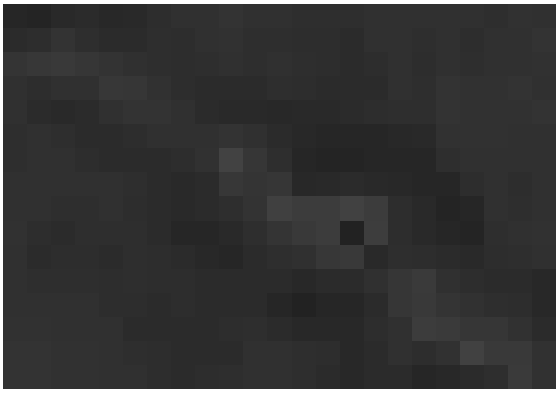
(a)



(b)



(c)



(d)

Figure 12. Images and distributions of mutual information. (a) 45th frame of dataset C. (b) 109th frame of dataset C. (c) distribution of mutual information corresponding to (a). (d) distribution of mutual information corresponding to (b). (c) and (d) are enhanced for the purpose of visualization. The brighter a region, the larger the mutual information is.

5. CONCLUSION AND FUTURE WORK

In this paper, we propose a novel algorithm to estimate cell migration velocity. As individual cell segmentation and tacking are avoided, this algorithm is efficient and robust. Our experiments show that this algorithm is also accurate and can be used to measure cell motility over a long time. In the future, we would like to extend our work to estimation of cell division, merging, and growth based on our regression framework.

6. ACKNOWLEDGMENTS

The authors acknowledge support of the National Science Foundation HRD 0833184.

7. REFERENCES

[Cel09] Cell Applications, Inc., Human Endothelial Cells, http://www.cellapplications.com/endothelial-cells.

[DSA+08] J. G. G. Dobbe, G. J. Streekstra, B. Atasever, R. van Zijderveld, and C. Ince, Measurement of functional microcirculatory geometry and velocity distributions using automated image analysis. Journal of Medical and Biological Engineering and Computing, Volume 46, Number 7, Pages 659-670, 2008.

[HZ04] Hartley, R. and Zisserman, A., Multiple View Geometry in Computer Vision, 2nd Edition, Cambridge University Press, ISBN: 0521540518, 2004

[LGM04] Lamberti, F., Gamba, A., and Montrucchio, B, Computer-assisted analysis of in-vitro vasculogenesis and angiogenesis processes. Journal of WSCG, Volume 12, Number 1-3, Pages 237-244, 2004.

[LMC+08] Kang Li, Eric D. Miller, Mei Chen, Takeo Kanade, Lee E. Weiss, and Phil G. Campbell, Cell population tracking and lineage construction with spatiotemporal context. Journal of Medical Image Analysis, Volume 12, Issue 5, Pages 546-566, 2008.

[Mar09] Charles Marcus, Cancer cell migration and movement, http://marcuslab.harvard.edu/index.shtml, Department of Physics, Harvard University.

[MDI+09] E. Meijering, O. Dzyubachyk, I. Smal, and W.A. van Cappellen, Tracking in Cell and Developmental Biology. Seminars in Cell and Developmental Biology, Volume 20, Issue 8, Pages 894-902, 2009.

[Otu79] Otsu, N., A threshold selection method from gray level histograms. IEEE Transactions on Systems, Man, and Cybernetics - Part B, Volume 9, pages 62–66, 1979.

[PMV03] Pluim, J.P.W., Maintz, J.B.A., and Viergever, M.A., Mutual-information-based registration of medical images: a survey, IEEE Transactions on Medical Imaging, Volume 22, Issue 8, Pages 986-1004, 2003.

[WZH+09] Chih-Chieh Wu, Geoffrey Zhang, Tzung-Chi Huang, and Kang-Ping Lin, Red blood cell velocity measurements of complete capillary in finger nail-fold using optical flow estimation. Journal of Microvascular Research, Volume 78, Issue 3, Pages 319-324, 2009.

[Yam09] Soichiro Yamada, Keratinocyte migration, http://yamadalab.ucdavis.edu/, Biomedical Engineering, University of California at Davis.

[YPW+04] Justin C Yarrow, Zachary E Perlman, Nicholas J Westwood, and Timothy J Mitchison, A high-throughput cell migration assay using scratch wound healing, a comparison of image-based readout methods, BMC Biotechnology, Volume 4, Number 1, 21, 2004

Caustic Object Construction Based on Multiple Caustic Patterns

Budianto Tandianus
Nanyang Technological University, Singapore
budi0010@ntu.edu.sg

Henry Johan
Nanyang Technological University, Singapore
henryjohan@ntu.edu.sg

Hock Soon Seah
Nanyang Technological University, Singapore
ashsseah@ntu.edu.sg

ABSTRACT

Inverse caustic problem, that is computing the geometry of a reflector and/or refractor based on a given caustic pattern, is currently not widely studied. In this paper, we propose a technique to solve the inverse caustic problem in which we compute the geometry of a semi-transparent homogeneous refractive object (caustic object) given a directional light source and a set of caustic patterns (each pattern is considered to be formed at a specified distance from the caustic object). We validate the results by using mental ray (software rendering). The novelty of our research is that we consider a set of caustic patterns whereas existing techniques only consider one caustic pattern. We employ a stochastic approach to simulate the refracted light beam paths that can approximately reconstruct the input caustic patterns. Working backward, from the computed refracted light beam paths we compute the geometry of the caustic object that can produce such light beam paths. Due to having multiple caustic patterns as the inputs, it is a challenge to reconstruct the input caustic patterns because of the differences in their shapes and intensities. We solve this problem by using a two-step optimization algorithm in which we adjust the position and size of the caustic regions in the first step and we adjust the caustic shapes in the second step. Our technique is able to construct a caustic object for a various types of input caustic patterns.

Keywords: caustics, photon, reconstruction, inverse problem, stochastics

1. INTRODUCTION

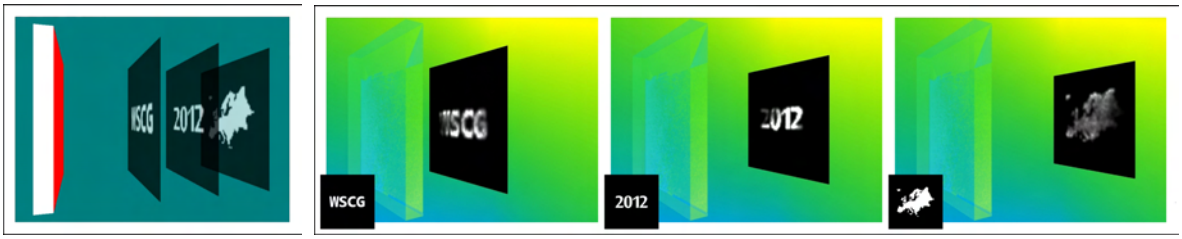
Recently, there is a growing interest in inverse problem research in Computer Graphics due to the possibility of controlling the creation of visual effects. By using the inverse techniques, the design process becomes easier as the artists can just specify the intended effects directly instead of performing the iterative trial-and-error process. However, inverse problem is generally difficult as in most cases there is no unique bijective relationship between the output and the input (i.e., given an output, there are many input possibilities that can generate such output).

In the inverse caustic problem, given an input caustic pattern (shape, intensity, and location from the caustic object) and a light source, we have to compute the geometry of the caustic object that can produce a caustic pattern similar to the input caustic pattern. Inverse caustic problem is hard to solve because the input caustic pattern only contains the irradiance magnitude and it does not have incident light direction information (and also the reflected and/or refracted light paths). Up to now, the inverse caustic problem is not widely studied and the existing work only consider a single input caustic pattern.

In this paper, we propose a new inverse caustic problem, that is computing the caustic object given multiple input caustic patterns formed on a caustic receiver (diffuse and non-transparent surface) at various distances from the caustic object. We show an example in Figure 1.

Our basic idea for solving this problem is as follows. We subdivide one side of the caustic object and also the caustic patterns into regular cells. The light beam refracted by each caustic object cell will pass through one caustic cell of each caustic pattern. We try to compute the orientation of each caustic object cell such that the combination of the refracted light beams of all caustic object cells can approximately reconstruct the input caustic patterns.

We use a stochastic approach in our technique and we represent each input caustic pattern as a 2D probability mass function (pmf) by considering the brightness of a caustic cell as the probability of a light beam might pass through it (i.e., the brighter the caustic cell is, the higher probability or the more likely a light beam is considered to pass through it). Hence, for each cell of the caustic object, we use the pmfs of the caustic patterns to determine to which direction the caustic object cell refracts a light beam. From the determined refracted light beam



(a) Scene setup. (b) Computed caustic object and approximately reconstructed caustic patterns.

Figure 1: (a) Scene setup. We compute a caustic object (the leftmost box), specifically the surface geometry of the side (shown in red color) facing the caustic patterns, given three caustic patterns (WSCG, 2012, and Europe) to be formed on a caustic receiver at three distances from the caustic object, with a directional light source orthogonal to the caustic object illuminates from the left. (b) mental ray renderings of caustics produced by our computed caustic object (final output). Input caustic patterns are shown in the insets at each image. The computational time is 9.0 hours.

direction, we can compute the orientation of the caustic object cell.

Due to differences of the input caustic patterns (in terms of shapes and intensities), it is hard to compute the caustic object that can satisfy all the input caustic patterns. Thus, we relax the input requirements by slightly adjusting the sizes, positions, and shapes of the non-zero intensity regions of the input caustic patterns. Moreover, we also allow a small amount of light beam which has passed through several caustic patterns to miss or overshoot the rest of the caustic patterns. We compute these adjustments by using optimization techniques. We validate our results by performing rendering simulation using mental ray [men12a], a robust industry standard rendering engine.

2. RELATED WORK

Unknown Input Given only the output, the input that can produce such output is computed. This is a hard problem since the *a priori* knowledge of the input is not available. One example is the work presented by Bottino et al. [Bot01a] and Mitra et al. [Mit09a]. They compute a 3D geometry that can satisfy the inputs which consist of a set of silhouettes or shadow patterns.

Inverse Caustics One of the earliest work in inverse caustic is presented by Patow and Pueyo [Pat04a]. They compute the reflector shape in an optical set (consists of a reflector, light source, and diffusor) given the radiance distribution as an input. They represent the reflector as grids and they iteratively adjust the grid vertices such as positions and number of vertices based on the similarity with the intended radiance distributions. The whole process took many days even though they can obtain radiance distribution similar to the input. They improve the work by allowing the user to set the range of the solution space [Pat07a] (the lower bound and the upper bound of the reflector shape). Hence, a user has more control in determining the reflector shape. They later increase the performance by using GPU [Mas09a] and they can reduce the processing time into magnitude of hours.

In parallel with the aforementioned work, Anson et al. [Ans08a] represent the reflector as a NURBS surface and Finckh et al. represent the reflector as a B-Spline surface [Fin10a]. As a result, during the optimization they optimize the control points instead of grid vertices which in the end can produce smooth reflectors in a relatively fast speed (due to the small number of parameters to be optimized). However, as Papas et al. also mention [Pap11a], the parameterized technique has a difficulty with highly complex caustic images, thus Finckh et. al [Fin10a] cannot reproduce all frequencies of the caustic pattern and Anson et. al [Ans08a] assume the shape of the caustic pattern to be circular.

Weyrich et al. generate a microgeometry reflector given a single reflected caustic pattern input [Wey09a]. The caustic object is subdivided into uniform cells (facets), and they compute the optimized orientation of each cell that can produce a caustic pattern similar to the input pattern. Papas et al. [Pap11a] improve the work of Weyrich et al. [Wey09a] by generating a refractor caustic object on a larger scale. Moreover, they are able to prevent noise on the reconstructed caustic pattern by computing the surface of each facet based on the Gaussian distribution. Similar to Weyrich et al. [Wey09a], they employ several optimization costs in order to generate the caustic objects. In the most recent development, Yue et al. [Yue12a] emphasize on modularity by reconstructing an input caustic pattern from a caustic object which consists of many smaller pieces of caustic object cells. Their caustic object cells are divided into ten types with each type refracts light to a predefined direction.

Comparison In all these work, the input is only a single caustic pattern. On the other hand, in this paper we propose a new challenge in which we compute the geometry of a caustic object based on a set of input caustic patterns.

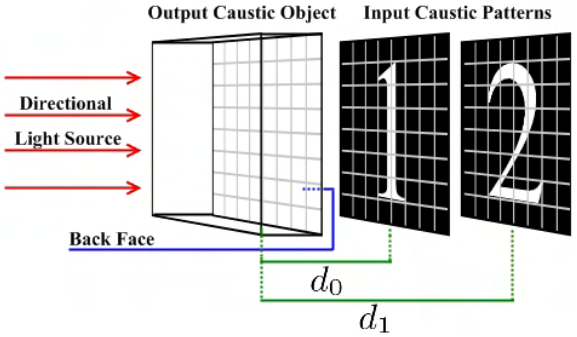


Figure 2: Scene setup. Our algorithm computes the normal/orientation of each caustic object cell. Caustic pattern '1' is formed when the caustic receiver is at distance d_0 from the caustic object and similarly caustic pattern '2' at d_1.

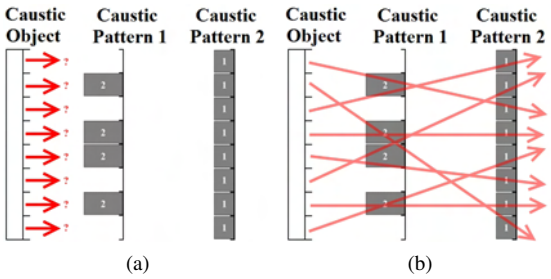


Figure 3: Problem formulation. (a) Given two caustic patterns at two different locations (with the intensity of each caustic cell is denoted by the size of the cell), compute light refraction direction (red arrow) of each caustic object cell such that the refracted light collectively can generate caustic patterns similar to the input caustic patterns. (b) Light refraction combination that can satisfy the input caustic patterns.

3. BASIC IDEA OF OUR METHOD

Scene Setup The scene setup is shown in Figure 2. The scene consists of three components, a caustic object (of a box shape), a caustic receiver (a planar surface where the caustic patterns are formed), and a directional light source (whose direction is orthogonal to the caustic object). Both the caustic receiver and the caustic object are positioned coplanar, with the caustic receiver is on one side of the caustic object (assumed to be the **back face** of the caustic object, facing (0, 0, -1) direction) and incoming light direction is on the other face of the caustic object (assumed to be the front face of the caustic object, facing (0, 0, 1) direction). We assume the caustic receiver and the caustic object to have the same spatial dimension (i.e. same width and height) and orientation. Therefore, the extent of the region of interest of each caustic pattern is bounded by the shape of the caustic receiver

The caustic patterns and the back face of the caustic object are subdivided into a regular grid of cells. Each cell of a caustic pattern stores the total caustic intensity on that particular cell. We call cells of caustic patterns which have non-zero caustic intensity as **caustic cells** and cells with zero caustic intensity as **empty cells**. The collection of caustic cells of a caustic pattern is collectively called as a **caustic region**. For each cell of the caustic object (**caustic object cell**), we compute its orientation such that it can produce a refracted light beam to a specific direction. In the rest of this paper, we refer to each refracted light beam as light and we represent each light beam in the following Figures 3 and 4 as an arrow.

Problem Formulation We compute the back face of a caustic object C given a set of p grayscale caustic patterns such that each caustic pattern j will be formed on the caustic receiver when the receiver is located at the user-input distance d_j from the caustic object. Specifically, we compute the orientation of each caustic object cell at the back face of the caustic object such that the cell refracts the incoming light into a direction that passes through parts of the caustic regions. Collectively, the light refracted from all caustic object cells is expected to pass through all the input caustic cells thus reconstructing the input caustic patterns. As mentioned in Section 1, the input caustic patterns only provide the estimate of the amount of refracted light arriving at caustic receiver cells, not the light directions. Hence, the main challenge is to compute refracted light paths that can approximately reconstruct all the given caustic patterns. This problem is illustrated in Figure 3. The orientation of each cell can then be determined based on the path of its refracted light.

Solution As explained above, the task is to compute refracted light direction combinations such that they can approximately reconstruct the input caustic patterns. In this case, more light is expected to pass through brighter caustic cells compared to darker caustic cells. Hence, to solve this, we simulate the direction of the refracted light of each caustic object cell by using a stochastic approach. The idea is to use the caustic intensity in each caustic cell as the probability that we will refract a light to that caustic cell (i.e. the brighter the input caustic pattern is, the more likely it is chosen as a refracted light target).

We represent the set of caustic patterns as a set of normalized 2D probability mass functions $\mathbf{P} = \{f_{P_0}, f_{P_1}, f_{P_2}, \ldots f_{P_{p-1}}\}$ (each in $\mathbf{P}$ is the pmf of the user-input caustic pattern on the caustic receiver when the receiver is located at the user-input distance from the caustic object). The pmf of each caustic pattern is defined by using the grayscale value (intensity) of the caustic pattern in which the probability at each caustic cell is the grayscale value of that particular cell in the caustic pattern. Each pmf f_{P_j} is normalized by

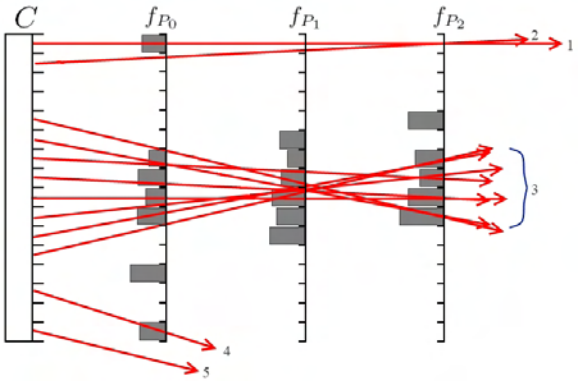


Figure 4: Each numbered arrow denotes the refracted light from the cells of the caustic object C and the gray blocks denote the probability of caustic cells. Light #1 and #2 have joint pmf of zero since their paths pass through at least one empty cell. Each light in #3 has the probability greater than zero since the light pass through non-zero cells. Light #4 is allowed even though it misses some of the caustic patterns. We will explain this further in Section 4.2. Light #5 is not valid because it does not intersect any caustic patterns

dividing the probability of each of its cell with the total probability of all cells of f_{P_j}.

We assign a random variable X_i for each i-th cell of the caustic object. For each X_i, the probability value of each possible refracted light direction $\mathbf{x}$ is computed by multiplying the probability of each caustic cell passed by the light refracted to direction $\mathbf{x}$ (see Figure 4), as shown in Equation 1.

$$f_{X_i}(\mathbf{x}) = \prod_{j=0}^{p-1} f_{P_j}(g_j^i(\mathbf{x})), \tag{1}$$

with $g_j^i(\mathbf{x})$ is a mapping function. The mapping function is basically a ray casting function in which the light is shot from the i-th caustic object cell to the caustic pattern j with the direction of $\mathbf{x}$ and return the intersected cell (of caustic pattern j). Then, for each caustic object cell or each X_i we assign a refracted light direction by using the Acceptance-Rejection method [vN51a] with the distribution based on the sampled joint probability mass function of all caustic patterns (Equation 1).

Once we obtain the refracted light direction for each caustic object cell, we compute its normal or orientation based on the user-input index of refraction of the caustic object, incoming light direction (orthogonal to the caustic object), and the obtained refracted light direction by solving the Snell's Equation (see Appendix A). Afterward, we perform rendering simulation using the mental ray to asses the approximate reconstructed caustic patterns (as shown in Figure 6a).

Note that our technique can also be applied to point light sources and directional light sources non-orthogonal to the caustic object. In the rest of the paper, the terms caustic patterns and pmfs are interchangeable as we use the term pmfs when we emphasize on the mathematical representation of the caustic patterns.

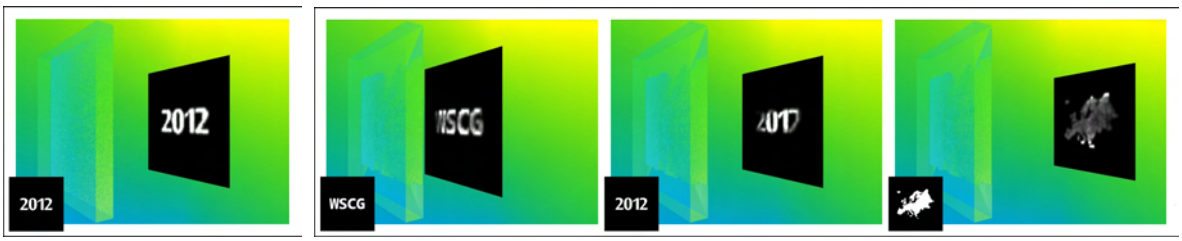
4. IMPROVING THE RECONSTRUCTED CAUSTICS

The solution in Section 3 may not be able to reconstruct the caustic patterns very well if the input consists of multiple patterns. If we only have a single caustic pattern (shown in Figure 5a), then we can reconstruct it very well. However, if we add two additional caustic patterns, then some parts of the input caustic patterns are missing (Figure 5b).

Reconstruction problem As explained in Section 3 (and shown in Figure 4), some refraction directions have zero joint pmf when they pass through at least one empty caustic cell. As a result, if all possible refraction directions from every caustic object cell pass through a caustic cell of a caustic pattern but they also pass through the empty cells of other caustic patterns, then the aforementioned caustic cell cannot be reconstructed (we call such cell as a **missing caustic cell**). As seen in Figure 4, the top and bottom caustic cells in f_{P_1} are missing caustic cells since the refracted light that pass through these caustic cells also pass through empty cells in the other caustic patterns.

Proposed solution Based on the given input caustic patterns and their configurations (positions and sizes), it might not be possible to compute the caustic object that can well reconstruct the original input caustic patterns. Thus, we propose a method to relax the input requirement by allowing slight changes to the positions, sizes, and shapes of the caustic regions. Our proposed method consists of two steps. In the first step, we optimize the size and position of the caustic regions by slightly adjusting the size and position given by the user (Section 4.1). In the second step, the boundaries of each caustic region are adaptively extended such that they enable the reconstruction of the missing caustic cells on the other caustic patterns (Section 4.2). We also compute the amount of light that is allowed to overshoot or to miss some caustic patterns.

In both optimization steps, we use Simulated Annealing [Kir83a]. The main reason we use Simulated Annealing is because the problem cannot be solved analytically. However, there are also some possible optimization techniques such as Particle Swarm, Ant Colony, and Genetic Algorithm. However, those techniques require keeping the record of multiple possible solutions at once, hence it is not efficient for our case (as seen in Equation 3, the cost computation requires reconstruction of the caustic patterns multiple times using the adjusted input caustic patterns). Moreover, in some of the related work [Wey09a, Fin10a, Pap11a], Simulated Annealing is also used. After applying our proposed



(a) (b)

Figure 5: (a) Only a single caustic pattern and it can be reconstructed very well. (b) Two additional caustic patterns cause some parts of the input caustic patterns to be missing.

solution, the reconstructed caustic patterns from the test case in Figure 5b are improved as seen in Figure 1b.

Cost computation In every iteration of both optimizations, we use the root mean square to compute the cost or degree of possibility that the adjusted input caustic pattern can be approximately reconstructed (in order to guide the simulated annealing). The root mean square is computed as the difference between the normalized reconstructed caustic patterns $\mathbf{Z} = \{f_{Z_0}, f_{Z_1}, \dots f_{Z_{p-1}}\}$ and the normalized adjusted input caustic patterns $\mathbf{D} = \{f_{D_0}, f_{D_1}, f_{D_2}, \dots f_{D_{p-1}}\}$. Only in the cost computation here, the normalization of caustic patterns in $\mathbf{Z}$ and $\mathbf{D}$ are computed by dividing the value of each caustic cell $\mathbf{t}$ with the maximum caustic cell value of the caustic pattern $\mathbf{t}$ belongs to. We compute the cost in this way such that the maximum cost (which is in the worst case scenario, for example the input caustic patterns are not reconstructed at all) is 1.0. The cost computation is shown in Equation 3.

$$Cost = \frac{1}{p}\sum_{j=0}^{p-1}\sqrt{\frac{1}{n(\mathbf{W}(j))}\sum_{i=0}^{\beta}(f_{D_j}(\mathbf{t}_i) - f_{Z_j}(\mathbf{t}_i))^2}, \quad (2)$$

with

$$\mathbf{W}(j) = \{\mathbf{t} | f_{D_j}(\mathbf{t}) + f_{Z_j}(\mathbf{t}) > 0\}, \quad (3)$$

and $\mathbf{t}$ is the caustic cell, p is the number of caustic patterns, $n(\mathbf{W}(j))$ is the number of elements of a set of caustic cells contributing to the cost computation, and β is the total number of cells of each caustic pattern (in our experiments, $\beta = 64 \times 64 = 4096$). For more accurate computation of $\mathbf{Z}$, we approximately reconstruct the caustic patterns 32 times and accumulate their caustic cell values, and finally we divide the value of each caustic cell by 32 in order to get $\mathbf{Z}$.

We reconstruct the caustic patterns by computing the refracted direction as explained in Section 3 and then for each caustic cell we accumulate the amount of refracted light that intersects it. We use the modified pmfs (which are adjusted in each optimization step) to compute the joint pmfs (Equation 1).

4.1. Adjusting the Size and Position

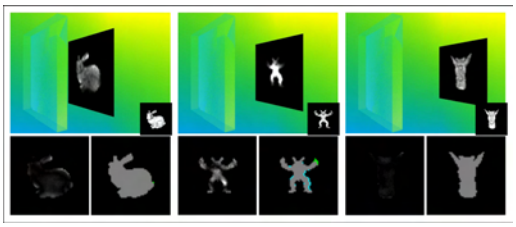
In this first optimization step, we relax the input caustic pattern configurations by iteratively adjusting the size and position of the input caustic region. Adjusting the position is basically translating the caustic regions in 3D space (translation in x, y, z). This means we also adjust the input distance (translation in z) between the caustic object and the caustic pattern (caustic receiver).

In every iteration, we adjust the size and positions of the input caustic regions and compute the cost by using Equation 2 in order to guide the optimization iterations. The adjusted input caustic patterns are used as the input to the next optimization step (Section 4.2) and they are also the target caustic patterns for every iteration of the second step.

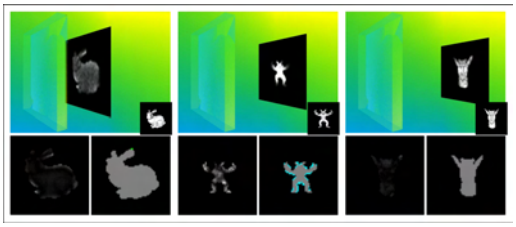
4.2. Extending Caustic Regions and Overshooting Refracted Light

After performing the first optimization step, there might be some missing caustic cells left. Missing caustic cells are the caustic cells that cannot be reconstructed. To reconstruct some of these missing caustic cells, we slightly extend the shape of all input caustic regions. For example, in Figure 4, the middle-top and middle-bottom caustic cells of f_{P_1} cannot be reconstructed since all possible refracted light that passes through these cells in f_{P_1} have to pass through empty cells in either f_{P_0} or f_{P_2}. Hence, to solve this, we can extend the middle caustic regions in f_{P_0} and f_{P_2} up and down by one cell.

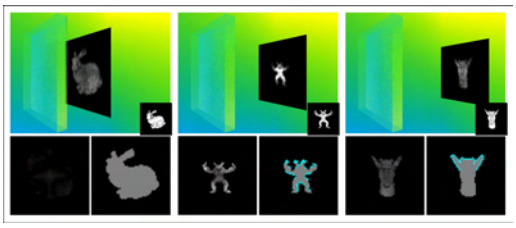
Some caustic cells especially around caustic patterns borders are hard to reconstruct as most light passing through these cells can miss other caustic patterns in behind. Thus, we relax this requirement by enabling some of the refracted light that passes through caustic cells of one or several caustic patterns to miss the rest of the caustic patterns (or region of interests of the rest of the caustic patterns). This is beneficial especially for the caustic cells on the border of the caustic patterns. For example, in Figure 4, if we enable light #4 to pass the bottommost caustic cell of f_{P_0} and miss the rest of caustic patterns (we call this **overshoot**), then



(a) Results without optimization. Cost : 4.45×10^{-1}



(b) Results after 1st optimization (Section 4.1). Cost : 4.08×10^{-1}

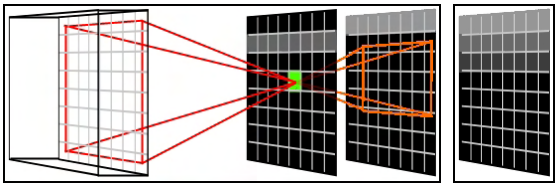


(c) Results after 1st and 2nd optimization (Section 4.2). Cost : 2.64×10^{-1}

Figure 6: Mental ray rendering results of the optimization steps. Input caustic patterns are shown at the bottom right of each screenshot in (a). We also show the missing caustic cell maps at the below right of each image (green cells show the missing caustic cells, gray cells show the caustic cells that can be reconstructed, and cyan cells show the extended caustic cells). For the visualization of the differences between the target and the reconstructed caustic patterns, we also show the caustic irradiance difference maps (assuming the total irradiance of each target caustic pattern is 1.0 and the total light emitted to the scene is 1.0, i.e. each caustic object cell refracts the light with the amount of 1.0 divided by the number of caustic object cells) at the below left of each image (from the darkest pixels with the least errors to the brightest pixels with the most errors). For the sake of visual clarity, we scale up the difference values by 5000. The computational time is 5.7 hours.

the bottommost caustic cell of f_{P_0} can be reconstructed. However, we still do not allow the refracted light of one caustic object cell to miss all of the caustic patterns (as in light #5).

Fully extending the caustic regions can deform the original caustics too much, and likewise if we allow too much light to overshoot the caustic patterns then the approximate reconstructed caustic patterns will have very low intensity. Hence, in this step, we apply an optimization to determine the appropriate caustic regions extensions amount $\mathbf{k} = \{k_0, k_1, \ldots k_{p-1}\}$ and light overshoot amount $\mathbf{o} = \{o_0, o_1, \ldots, o_{p-1}\}$ with $\mathbf{k}$ and $\mathbf{o} \in [0,1]$ (i.e., a k and an o value for each caustic pattern).



(a) (b)

Figure 7: (a) A simple example of missing caustic cell projection (is explained in Section 4.2). Gray cells are the caustic cells and the green cell is the missing caustic cell. (b) We extend the second caustic pattern (two cells away) with gradually decreasing intensity.

Extending Caustic Regions To enable the missing caustic cells of a caustic pattern j to be reconstructed, we have to firstly compute at most how many s_b unit cells away the caustic region boundaries of the other caustic patterns b ($0 \leq b \leq p-1, b \neq j$) have to be extended. Afterward, for every caustic pattern b, we extend its caustic region with the amount of $s_b \cdot k_b$. In order to enable smooth extension, we extend the caustic regions with linearly decreasing intensity (or probability value).

To do this, for every missing caustic cell of the caustic pattern j, we project it from every caustic object cell to the empty cells of other caustic patterns b ($0 \leq b \leq p-1, b \neq j$). We perform this projection for the missing caustic cells of all caustic patterns. This projection example is shown in Figure 7a in which we project the missing caustic cell (shown in green color). Afterward, for each caustic pattern b, we obtain the maximum distance (s_b) between its caustic region boundaries and its empty cells that receive the projections of the missing caustic cells (of other caustic patterns). In Figure 7a example, it is five cells ($s_b = 5$) away for the second caustic pattern and in Figure 7b the caustic region boundary is extended two cells away (if $k_b = 0.4$).

Some of the missing caustic cell projections might miss the other caustic patterns b. For example, if the missing caustic cell in Figure 7a is one or two cells to the right, then some of the projections will overshoot or will not hit the second caustic pattern. We use this information to control the possibility of the refracted light to overshoot each caustic pattern.

Overshooting Refracted Light To improve the results, we also enable the refracted light to overshoot some of the caustic patterns. Thus, during the missing caustic cells projections, we also compute the ratio (e_b) between the amount of these projections that do not hit caustic pattern b and the total amount of these projec-

tions toward caustic pattern b (i.e. sum of the missing caustic cell projections that hit and do not hit caustic pattern b). e_b is essential since it provides the information on the amount of probability that the refracted light miss plane b. Hence, the probability h_b that we will refract the light to miss the caustic pattern b is shown in Equation 4.

$$h_b = e_b \cdot o_b \cdot f_{P_b}(\mathbf{t}_{max}), \tag{4}$$

with o_b is the coefficient to control the probability of overshooting b-th caustic pattern and $\mathbf{t}_{max}$ is a cell of f_{P_b} with the highest probability value.

We show the optimization progression in Figure 6.

5. GEOMETRY CONSTRUCTION

As explained in Section 3 we use the joint pmf (Equation 1) of the caustic patterns to compute the refraction direction of each caustic object cell. From the refraction direction, we can obtain the normal of the particular caustic object cell. However, if we perform the optimizations in Section 4, then we use the modified pmfs (output from both optimization steps) to compute the joint pmf (and ultimately the normal of each caustic object cell).

Based on the computed orientation of each caustic object cell, we can obtain the caustic object geometry by computing the x,y,z coordinates of the four corners of each caustic object cell. The x,y coordinates of each caustic object cell corner can be easily found as the caustic object is uniformly subdivided. To compute the z coordinates of the caustic object cell corners, we firstly assume that the z coordinate of all caustic object cell middle points to be the same ($z = 0.0$). Next, we use the dot product operation, i.e. dot product between the normal of the caustic object cell and the vector from the caustic object cell middle point (we know its x,y coordinates from the uniform subdivision and also its z coordinate which is 0.0) to each caustic object cell corner (we know its x,y coordinates) must be equal to zero. In this case, the only unknown value is z of the caustic object cell corner.

Since the edges of the neighbouring caustic object cells do not have the same slope or the same z coordinate on both endpoints, there are vertical open spaces between caustic object cells (shaded with lighter gray in Figure 8), we generate additional polygons to close those gaps.

6. RESULTS

We present some results computed using our technique in Figures 1, 6, 10. In all of the test cases, the index of refraction of the caustic objects are 1.5, the resolution of the caustic objects are 128×128 and the resolution of the caustic receiver is 64×64. Resolution refers to

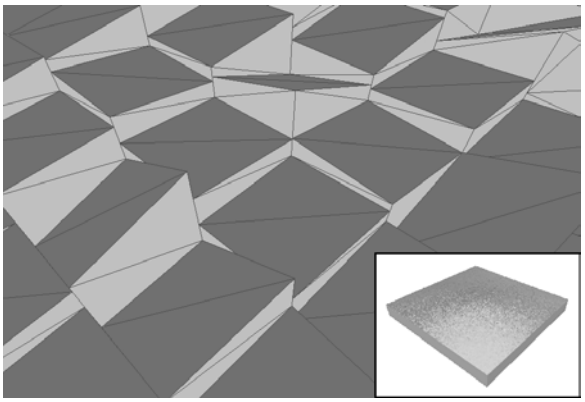
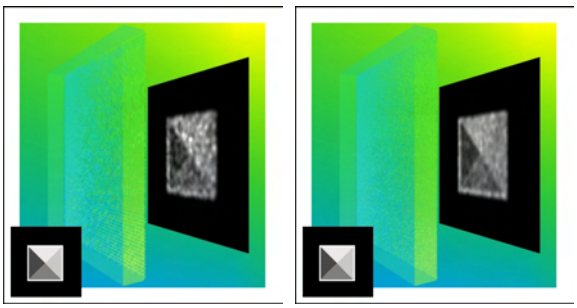


Figure 8: Caustic object geometry (inset) of Figure 6c with a zoom-in view. In the zoom-in view, each caustic object cell consists of two co-planar triangles shaded with darker gray. We also generate additional vertical polygons (shaded with lighter gray) to close the gaps between caustic object cells.

the number of cells. If we assume the spatial size to be 1.0×1.0, then the size of each caustic object cell is $1.0/128 \times 1.0/128$ and the size of each caustic cell is $1.0/64 \times 1.0/64$.

We use higher resolution for caustic object cells since we want to have more variations on the refracted light paths so that we can better reconstruct the contrast (or intensity variations) of the given caustic patterns. We show a difference example with a single caustic pattern case in Figure 9, a simple caustic pattern (resolution 64×64) reconstructed with a resolution 64 x 64 caustic object and a resolution 128 x 128 caustic object.



(a) (b)

Figure 9: A simple test case (one caustic pattern, with resolution 64×64) reconstructed with different caustic object resolutions. (a) Resolution 64 x 64 caustic object. (b) Resolution 128 x 128 caustic object.

We use the same parameters for the simulation annealing for both optimization steps in all experiments, i.e. 10 cycles of 10 iterations, Boltzmann's constant of 1.0, and temperature reduction factor of 0.5.

The experiments were performed on two comparable PCs. The specification of the first PC is Intel i7 920 2.67 GHz (CPU) with NVIDIA GeForce GTX 285

(GPU) and the specification of the second PC is Intel i7 880 3.07 GHz (CPU) with NVIDIA GeForce GT 330 (GPU). In the implementations, we calculate the joint pmf by rendering each caustic pattern and then we multiply them by using alpha blending (hence the use of GPU). For the rest of the computations such as Simulated Annealing and Acceptance-Rejection method, we perform them on CPU.

From the results, we can observe that the caustic objects generated using our technique can approximately reconstruct various types of input caustic patterns, especially for the WSCG (Figure 1b), fruits (Figure 10a), and rotating star (Figure 10c) test cases. The degree of difficulty in reconstructing the caustic patterns mostly depends on the number of caustic patterns, similarity between shapes, and the number of caustic cells in the input caustic patterns. As shown in Equation 4, due to the multiplication in computing the joint pmf, the probability of a particular refraction direction can become zero if the refracted light passes through empty cells. With the increasing number of caustic patterns especially the ones with different shapes, the chances that we have many refraction directions with zero probability also increase. We can see this from the results in Figures 1 and 6 (three input caustic patterns) where we can reconstruct better compared to the results in Figures 10 (four or more caustic patterns).

The shapes and orientations of caustic patterns can also affect the reconstruction difficulty. The test cases with similar caustic patterns, can be approximately reconstructed pretty well since similar refracted light paths are sufficient to reconstruct the caustic patterns. This is evident by comparing Figure 10a and Figure 10b. The caustic patterns in Figure 10a have near round shapes and approximately the same orientations. In contrast, the test case in Figure 10b have pretty different shapes (and orientations). Hence, there are few light that can pass through the endpoints of the bars compared to the middle regions of the bars.

Note the difficult test case shown in Figure 10b, four bar caustic patterns with alternating orientations. As we can see, the hardest parts to reconstruct are the ones near the two endpoints of the bars and on the other hand the parts around the centers are the easiest. This is due to the alternating shapes which cause the light paths to always pass through the center of the caustic regions. As we allow the refracted light to miss some of the caustic patterns, we are able to approximately reconstruct the top and bottom parts of the first caustic pattern. However, the consequence is that the last caustic pattern appears much dimmer.

In many cases, light tends to converge to the middle caustic patterns and as a result the center regions of these caustic patterns become relatively brighter compared to the other caustic patterns (for example, the Armadillo caustic pattern in Figure 6 exhibits this effect). This is due to two reasons. First, the caustic regions are positioned approximately at the center of the caustic patterns. Second, because the size of the caustic regions are mostly smaller than the caustic object. Therefore, some caustic object cells have to refract the light in the diagonal directions. This is illustrated in Figure 4 in which some of the light grouped to #3 have to be refracted in the diagonal directions.

Note that the relative depth of caustic patterns also affects the quality, as it is very difficult to reconstruct if the caustic patterns are located very near to each other. This is because the refracted light paths will intersect the patterns at similar locations and thus it is difficult to reconstruct caustic patterns with different shapes.

Please refer to the submitted **video** to see the progressive changes of the caustic patterns as the caustic receiver is moved.

7. APPLICATIONS

The inverse caustics has several potential applications.

Arts As shown in Figure 10, our technique can generate caustic objects that can produce several interesting caustic effects (similar to the intention in the inverse shadow [Mit09a]). Therefore, we hope that our work can encourage more exploration in caustic arts.

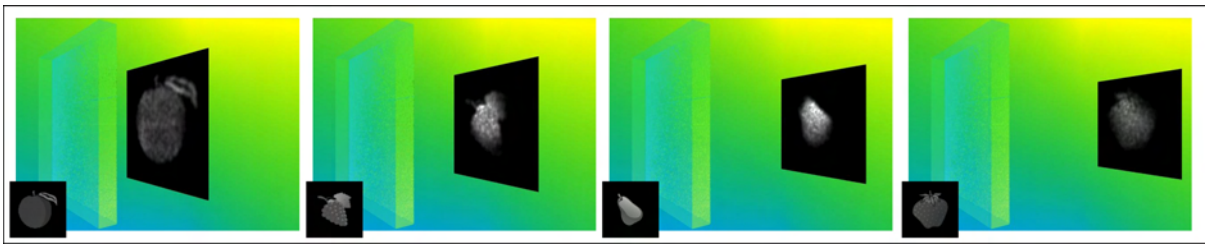
Information Encoding Information (such as serial numbers, passwords) can be encoded as caustic patterns of encrypted 2D images. Only when the requirements (such as light direction and caustic receiver distance) are known, we can recover the original information. We show an example in Figure 11 in which we encrypt **WSCG** and **2012** into two QR barcode patterns.

Validation Tests By using the computed caustic object, we can validate some processes such as rendering process (validating the correctness of caustics rendering algorithm) or manufacturing (validating the quality of produced glasses or light sources).

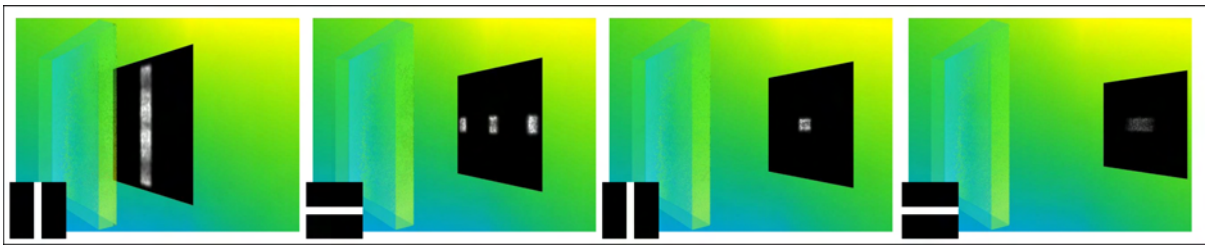
8. CONCLUSIONS AND FUTURE WORK

We have presented an inverse caustic problem and a novel technique which computes a caustic object given a set of caustic patterns with each pattern is positioned at a user-input distance from the caustic object. Our proposed technique is based on a stochastic approach, and it is augmented with two optimization steps that can alleviate the missing caustic problems. We have validated our results by performing physically rendering simulation using mental ray, and the caustic object generated using our technique can approximately reconstruct various types of input caustic patterns.

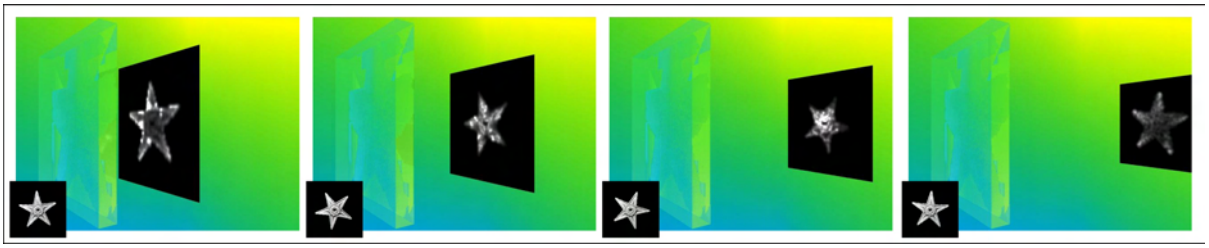
In the future, we would like to improve the quality of the reconstructed caustics in terms of smoothness. It



(a) Fruits (four caustic patterns). Computational time : 7.7 hours.



(b) Four bars (four caustic patterns). Computational time : 15.7 hours.



(c) Rotating Star (nine caustic patterns). Computational time : 27.6 hours.

Figure 10: More results. Note that the caustic pattern set in (c) contain similar patterns, as they are frames of a simple animation.

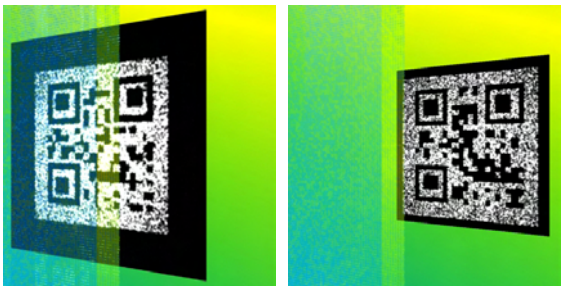


Figure 11: An application of information encoding. We encode **WSCG** and **2012** into two QR barcode patterns.

is also interesting to consider more complex light situations such as area light sources and dynamic light sources. It is challenging to use area light sources since they emit light to many directions from every point in the area light sources. As for the dynamic light sources, it is interesting to generate unique caustic pattern for each given light source direction (in this case, caustic object and caustic receiver are static). Last but not least, we would like to fabricate a real caustic object based on the computed geometry.

9. ACKNOWLEDGEMENTS

This work has been partially supported by the National Research Foundation grant, which is administered by the Media Development Authority Interactive Digital Media Programme Office, MDA (IDMPO). We would like to express our gratitude to Stanford Computer Graphics Laboratory for the 3D models (bunny, and armadillo) which are used to generate the input caustic patterns.

10. REFERENCES

[Ans08a] Anson, O., Seron, F.J., and Gutierrez, D.. NURBS-Based inverse reflector design. in CEIG08: Congreso Español de Informática Gráfica, Eurographics Association, Barcelona, Spain2008. pp. 65–74.

[Bot01a] Bottino, A., Cavallero, L., and Laurentini, A.. Interactive reconstruction of 3-D objects from silhouettes. in WSCG. 2001. pp. 230–236.

[Fin10a] Finckh, M., Dammertz, H., and Lensch, H.P.A.. Geometry construction from caustic images. in Proceedings of the 11th European Conference on Computer Vision: Part V, Springer-Verlag, Berlin, Heidelberg2010. ECCV'10. pp. 464–477.

[Kir83a] Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P.. Optimization by simulated annealing. Science. vol. 220, no. 4598, pp. 671–680, 1983.

[Mas09a] Mas, A., Martín, I., and Patow, G.. Fast inverse reflector design (FIRD). Computer Graphics Forum. vol. 28, no. 8, pp. 2046–2056, 2009.

[men12a] mental ray. http://www.mentalimages.com/products/mental-ray.html, 2012.

[Mit09a] Mitra, N.J., and Pauly, M.. Shadow art. ACM Trans. Graph.. vol. 28, pp. 156:1–156:7, 2009.

[Pap11a] Papas, M., Jarosz, W., Jakob, W., Rusinkiewicz, S., Matusik, W., and Weyrich, T.. Goal-based caustics. Computer Graphics Forum. vol. 30, no. 2, pp. 503–511, 2011.

[Pat04a] Patow, G., Pueyo, X., and Vinacua, A.. Reflector design from radiance distributions. World Scientific. vol. 10, no. 2, pp. 211–235, 2004.

[Pat07a] Patow, G., Pueyo, X., and Vinacua, A.. User-guided inverse reflector design. Computers & Graphics. vol. 31, no. 3, pp. 501–515, 2007.

[vN51a] von Neumann, J.. Various techniques used in connection with random digits. monte carlo methods. vol. 12, pp. 36–38, 1951.

[Wey09a] Weyrich, T., Peers, P., Matusik, W., and Rusinkiewicz, S.. Fabricating microgeometry for custom surface reflectance. ACM Trans. Graph.. vol. 28, pp. 32:1–32:6, 2009.

[Yue12a] Yue, Y., Iwasaki, K., Chen, B., Dobashi, Y., and Nishita, T.. Pixel art with refracted light by rearrangeable sticks. Computer Graphics Forum. vol. 31, no. 2, pp. 575–582, 2012.

A. NORMAL COMPUTATION

Given the Snell's Equation

$$\begin{aligned} \eta_1 \sin\theta_1 &= \eta_2 \sin\theta_2, \\ \eta\sqrt{1-\cos^2\theta_1} &= \sqrt{1-\cos^2\theta_2}, \\ \eta\sqrt{1-(-\mathbf{N}\cdot\mathbf{M})^2} &= \sqrt{1-(\mathbf{N}\cdot\mathbf{R})^2}, \\ A_1N_xN_x + A_2N_xN_y + A_3N_yN_y &= 1, \end{aligned} \tag{5}$$

with η_1 is the index of refraction of the caustic object, η_2 is the index of the refraction of air, $\mathbf{N}$ is the normal, $\mathbf{M}$ is the inverse incoming light direction, $\mathbf{R}$ is the refracted light direction, θ_1 is the angle between $\mathbf{M}$ and the inverse normal, θ_2 is the angle between $\mathbf{M}$ and the normal, and

$$\eta = \frac{\eta_1}{\eta_2} \qquad A_1 = \frac{\eta^2 M_x M_x - R_x R_x}{\eta^2 - 1}$$
$$A_2 = 2\frac{\eta^2 M_x M_y - R_x R_y}{\eta^2 - 1} \qquad A_3 = \frac{\eta^2 M_y M_y - R_y R_y}{\eta^2 - 1}$$

Since the normal vector $\mathbf{N}$ is normalized, the solution lies on a unit circle and as a result $N_x = \cos\phi$ and $N_y = \sin\phi$. Hence, Equation 5 can be simplified to

$$(2A_3 - 2)\tan^2\phi + 2A_2\tan\phi + (2A_1 - 2) = 0. \tag{6}$$

Equation 6 is basically a quadratic equation and the angle ϕ can be obtained by solving the quadratic equation.

Interactive BRDF Estimation for Mixed-Reality Applications

Martin Knecht
Vienna University of Technology
knecht@cg.tuwien.ac.at

Georg Tanzmeister
Vienna University of Technology
georg.tanzmeister@tuwien.ac.at

Christoph Traxler
VRVis Zentrum für Virtual Reality und Visualisierung Forschungs-GmbH
traxler@vrvis.at

Michael Wimmer
Vienna University of Technology
wimmer@cg.tuwien.ac.at

ABSTRACT

Recent methods in augmented reality allow simulating mutual light interactions between real and virtual objects. These methods are able to embed virtual objects in a more sophisticated way than previous methods. However, their main drawback is that they need a virtual representation of the real scene to be augmented in the form of geometry and material properties. In the past, this representation had to be modeled in advance, which is very time consuming and only allows for static scenes.
We propose a method that reconstructs the surrounding environment and estimates its Bidirectional Reflectance Distribution Function (BRDF) properties at runtime without any preprocessing. By using the Microsoft Kinect sensor and an optimized hybrid CPU & GPU-based BRDF estimation method, we are able to achieve interactive frame rates. The proposed method was integrated into a differential instant radiosity rendering system to demonstrate its feasibility.

Keywords

BRDF estimation, reconstruction, augmented reality

1 INTRODUCTION

Many mixed-reality applications require or at least desire a consistent shading between virtual and real objects. Examples are product presentations, virtual prototyping, architectural and urban visualizations and edutainment systems. Here virtual objects should smoothly blend into the real environment and provide a plausible illusion for users. They need to be rendered in a way that makes them hard to distinguish from real objects. Some recently published methods [14, 7] consider the mutual light interaction between real and virtual objects, so that they indirectly illuminate or shadow each other.

Beside the geometry of the scene and the real lighting conditions, the BRDFs of real objects are needed to simulate these mutual shading effects. Acquiring this data in a pre-processing step would diminish the dynamic and interactive nature of mixed-reality systems, and would also make it necessary to track the previously modeled movable real objects. In this paper, we introduce a BRDF estimation method that runs at interactive frame rates. It is based on real-time reconstruction using the structural light scanner provided by Microsoft's Kinect sensor [11]. The real lighting conditions are captured by a camera with a fish-eye lens from which light sources are derived.

Our contribution is best characterized by the unique features of our BRDF estimation approach, which are:

- It runs at interactive frame rates.
- It does not need any pre-processing.
- It utilizes a novel K-Means implementation executed on the GPU.

2 RELATED WORK

BRDF estimation has a long history of research and a variety of methods have been presented. Our approach belongs to the class of image-based methods, which are sometimes synonymously called *Inverse Rendering*. These methods try to fit parameters of an underlying, sometimes rather simple, BRDF model, like

the Phong [15] or Ward model [20], from images of a scene. Yu et al. introduced *Inverse Global Illumination* [23], where reflectance properties are derived from a sparse set of HDR images considering also indirect illumination. The geometry is pre-modeled and partitioned into surfaces with similar materials. The direct light sources must also be known. An optimization algorithm then calculates diffuse and specular components separately. Although the concept is sound and forms the basis of newer algorithms, it needs a lot of manual pre-processing. Sato et al. [17] presented a method that also performs a reconstruction of the object's geometry from range images, which is then used to estimate diffuse and specular parameters from the same images.

Boivin and Gagalowicz [2] use a single LDR image in addition to a geometric model including light sources. Starting with a Lambertian model, they iteratively compare renderings with the original image and consider more and more complex reflectance models as long as the difference is too large. Though their solution is scalable with regard to accuracy, it is still time consuming and requires pre-processing. Mercier et al. [10] were the first to present a fully automatic method to recover the shape and reflectance properties of a single object and the position of light sources from a set of calibrated images. For that purpose, the object and light sources are fixed on a turntable, and photographs are taken every 5 degrees. The geometry is approximated by *Shape From Silhouette* (SFS) from Szeliski [19]. The method is very accurate and does not need any preprocessing, but the special setup makes it unsuitable for mixed-reality. Xu and Wallace [22] used a depth sensor and a stereo intensity image to acquire an object's reflectance properties and parameters for multiple light sources. Although using a depth map comes close to our approach, their method is restricted to a single object. Furthermore, calculating light source parameters from intensity images introduces inaccuracies for flat surfaces.

Zheng et al. [25] presented a solution that is similar to that of Mercier et al. [10]. One big difference is that they use measured lighting conditions instead of deriving this information from the images, which minimizes the estimation error. They then apply the highlight removal algorithm from Ortiz and Torres [13] before clustering images into regions with similar diffuse materials using K-Means. The parameters of the Ward model are then obtained for each cluster by non-linear optimization. Their algorithm is very robust, since after estimating specular factors, diffuse factors are re-estimated in order to compensate for errors caused by wrong clustering or inaccurate geometry.

Like Mercier's method, the approach is based on a controlled setup, which does not meet our requirements. This especially concerns reconstruction by SFS and measurement of the light source. Their estimation pipeline however is very efficient and so we based our work on it. For example we also use an adaptation of the highlight removal technique from Ortiz and Torres [13] and we also use K-Means [9] for clustering.

Several efficient implementations of the K-Means algorithm on the GPU already exist. Almost all of them use a hybrid GPU/CPU approach, where the new cluster centers in each iteration are either entirely or at least partially calculated on the CPU [5, 8, 24, 21]. In all of the aforementioned papers CUDA is used to perform the calculations on the GPU.

To our knowledge there is only one method which was proposed by Dhanasekaran and Rubin [4] where the whole K-Means algorithm is done entirely on the GPU eliminating the need of continuously copying data via the PCIe bus. However, in contrast to Dhanasekaran and Rubin's work which relies on OpenCL, we use a different approach that utilizes mipmaps to calculate the center of each cluster using DirectX.

Generally speaking, all these previous image-based BRDF estimation methods work off-line and have running times ranging from a couple of minutes to several hours. Furthermore they are restricted to static scenes. Mixed-reality applications are highly interactive and dynamic according to Azuma's definition [1]. Hence our motivation was to design and develop a method that runs at interactive frame rates and can thus handle highly dynamic scenes.

3 OVERVIEW

Estimating material characteristics for mixed-reality applications is a challenging task, due to several constraints. On top of it, is the time constraint, since the applications have to be interactive. Then the observered scenes usually exhibit a certain degree of dynamics and materials that just appeared in the camera frustum need to be estimated immediately. As described in the introduction several methods for BRDF estimation exist but all of them are designed for offline purposes. They all try to get a very accurate BRDF estimation. In our case this goal must be lowered to achieve interactive frame rates. The resulting diffuse and specular reflectance maps are used in a differential instant radiosity (DIR) system where the goal is to get visually plausible images instead of physically correct ones. Mapping this idea to our BRDF estimation method, our goal is to find BRDFs that emphasize the same visual cues to the user as the real materials would do.

Our BRDF estimation algorithm is mainly influenced by the ideas of Zheng et al. [25]. Their method was designed to work offline and had thus different requirements. As an adaption we modified their approach where necessary and made extensive use of the GPU

to gain interactive frame rates. The presented method can be divided into two main parts:

- Data acquisition: Capture data from the Microsoft Kinect sensor and a fish-eye lens camera to obtain color, geometry and lighting information.
- BRDF Estimation: Enhance the RGB input data and estimate diffuse and specular material characteristics.

Figure 1 illustrates the separate steps in the context of the two main parts of the method. Section 4 describes the data acquisition. Here normals are obtained with the use of the depth map we get from the Kinect and the lighting environment is approximated with the input stream of the fish-eye lens camera. The BRDF estimation is described in Section 5 where after intermediate steps the final diffuse and specular reflectance parameters are estimated. The output of our method is integrated into a DIR rendering system [7] and we directly render the fully defined geometry (including material characteristics) into a G-Buffer which stores the 3D position, the normal, the color and the material parameters needed for Phong shading.

4 DATA ACQUISITION

The Microsoft Kinect sensor is a relatively cheap device to capture a video and a depth stream simultaneously. The resolution of both streams is 640×480 pixels at a frame rate of 30Hz. Surrounding objects can be captured in a range between 0.45 and 10 meters. Figure 2 shows the input data provided by the Kinect sensor. The other source of input data is a IDS uEye camera with a fish-eye lens attached to capture the incident illumination.

4.1 Normal Estimation

We implemented two methods for normal estimation. The first one uses the Point Cloud Library (PCL) [16]. While the normals are of high quality, their computation takes too much time. The PCL functions need 223 milliseconds for one normal estimation step with a smoothing factor of 10. The estimation is performed on the CPU and therefore we implemented our own GPU normal estimation method that exploits temporal coherence (TC) between adjacent frames in a similar way as done by Scherzer et al. [18].

Our normal estimation is based on two render passes. The first pass performs subsampling and averaging of the normals from the previous frame. Furthermore, a curvature coefficient is calculated. The subsampling causes a smoothing on the normals of the previous frame. Let (i, j) be the row and the column of a given pixel in the previous frame. The average normal is

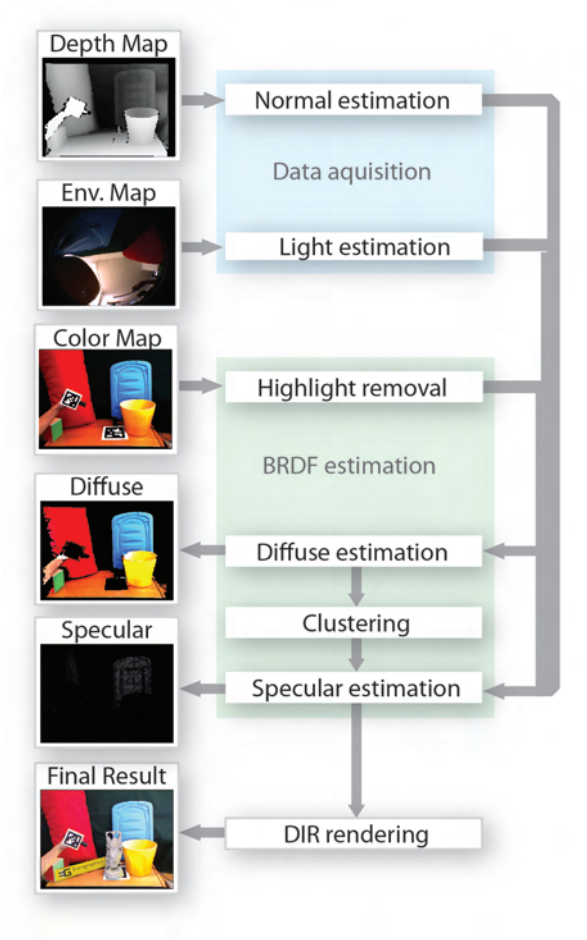


Figure 1: Shows the main steps in the BRDF estimation pipeline. Operations related to data acquisition are shown in the blue box (Section 4). Steps belonging to BRDF estimation and are shown in the green box (Section 5).

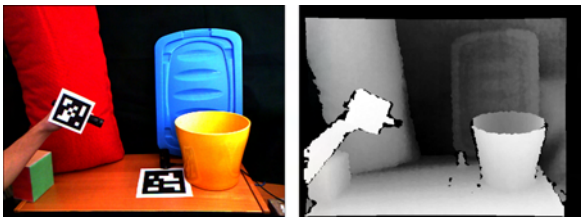


Figure 2: The left image shows the video input stream. The right image shows the normalized depth input stream. Both streams have a resolution of 640×480 with a frame rate of 30Hz.

then calculated by averaging over $(i-1, j)$, $(i+1, j)$, $(i, j-1)$ and $(i, j+1)$. Note that if no normal is available at a given pixel location, it will be discarded from the calculation. The curvature coefficient is calculated as follows:

$$curv_H(i,j) = N_{i-1,j} \cdot N_{i+1,j} \quad (1)$$

$$curv_V(i,j) = N_{i,j-1} \cdot N_{i,j+1} \quad (2)$$

$$curv(i,j) = \min\left[curv_H(i,j), curv_V(i,j)\right]^{128} \quad (3)$$

where the dot is the dot product operator. Note that the curvature coefficient goes to zero at sharp edges and to one at flat areas. The average normal and the curvature coefficient of the last frame are rendered to a render target with half the dimension of the rendering window.

The second rendering pass consists of two steps. In the first one a new normal is calculated from the point cloud delivered by the Microsoft Kinect sensor. We look up the 3D position $p_{i,j}$ at the current pixel (i,j) and two neighboring positions in horizontal $(i,j+4)$ and vertical $(i+4,j)$ direction. A distance value of four pixels showed good smoothing characteristics while edges were still preseverd. From these values, we can set up a surface normal as follows:

$$d_{i+4,j} = \frac{p_{i+4,j} - p_{i,j}}{|p_{i+4,j} - p_{i,j}|} \quad (4)$$

$$d_{i,j+4} = \frac{p_{i,j+4} - p_{i,j}}{|p_{i,j+4} - p_{i,j}|} \quad (5)$$

$$normal_{i,j} = d_{i+4,j} \times d_{i,j+4} \quad (6)$$

In the second step the information calculated by the first rendering pass is used to calculate an old average normal. First the lookup coordinates are calculated by using reprojection. In this way the camera movement from one frame to another can be canceled out. The curvature coefficient at the current pixel steers the mipmap level for the lookup of the previous normal. The new and the previous normal vectors are linearly combined depending on a confidence value calculated as follows:

$$c_N = |N_p \cdot N| \quad (7)$$

$$c = c_B * c_N + (1 - c_N) \quad (8)$$

where N_p is the previous averaged normal and N is the new normal. c_N is the confidence coefficient based on the similarity of the previous and the new normal. The resulting confidence is a linear blend between a base confidence c_B and 1, steered by c_N. To deal with disocclusions occurring during camera movement, we set the confidence value c to zero if the depth difference between the old frame and the new frame is larger than 0.1 meters. In this way, normals at dynamic elements get updated faster.

While the quality of the normals is not that high compared to the results of the PCL, our proposed method runs on the GPU and is thus faster (see Section 6). Furthermore note that the reprojection quality heavily depends on the tracking quality.

4.2 Light Estimation

The incoming light position must be known in order to be able to estimate a BRDF. The fish-eye camera captures the environment map and the DIR rendering system creates a dome of virtual point light (VPL) sources above the scene. We select a subset of these VPLs from the dome and use them for BRDF estimation. The selection criteria are that the VPLs have a high intensity and that there is no other selected VPL within certain distance.

5 BRDF ESTIMATION

Similar to Zheng et al. [25] highlights in the input image are removed and afterwards inverse diffuse shading is applied. However, in their approach the resulting buffer was just used for clustering. In contrast we also use this buffer as a diffuse reflectance map to keep the computation time low.

5.1 Highlight Removal

To estimate specular reflectance values similar colors need to be clustered since they are assumed to belong to the same material. However, specular highlights would form a separate cluster due to saturation, which is not desired. Our highlight removal is based on the work of Ortiz and Torres [13]. Instead of transforming the camera color image into the L1-Norm, we use the Hue Saturation Intensity (HSI) color space. Highlights should be detected at pixels where the color has high brightness but low saturation. As thresholds we set the minimum brightness to 0.9 and the maximum saturation to 0.1. In a first pass, the highlight detection result is written into a binary mask with a one where the brightness and saturation criteria are met and a zero otherwise. Then a morphological dilation with a disk (radius of 4 pixels) is performed. While Ortiz and Torres [13] perform a *Morphological Vectorial Opening by Reconstruction*, we use a rather simplistic reconstruction method. For each pixel that is masked as a highlight, a new color has to be found that ideally matches surrounding colors. We do this by iterating through neighboring pixels in an increasing circular manner until a pixel is found that is not masked as belonging to a highlight anymore. Then the color of the found pixel is used to substitute the color of the masked pixel. In this way, all highlights can be canceled out. Note that due to this highlight removal process, bright and weakly saturated objects may get misinterpreted as highlights. The results of the highlight removal operation are shown in Figure 3.

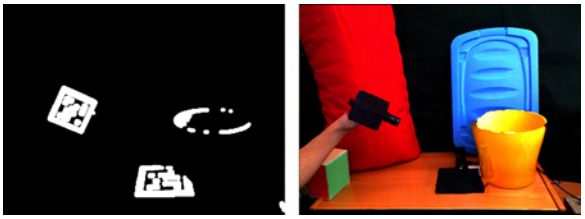


Figure 3: The left image shows the highlight mask. In a second step the masked pixels are filled as shown in the image on the right.

5.2 Diffuse Reflectance Estimation

After highlight removal we estimate the diffuse parameters per pixel by rephrasing the diffuse illumination equation. We end up with the following formula for the diffuse reflectance estimation k_d:

$$k_d = \frac{I}{\sum_{l=1}^{n} I_l (N \cdot L_l)}, \tag{9}$$

where I is the input intensity of the current pixel, I_l the intensity of the lth light source, L_l the direction towards the lth light source and n is the number of lights that are used for the BRDF estimation. Zheng et al. [25] estimate the diffuse parameters at a later stage because they used multiple RGB samples per vertex. We use the resulting buffer as diffuse reflectance map and as input for the clustering. The estimated diffuse reflectance map is shown in Figure 4. In the ideal case the different objects would have completely flat colors. However, this is not the case due to several simplifications that introduce consecutive errors in the pipeline. First, the environmental light is represented by only a few virtual point lights. Second, no shadows or indirect illumination are taken into account and third, the normal estimation is not absolutely correct.

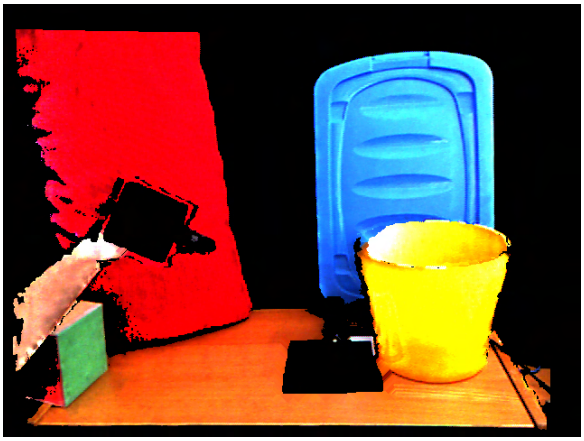


Figure 4: This image shows the estimated diffuse material component k_d. In the ideal case the objects would look perfectly flat and no variations due to different normals and thus illumination would be visible.

5.3 Clustering

Pixels with similar RGB colors in the diffuse reflectance map are assumed to have the same material and therefore need to be clustered. A novel K-Means implementation that is executed on the GPU performs the clustering. K-Means was introduced by Stuart P. Lloyd [9] and it consists of the following steps:

1. Randomly choose k cluster centers.
2. Assign each data element to the nearest cluster center using Euclidean distance.
3. Calculate new cluster centers by calculating the centroid over all data elements assigned to a specific cluster.
4. Repeat steps 2 & 3 until termination criteria are met.

Step 1: Initialize cluster centers: The resulting clusters heavily depend on the initial values chosen for the cluster centers. Thus if bad initial cluster centers are chosen, it might take many iterations until convergence. For each frame, we therefore use one to two different initial cluster centers. The first set uses the cluster centers from the previous frame and if the stopping criteria are met (see step 4) the next iteration is not executed anymore. However, if they are not met, the second set is executed with random cluster center values.

Step 2: Assign element to nearest cluster: Step two is adapted slightly so that step 3 can be executed on the GPU. Instead of just outputting the nearest cluster id and the minimum distance, we need to render each color pixel into multiple render targets. The idea is that each cluster has its own render target and pixels rendered into a given render target only belong to a certain cluster. We used eight simultaneous render targets and can handle six clusters each time a screen space pass gets executed. The following information is stored on a per-cluster basis for each pixel:

- The *RGB* color value
- The minimum distance to the nearest cluster center
- A binary flag that defines to which cluster the pixel belongs

The *RGB* color and minimum distance can be stored in one texture buffer with four floating point values. For the binary flags of all six clusters, we used two textures where every cluster gets assigned to one color channel. Depending on the cluster id assigned to a given pixel color, the color and distance information is only written into the textures assigned to the specific cluster. All other render target values are set to zero.

Step 3: Calculate new cluster centers: In step three we need to calculate the average *RGB* value for each cluster which is then used as a new cluster center. For a texture T with a size of $2^n \times 2^n$, there are n mipmap levels that can be created. The smallest mimap level with a size of 1×1 stores the average value of all data in texture T. However, we only want the average *RGB* color of those pixels that belong to a given cluster and ignore those that were set to zero.The cluster center can therefore be calculated using a combination of the two lowest mipmap levels from the color texture and the binary flag texture as follows:

$$cluster_c(T_{RGBD}, T^*) = \frac{avg(T_{RGBD})}{\sum_{i=0}^{n} \sum_{j=0}^{n} T^*_{i,j}} \tag{10}$$

where T_{RGBD} is a cluster specific texture containing the *RGB* color values and the distance value. T^* is the binary texture for the cluster having ones where pixels are assigned to that cluster and zeros otherwise.

Step 4: Repeat steps 2 & 3: In the original K-means method [9], the second and third steps are repeated until no data element changes the cluster anymore. This stopping criteria is too conservative for our needs. We need a fast clustering algorithm and thus have lowered the stopping criteria: First only a maximum number of 20 iterations are performed defining the upper bound for the computation time. Second if the variance change from one iteration to another drops below 10^{-4}, no further iterations are executed. By exploiting temporal coherence a low variance solution may be available after the first iteration and no new cluster center set needs to be processed. Note that the variance is calculated in a similar way to the cluster centers by exploiting mipmapping. As the squared distances for each pixel to the cluster centers are already calculated in the shader, the variance can be calculated nearly for free in step 2.

If the first run with the old cluster centers as initial values does not converge, the second run with random cluster centers gets executed. Then the cluster centers with the lower variance value are used for BRDF estimation. However, always using just the previous cluster centers could lead to a local minimum for clustering and there would be no way out to maybe find the global one. For this reason, in every fifth frame, the second iteration with random cluster centers will be executed anyway. Figure 5 shows the resulting clusters after K-Means is applied on the diffuse reflectance map.

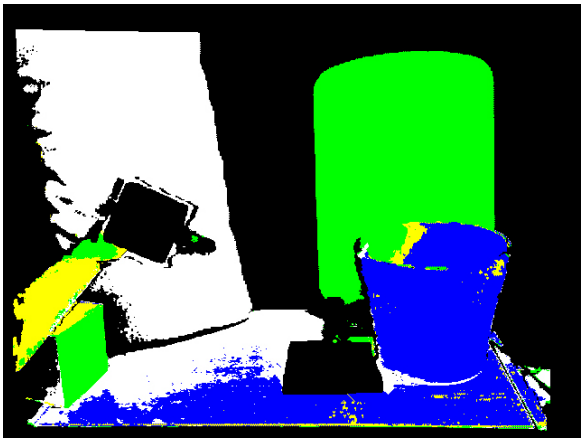


Figure 5: This figure shows the resulting clusters after K-Means is applied.

5.4 Specular Reflectance Estimation

One of the last steps needed in the BRDF estimation pipeline is the estimation of the specular intensity k_s and specular power n_s values per cluster. We assume white highlights and thus k_s is reduced to a scalar value. The paramters are estimated similar as proposed by Zheng et al. [25]. However, there are two main differences in our method. First, the solver works partly on the GPU and thus gains more speed than just a plain CPU implementation. Second, the positions of the light sources are not chosen to be fixed variables. The reason for this is that the positions are evaluated using importance sampling and thus can vary over time and furthermore need not to be at the exact position where a small light source is placed. However, the position of a light source highly influences the position of the specular reflection and therefore small variations of the initial positions are allowed to the solver.

For the non-linear optimization, a Nelder-Mead algorithm was used [12] with the following objective function evaluated on the GPU:

$$F_j = \sum_i \left[I_i - \sum_{l=1}^{n} I_l k_d (N \cdot L_l) + I_l k_s (V \cdot R_l)^{n_s} \right]^2 \quad (11)$$

where i iterates over all pixel intensities I_i which are related to cluster j. I_l is the intensity of the lth light source and k_d is the diffuse intensity vector of a cluster, which is set to the cluster center color. Note that for the specular component estimation, k_d is fixed and only the light source positions as well as k_s and n_s can be varied by the solver. N is the normal vector of the surface and R_l the reflection vector of the lth light source. V is a view vector pointing towards the camera. The result of the specular reflectance estimation is shown in Figure 6. Figure 7 shows a simple Phong-illuminated rendering on the left using the estimated k_d, k_s and n_s Phong illumination parameters. In this case, the same VPLs are used for illumination that are also used to estimate the BRDFs. In the image on the right side a rendering using DIR with an additional virtual pocket lamp is shown. Note the yellow indirect illumination on the real desk and on the virtual Buddha.

Figure 6: This image shows the result of the specular component estimation.

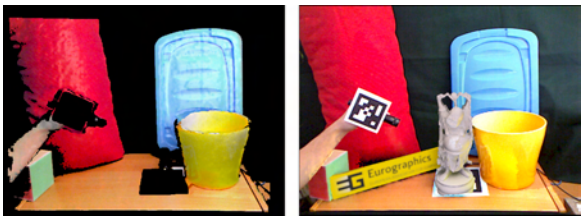


Figure 7: The left image shows a simple Phong rendering with the VPLs used for BRDF estimation. In the right image a virtual pocket lamp illuminates the real scene. Note the yellow color bleeding on the real desk and on the virtual Buddha.

6 RESULTS

Our computer system consists of an Intel Core 2 Quad CPU at 2.83 GHz and an NVIDIA GeForce GTX 580 with 1.5 GB of dedicated video memory. The software is running on Windows 7 64-bit and implemented in C# and DirectX 10. Cameras used in this scenario are the Microsoft Kinect sensor and an IDS uEye camera to capture the environment map.

6.1 Normal estimation

The two methods used for normal estimation differ in quality and computation time. Figure 8 shows a side-by-side comparison of the normal maps. The left normal map is calculated with the PCL library and a smoothing factor of 10. The average computation time is 223 milliseconds. The right normal map is computed with our proposed method in about 0.57 milliseconds. The PCL based normal map has a lot of holes, shown in grey color. In our method these holes are filled with the normals of the neighboring pixels. Even though these normals are not correct from a reconstruction point of view, they reduce the visible artifacts a lot. Furthermore note that our method produces sharper edges.

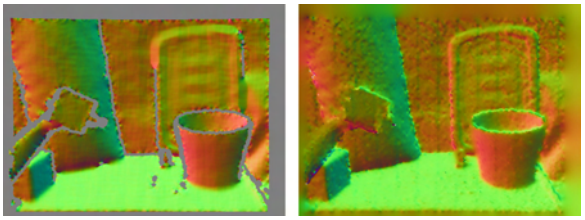


Figure 8: Comparison of the two implemented normal estimation methods. Left: Normals estimated using the PCL library [16] in 223 milliseconds. In grey areas no normal could be estimated by the PCL. Right: Our proposed method which takes 0.57 milliseconds.

6.2 K-Means clustering

We compared the proposed K-Means clustering implementation against the OpenCV library [3]. In the test setup a data set of 640×480 3-vector elements needed to be clustered. We ran both algorithms 5 times each time with different initial cluster centers. The interaction count of each run was set to 20. Note that no temporal coherence was exploited in order to get comparable results. The measured times include all the 5 runs and do not include the setup of the random data elements. Table 1 shows the execution times in seconds for 6 and 12 clusters.

Clusters	OpenCV	GPU K-Means
6	3.94s	0.33s
12	7.07s	0.44s

Table 1: Shows a comparison between the K-Means implementation from OpenCV [3] and our GPU implementation. Both algorithms ran 5 times with 20 iterations. Timings show the total execution of the 5 runs in seconds.

Table 2 shows the average iteration count needed for the first 50 frames to get below the variance change threshold. The columns show the average number of iterations for 6 clusters (6C) and for 12 clusters (12C). The rows show if the cluster centers from the previous frame were used (frame^{-1}) or if the cluster centers where chosen randomly (random). We set the maximum iteration count to 30, which was never reached during this test.

Initials	Avg. Iter., 6C	Avg. Iter., 12C
random	9.00	11.47
frame^{-1}	7.53	6.98

Table 2: Shows the average iteration count when reusing the cluster centers from the previous frame or taking random new ones.

6.3 Performance Analysis

The BRDF estimation pipeline has several steps. Table 3 gives an overview on the time spent for each one. In this setup, 6 clusters and 4 VPLs were used to approximate the BRDFs.

Stage	Time in ms
Normal Estimation	0.57ms
Highlight Removal	0.94ms
Diffuse estimation	0.23ms
K-Means	39.08ms
Specular estimation	315.76ms
Total time:	356.58ms

Table 3: Shows the time spent on each pipeline stage.

It clearly shows that the specular estimation step consumes by far most of the time. However, if it is possible not only to use a hybrid CPU / GPU version for the optimization but a complete GPU solution, the performance should increase a lot.

Two main parameters can be tweaked to get a better performance for a given scenario. One parameter is the number of materials that are estimated every frame. The

second parameter is the number of virtual point lights that are used to approximate the surrounding illumination. Table 4 shows the impact of different cluster and VPL settings with a fixed maximum iteration count for the specular estimation set to 50.

VPLs	6 Cluster (fps)	12 Cluster (fps)
1	3.82	2.41
8	2.23	1.30
128	1.70	1.01
256	0.99	0.59

Table 4: Shows the average fps with different VPL and cluster settings.

We also investigated how the maximum iteration count for the specular estimation solver reduces the total error. Interestingly, the change of the error was extremely small regardless how large the value was set. We think that this has to do with the large amount of pixels that are available in a cluster. Compared to that the area of a specular highlight is relatively small and thus correct estimations will only have a small impact on the total error.

Furthermore it turned out that the solver has difficulties in finding appropriate values in certain cases. Sometimes there is simply no highlight due to a given VPL. We therefore introduced a threshold value for a maximum error. If the error is too large, we set the specular intensity k_s to zero. Another problem could be that the solver just has one single point of view per frame whereas Zheng et al. [25] used several photographs to perform a specular estimation. Recently upcoming techniques, however, promise to greatly improve the problem of temporal coherent BRDF estimation (see Section 8).

6.4 Critical Scenario

A critical scenario is shown in Figure 9 on the left. It shows a blue notebook and a horse made from porcelain placed on a wooden box. The light is mainly coming from the direction of the spotlight partially visible on the left side of the image, causing specular highlights on the blue notebook envelope and the wooden box. The number of clusters used in this scenario is six, and four virtual point lights are used to estimate the surrounding illumination. The average frame rate is 2.3 fps.

In this scenario, the clustering is not as distinct regarding the objects compared to the first scenario. Due to the highlights caused by the spotlight, the clustering step creates different clusters for the same material as shown in Figure 9 (right). Furthermore, the Kinect sensor has troubles finding depth values at the white borders of the tracking marker, resulting in holes in the estimations (see Figure 10 (left)). Figure 11 shows the resulting diffuse (left) and specular (right) estimations. The Phong-shaded result is shown in Figure 12.

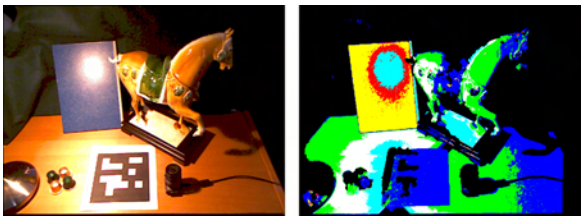


Figure 9: The left image shows the video input captured by the Kinect sensor. On the right side, the clustering result is shown.

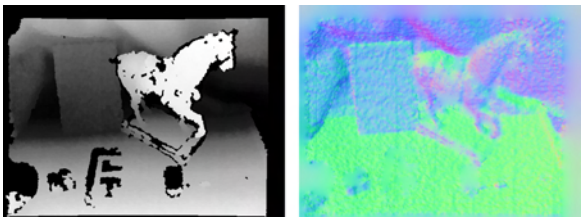


Figure 10: This figure shows the depth values acquired by the Kinect sensor on the left. Note that it failed to measure depth at the white borders of the tracking marker and the black fish-eye lense camera. On the right side the normal estimation from our proposed method is shown.

Figure 11: This figure shows the scene rendered with just the diffuse estimation (left) and specular estimation (right).

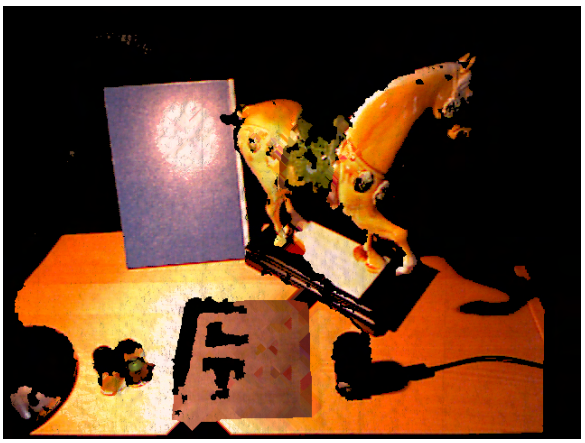


Figure 12: This figure shows the Phong-shaded result combining the diffuse and specular estimations.

7 LIMITATIONS AND DISCUSSION

Some limitations of our method are imposed by the Microsoft Kinect sensor, which is a structural light scanner. In general, depth values cannot be calculated when the light pattern is not recognized by the system. This

happens when objects are very glossy, reflective, transparent or absorb too much of the infrared light. The infrared pattern also vanishes in direct sunlight, making the approach unsuitable for outdoor mixed-reality. Furthermore, the border of curved objects is also often missing from the depth map because the projected light pattern is too distorted there.

Since bright pixels are assumed to be highlights due to specular reflections, bright and weakly saturated objects may be misinterpreted as highlights. Furthermore, shadows are not considered directly in the current implementation. Pixels with a brightness value below a certain threshold are simply discarded.

The K-Means clustering approach uses a variance value to decide whether further iterations are needed or not. However, there is no estimation of the optimal amount of clusters right now. This number must be specified by the user in advance and highly depends on the materials available in the scene.

Although temporal coherence is exploited at several stages in the pipeline, we do not continuously integrate already-seen geometry data. This would be helpful as a given point in the scene could be viewed under different viewing angles, leading to a better BRDF estimation, but could also lead to problems with moving objects.

Due to the real-time constraints several simplifications are introduced. The environmental illumination is approximated using a few virtual point lights, the normals have a lower quality compared to the PCL library and the clustering therefore also introduces some errors. All these simplifications lower the quality of the final BRDF estimation. However, since DIR mainly tries to compute visually plausible results rather than being physically correct, the estimated BRDFs should have a sufficient quality for mixed-reality scenarios.

8 CONCLUSION AND FUTURE WORK

We introduced a method to estimate the BRDFs of an augmented scene at interactive frame rates. The method does not need any precomputation, which makes it suitable for mixed-reality applications. The Microsoft Kinect sensor serves as a data input source to reconstruct the surrounding environment in the form of geometry and material properties. First, normals are estimated using a screen-space method exploiting temporal coherence. In the next pipeline stage we propose an adapted K-Means implementation that is specially tailored towards BRDF estimation and fast execution on the GPU. Temporal coherence is exploited here too, which allows us to find clusters faster than with a conventional implementation. The Phong parameter estimation is performed using a hybrid CPU / GPU variation of the Nelder-Mead optimization algorithm. The results demonstrate the feasibility of this method for mixed-reality applications.

In the future, we plan to enhance the quality and speed of this BRDF estimation method. It should be possible to gain a lot of speed by porting the Nelder-Mead optimization method to the GPU. Furthermore, recent techniques like KinectFusion [6] could greatly enhance the quality of the BRDF estimation.

9 ACKNOWLEDGEMENTS

This work was supported by a grant from the FFG-Austrian Research Promotion Agency under the program “FIT-IT Visual Computing” (project no. 820916). Studierstube Tracker is kindly provided by Imagination Computer Services GmbH.

10 REFERENCES

[1] Ronald T. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, August 1997.

[2] Samuel Boivin and Andre Gagalowicz. Image-based rendering of diffuse, specular and glossy surfaces from a single image. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH ’01, pages 107–116, New York, NY, USA, 2001. ACM.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb’s Journal of Software Tools*, 2000.

[4] Balaji Dhanasekaran and Norman Rubin. A new method for gpu based irregular reductions and its application to k-means clustering. In *Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units*, GPGPU-4, pages 2:1–2:8, New York, NY, USA, 2011. ACM.

[5] Bai Hong-tao, He Li-li, Ouyang Dan-tong, Li Zhan-shan, and Li He. K-means on commodity gpus with cuda. In *Proceedings of the 2009 WRI World Congress on Computer Science and Information Engineering - Volume 03*, CSIE ’09, pages 651–655, Washington, DC, USA, 2009. IEEE Computer Society.

[6] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST ’11, pages 559–568, New York, NY, USA, 2011. ACM.

[7] Martin Knecht, Christoph Traxler, Oliver Mattausch, Werner Purgathofer, and Michael Wimmer. Differential instant radiosity for mixed reality. In *Proceedings of the 9th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '10, pages 99–108, 2010.

[8] You Li, Kaiyong Zhao, Xiaowen Chu, and Jiming Liu. Speeding up k-means algorithm by gpus. In *Proceedings of the 2010 10th IEEE International Conference on Computer and Information Technology*, CIT '10, pages 115–122, Washington, DC, USA, 2010. IEEE Computer Society.

[9] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982.

[10] Bruno Mercier, Daniel Meneveaux, and Alain Fournier. A framework for automatically recovering object shape, reflectance and light sources from calibrated images. *International Journal of Computer Vision*, 73:77–93, June 2007.

[11] Microsoft. Kinect sdk: research.microsoft.com/en-us/um/redmond/projects/kinectsdk/.

[12] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[13] F Ortiz and F Torres. Automatic detection and elimination of specular reflectance in color images by means of ms diagram and vector connected filters. *IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews*, 36(5):681–687, 2006.

[14] Saulo A. Pessoa, Guilherme de S. Moura, Veronica Teichrieb, and Judith Kelner. Photorealistic rendering for augmented reality: A global illumination and brdf solution. In *Virtual Reality*, pages 3–10, 2010.

[15] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18:311–317, June 1975.

[16] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[17] Yoichi Sato, Mark D. Wheeler, and Katsushi Ikeuchi. Object shape and reflectance modeling from observation. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 379–387, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[18] Daniel Scherzer, Stefan Jeschke, and Michael Wimmer. Pixel-correct shadow maps with temporal reprojection and shadow test confidence. In Jan Kautz and Sumanta Pattanaik, editors, *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)*, pages 45–50. Eurographics, Eurographics Association, June 2007.

[19] Richard Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Underst.*, 58:23–32, July 1993.

[20] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '92, pages 265–272, New York, NY, USA, 1992. ACM.

[21] Ren Wu, Bin Zhang, and Meichun Hsu. Clustering billions of data points using gpus. In *Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*, UCHPC-MAW '09, pages 1–6, New York, NY, USA, 2009. ACM.

[22] S. Xu and A. M. Wallace. Recovering surface reflectance and multiple light locations and intensities from image data. *Pattern Recogn. Lett.*, 29:1639–1647, August 2008.

[23] Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: recovering reflectance models of real scenes from photographs. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 215–224, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[24] M. Zechner and M. Granitzer. Accelerating k-means on the graphics processor via cuda. In *Intensive Applications and Services, 2009. INTENSIVE '09. First International Conference on*, pages 7 –15, april 2009.

[25] Zuoyong Zheng, Lizhuang Ma, Zhong Li, and Zhihua Chen. Reconstruction of shape and reflectance properties based on visual hull. In *Proceedings of the 2009 Computer Graphics International Conference*, CGI '09, pages 29–38, New York, NY, USA, 2009. ACM.

A Hierarchical Splitting Scheme to Reveal Insight into Highly Self-Occluded Integral Surfaces

Andrea Brambilla
University of Bergen
Bergen, Norway
andrea.brambilla@uib.no

Ivan Viola
University of Bergen
Bergen, Norway
ivan.viola@uib.no

Helwig Hauser
University of Bergen
Bergen, Norway
helwig.hauser@uib.no

Abstract

In flow visualization, integral surfaces are of particular interest for their ability to describe trajectories of massless particles. In areas of swirling motion, integral surfaces can become very complex and difficult to understand. Taking inspiration from traditional illustration techniques, such as cut-aways and exploded views, we propose a surface analysis tool based on surface splitting and focus+context visualization. Our surface splitting scheme is hierarchical and at every level of the hierarchy the best cut is chosen according to a surface complexity metric. In order to make the interpretation of the resulting pieces straightforward, cuts are always made along isocurves of specific flow attributes. Moreover, a degree of interest can be specified, so that the splitting procedure attempts to unveil the occluded interesting areas. Through practical examples, we show that our approach is able to overcome the lack of understanding originating from structural occlusion.

Keywords: flow visualization, illustrative visualization, occlusion management.

1 INTRODUCTION

Flow phenomena are present at very different scales in our world, and they influence many aspects of our daily life: winds and water currents determine weather and climate, the stream of air around vehicles affects their speed and stability, the flow of blood in our vessels is fundamental for our good health condition. Understanding their behaviour is therefore highly relevant in many fields, and several years of research in *flow visualization* have produced a wide set of tools to accomplish this difficult task [PVH+02].

Flow behaviour can be analyzed from different points of view, according to the specific needs of the user. In particular, field experts are often interested in the trajectories of massless particles that are advected by the flow, which are commonly visualized using *integral curves*. Specifically, a *path line* represents the trajectory of a massless particle seeded from a specific starting location. Similarly, a *path surface* conveys the trajectories of a set of particles seeded along a $1D$ curve.

Integral surfaces are very expressive, but have a major downside: in correspondence with areas of swirling motion, like vortices and eddies, they tend to fold and twist, becoming very intricate and difficult to understand (Figures 2, 7, and 8). In this paper, we present a procedure which aims at solving this issue using techniques from traditional handcrafted illustration, such as cutting and splitting (Figure 1). These concepts have been frequently applied in medical visualization scenarios, but their application in the context of flow visualization has been limited. This is probably due to the fact that identifying well defined objects in flow data is very challenging. An overview of related approaches is presented in Section 2.

We propose a general surface splitting methodology based on two main concepts: a *cut space* defines possible ways to split a surface so that the resulting pieces have a clear meaning, while a *complexity measure* de-

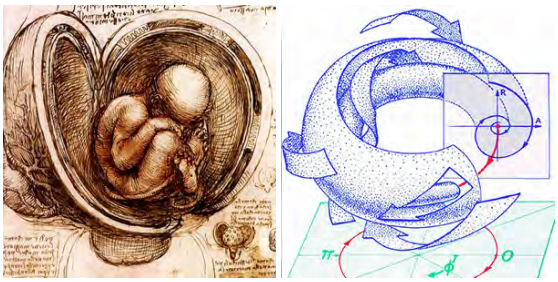


Figure 1: (left) Example of a cut-away view in a traditional illustration by Leonardo da Vinci [dV11]. (right) Illustration of a stream surface with cuts and clipping planes, by Abraham and Shaw [AS82].

termines a degree of occlusion at every point on the surface. We iteratively split the surface according to a cut from the cut space, so that the complexity is reduced the most. To improve the versatility of our approach, we allow the user to specify a degree of interest (DoI) function over the surface, which is combined with the complexity measure when the cut is chosen. Details on the splitting algorithm can be found in Section 3.

The resulting pieces of the surface are presented in a tree-like structure, and pieces of interest can be visualized either separated from the rest of the flow structure, or with a semi-transparent context (Figure 2). We use a stream surface extracted from the ABC flow to illustrate our method. We then show the application of our method on two datasets from application fields. Section 4 describes this process and provides a short discussion on timings and computational complexity.

Compared to the current state of the art, the main contributions of our work are:

- a general methodology for the design of surface cuts
- the first (to the best of our knowledge) splitting approach for integral surfaces
- a novel complexity measure for surfaces, which can take into account the importance of the data
- a helpful tool for the analysis of stream surfaces.

2 RELATED WORK

According to one of the most well-known categorizations [PVH+02], flow visualization techniques can be classified in four groups: direct, texture-based, geometric and feature-based visualization. Our work is related to the third category. Geometric approaches in fact aim at visualizing flow data through *integral structures*. The most common types of $1D$ integral curves are

- *streamlines*: curves tangent to the flow field in every point at a specific time instant
- *path lines*: the trajectories of massless particles in steady or unsteady flows
- *streak lines*: formed by particles continuously released in the velocity field from a specific location
- *time lines*: curves connecting a set of particles simultaneously released along a seeding curve.

These concepts can be extended to $2D$ and $3D$, obtaining surfaces and volumes respectively. Interested readers can refer to to the excellent survey by McLoughlin et al. [MLP+10] for more details.

Flow datasets are often multidimensional, multivariate and very dense. In these cases, traditional flow visualization approaches often suffer from cluttering and occlusion problems, which are commonly addressed with simple techniques, such as clipping, slicing or conventional transparency. A novel visualization research direction, called *illustrative visualization* [RBGV08], aims at solving these perceptual issues taking inspiration from traditional handcrafted illustrations.

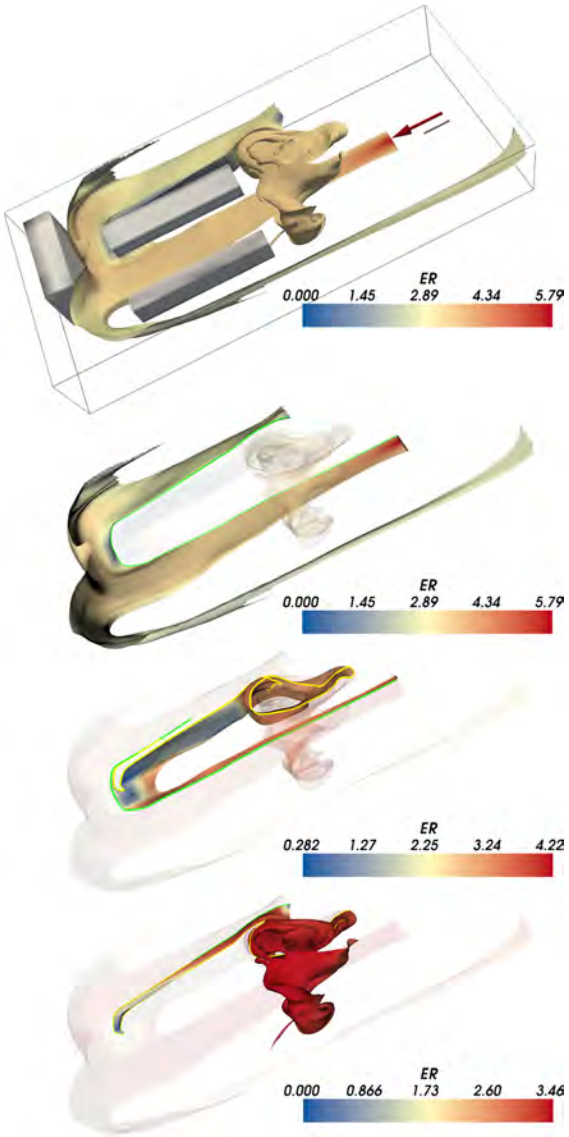


Figure 2: A stream surface extracted from a simulation of a gas leak on an oil platform. Top image: the initial surface with the position of the leak (red arrow) and the objects placed in the room (gray structures). Bottom three images: the surface pieces obtained after two cuts.

Cutting an object to reveal its inner parts is a common approach in illustrative visualization, and it can be applied in different ways. A typical example are *exploded views*: Li et al. [LACS08] apply this concept to show how composite objects are built. Ruiz et al. [RVB+08] suggest to subdivide a volume into oriented slabs according to the amount of information conveyed. More recently, Karpenko et al. [KLMA10] propose an

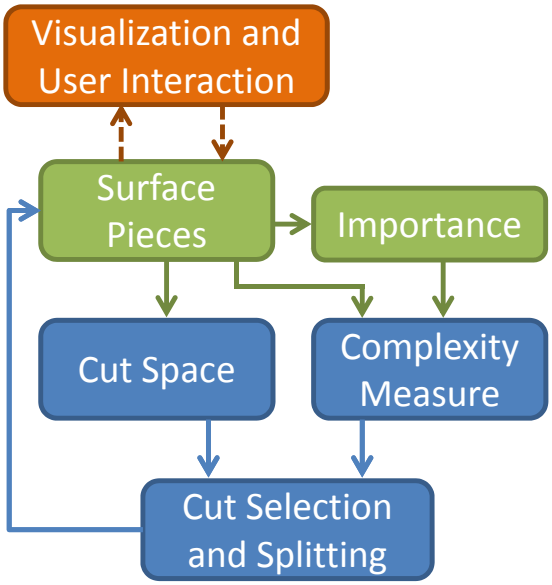


Figure 3: An overview of the splitting algorithm.

explosion strategy for mathematical surfaces based on surface symmetries.

If an importance measure is defined over the data, the visualization could be guided by these values. For instance, Viola et al. [VKG05] describe a volume rendering technique which discards the low-importance (context) portions of the volume occluding the relevant ones (focus). Similarly, Bruckner and Gröller [BG06] propose an exploded view strategy, where the occluding context is not discarded, but displaced in an intuitive way. Bruckner and Gröller also presented a concise overview of basic focus+context approaches in 2005 [BG05]. An effective combination of splitting and focus+context visualization has been presented by Balabanian et al. [BVG10]. Their work is focused on medical volumetric data and the splitting is based on a precomputed segmentation. The resulting pieces are displayed in a navigable graph, which was the main inspiration for our subdivision hierarchy.

Illustrative principles have been mainly adopted in medical visualization, but, especially in recent years, they are spreading to other contexts as well. For flow visualization, a fair number of illustrative techniques have been proposed [BCP+12]. The self-occlusion problem of integral surfaces have been initially addressed in an early paper by Löffelman et al. [LMGP97]: their approach cuts away pieces of the surface, generating results similar to the illustrations by Abraham and Shaw (Figure 1, right).

Two relevant focus+context approaches have been proposed in 2005 and 2007 respectively. The Eyelet particle tracing approach [WS05] shows integral surfaces passing through a specific point of high interest. In contrast, the technique by Correa et al. [CSC07] computes a deformation of the low importance data so that the focus is not occluded. More recently, two noteworthy approaches [HGH+10, BWF+10] propose to address the self-occlusion problem of stream surfaces through a smart use of transparency. They also adopt ad-hoc shading and texturing in order to improve depth perception and convey local flow behaviour.

Outside the context of flow visualization, similar issues have been investigated in connection with isosurfaces of scalar volumes. In this field, many techniques have been proposed (the contour spectrum [BPS97], Reeb graphs [FTAT00] and similarity maps [BM10], just to mention a few), but their applicability to flow data is still uncertain.

3 SURFACE SPLITTING

In the case of $3D$ flow fields, a stream surface is a $2D$ manifold. Our algorithm assumes it is represented by a triangular mesh. The mesh is defined by a set of points $P \subset \mathbb{R}^3$, and a set of triangles T. Flow data is sampled at each point in P: for instance, the velocity at a point $\mathbf{p} \in P$ is $\mathbf{v}(\mathbf{p})$. Linear interpolation is used to determine flow attributes over the triangles.

The structure of our general splitting framework is summarized in Figure 3. The splitting process is iterative and begins when the user requests to generate a cut. At this point two independent steps are performed: the complexity measure $cpx(\cdot)$ is computed for every $\mathbf{p} \in P$ and a set of potential cuts (the cut space) is generated. The complexity measure can take into account a degree of interes $doi(\cdot)$ defined over the points.

Notice that, regardless of how a cut is defined, it is always possible to reduce it to a *cutting curve* on the surface, i.e., the line along which a cut would split the surface. Therefore, for every potential cut, the complexity values are integrated along the corresponding cutting curve, and the cut with the highest overall complexity $CPX(\cdot)$ is chosen. The surface is finally split along the chosen cut, and the resulting pieces are inserted in the subdivision hierarchy (a binary tree) as children of the initial surface. The user can explore the tree and possibly request a new cut, executing again the whole procedure over all the leaves of the tree.

This is a general scheme to design effective splitting approaches, every step of the process can be customized according to the kind of surface of interest and to the desired results. In the following, we describe all the operations in detail and explain how we have tuned this framework in order to effectively split stream surfaces.

3.1 The Complexity Measure

The complexity measure $cpx(\cdot)$ is a function that associates a certain complexity value to every $\mathbf{p} \in P$. The meaning of this value depends on how the function is computed. Since our goal is to reduce occlusion, we define the complexity so that $cpx(\mathbf{p})$ represents how much $\mathbf{p}$ conceals the rest of the surface. However, to accurately evaluate such a measure, all the possible view-

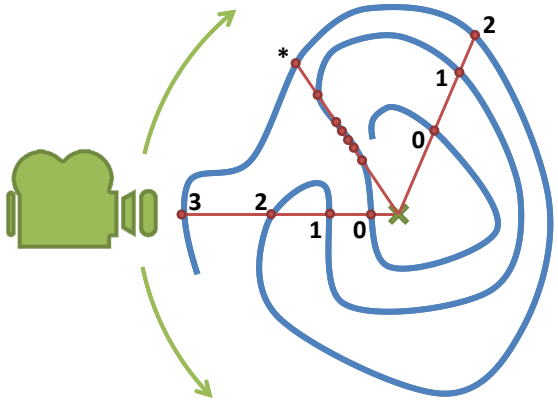


Figure 4: The typical visualization scenario. The camera (in green) moves circularly around the surface (in blue). The complexity measure, shown for a few points, is computed counting the intersections between the surface and the point-to-pivot line segment (in red).

points should be considered, which is too expensive to allow for user interaction. We opted for an approximation based on a simple consideration: datasets are frequently shown using a polar view, with the camera moving circularly around a pivoting point $\mathbf{o}$ placed at the center of the object of interest. Thus, we consider the amount of occlusion generated by $\mathbf{p}$ when the camera is looking directly at it, i.e., when it lies exactly between the camera and the pivot. Let $\mathbf{r} = \mathbf{o} - \mathbf{p}$ be the vector from $\mathbf{p}$ to $\mathbf{o}$, we set

$$cpx(\mathbf{p}) = \|X\| \tag{1}$$

where X is the set of intersection points between $\mathbf{r}$ and the surface mesh.

There is however an issue to solve: if $\mathbf{r}$ is tangent to portions of the surface, $cpx(\mathbf{p})$ can easily degenerate (Figure 4, middle red line). To attenuate this effect, we additionally take into account the angle between $\mathbf{r}$ and the surface normals $nrm(\cdot)$ at the intersection points

$$cpx(\mathbf{p}) = \sum_{\mathbf{x} \in X} \left| \mathbf{nrm}(\mathbf{x}) \cdot \frac{\mathbf{r}}{\|\mathbf{r}\|} \right| \tag{2}$$

Including the importance measure is straightforward. We have to modify the complexity function so that, if the occluded area is highly important, the complexity of the occluding points has to be high as well. We assume that the degree of interest function is a generic attribute $doi(\cdot)$ defined for every $\mathbf{p} \in P$:

$$cpx(\mathbf{p}) = \sum_{\mathbf{x} \in X} doi(\mathbf{x}) \left| \mathbf{nrm}(\mathbf{x}) \cdot \frac{\mathbf{r}}{\|\mathbf{r}\|} \right| \tag{3}$$

For the moment, we assume that $doi(\cdot)$ is defined at the beginning and never changes during the analysis phase; inclusion of interactive brushing techniques will be investigated in the future.

3.2 The Cut Space

The set of potential cuts can be defined in several ways. For example, Karpenko et al. [KLMA10] define it as a set of planes orthogonal to an explosion axis. Li et al. [LACS08], instead, define cuts as the boundaries of the components of the initial object. The fundamental requirement is that the elements of the cut space split the surface in meaningful and easily understandable pieces. In the case of flow data, defining such a space is not trivial: arbitrary cuts with a fixed geometry, such as planes or cubes, can reduce cluttering but the resulting pieces would be of difficult interpretation. Moreover, integral surfaces are not aggregate objects, so their building blocks cannot be easily defined.

One of the main characteristics of stream surfaces is that they have a semantically meaningful parametrization: every point on the surface lies in fact on the trajectory of one of the advected particles. Therefore, every point $\mathbf{p}$ can be associated with two parameters

- the *seeding point* $s(\mathbf{p})$: the location where the related particle has been seeded, expressed as a percentage of the length of the seeding line
- the *integration time* $t(\mathbf{p})$: the time needed by the related particle to travel from the seeding point to $\mathbf{p}$.

The isocurves of these two attributes are actually streamlines and time lines respectively. When a stream surface is split along one of these curves, the resulting pieces are stream surfaces as well. Therefore we define the cut space as the set of streamlines and time lines, corresponding to regular samples of their value ranges.

Notice that $s(\cdot)$ and $t(\cdot)$ are bijections. Therefore, in parameter coordinates, the surface is simply a portion of the $2D$ space, and the cuts become straight line segments parallel to the axis (Figure 5, left).

To improve the versatility of our system, we also provide the possibility of considering isocurves of arbitrary parameters. An example is shown in Figure 5, right, where the integration time has been replaced by the integration length, i.e., the arc length of the trajectory.

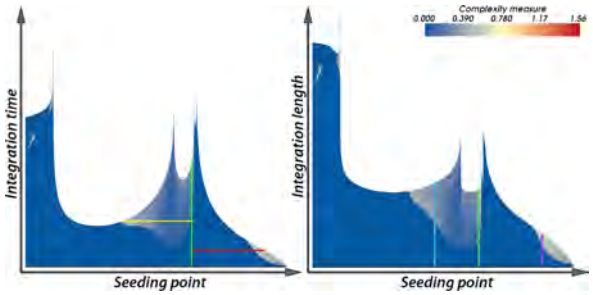


Figure 5: A stream surface from the ABC flow shown in parameter space, with three cuts. (left) parametrization given by the seeding point and the integration time. (right) The integration distance is used instead of the integration time.

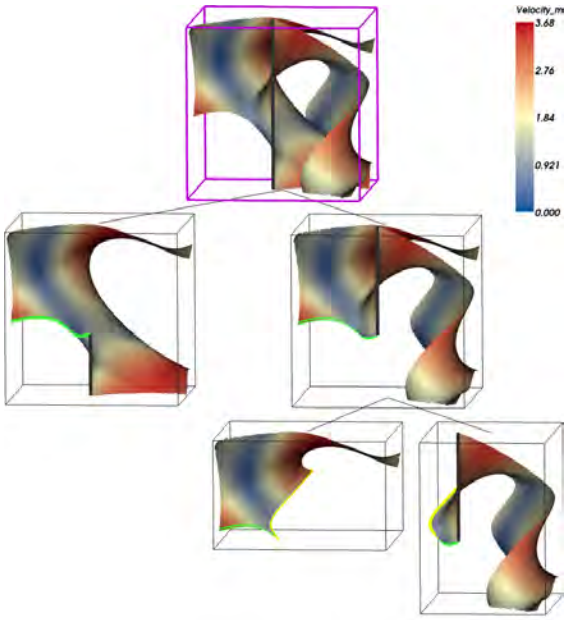


Figure 6: The tree obtained cutting two times a stream surface from the ABC flow. The first cut is made along a streamline (in green) and the second one along a time line (in yellow).

3.3 Surface Cutting

Given the space of potential cuts, we have to determine which cut would result in the most effective reduction of structural occlusion. Recall that the complexity measure has been already evaluated for every point on the surface. Then, we define the overall complexity $CPX(\cdot)$ of a cut Ω as the average complexity along it:

$$CPX(\Omega) = \frac{1}{length(\Omega)} \int_{\mathbf{x}\in\Omega} cpx(\mathbf{x}) \qquad (4)$$

An approximation of this integral is computed in the $2D$ parameter space as explained in Section 5.

The final step consists in selecting the cut with the highest overall complexity and using it to split the surface. However, the proposed complexity measure does not take into account the size of the resulting pieces. Usually, removing a relatively small piece from a large surface does not lead to a significant occlusion reduction. Therefore, we bias the cut selection in two ways: firstly we discard cuts that are shorter than a specified threshold. Then we adjust the complexity of the cuts according to the area ratio of the resulting pieces.

After the optimal cut is selected, the stream surface is split and the resulting pieces are inserted in the subdivision hierarchy as children of the split surface. We never had to modify the mesh structure to get well defined cuts, but, for low resolution models, a triangle splitting procedure may be required.

Notice that, if the surface has already been subdivided, the cut evaluation is performed on all the current pieces. Then, only the piece with the highest complexity cut is split.

The subdivision hierarchy is presented to the user as in Figure 6. At every node of the tree, the corresponding surface piece is displayed. The user can interact with this view to get an overall idea of the generated cuts. Then a single piece can be selected and visualized in a separate view in a focus+context manner: the piece of interest is rendered completely opaque while the rest of the surface can be optionally shown with variable transparency, as in Figure 7, bottom row.

4 DEMONSTRATION

In order to show the capabilities of our visualization system, we used it to explore stream surfaces extracted from one synthetic and two CFD datasets. In the following, we give details about the considered datasets and discuss the most relevant results.

4.1 ABC flow

The *ABC flow* is a synthetic dataset well known in flow visualization [DFH+86]. It is defined as a vector field over the domain $[0,2\pi]^3 \in \mathbb{R}^3$ and the velocities are given by:

$$\mathbf{v}(x,y,z) = \begin{pmatrix} A\sin(z)+B\cos(y) \\ B\sin(x)+C\cos(z) \\ C\sin(y)+A\cos(x) \end{pmatrix} \qquad (5)$$

which are solutions of the Euler equation for inviscid flow. We set $A = sqrt(3)$, $B = sqrt(2)$, and $C = 1$. An

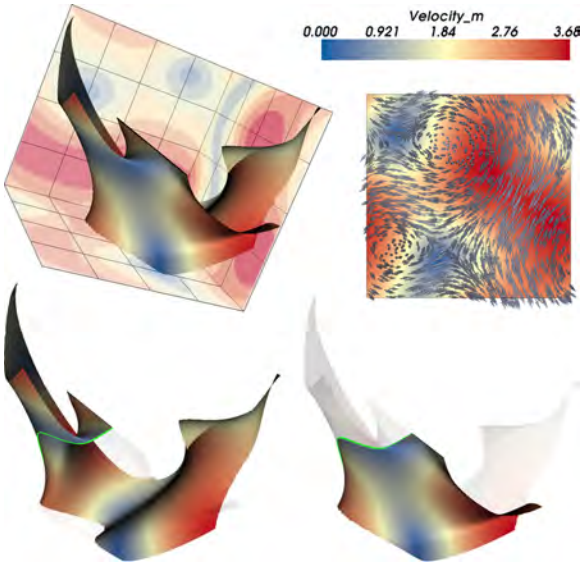


Figure 7: (top left) Overview of the ABC flow dataset, with a stream surface we extracted. (top right) A slice from the ABC flow where the velocity is depicted with glyphs. (bottom) The two pieces obtained by cutting the surface once, using the magnitude of the velocity as DoI. The complementary pieces of surface are shown semi-transparent to provide the context.

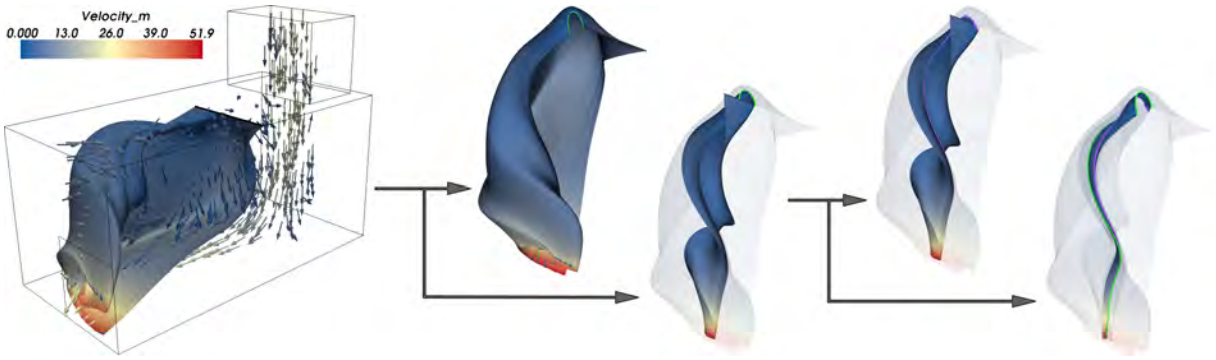


Figure 8: CFD simulation of a fluid flow in a box. The leftmost picture gives an overview of the dataset with the extracted stream surface. The other pictures show the surface after the first and the second cut.

overview of the dataset is given in Figure 7: the top left picture shows the boundaries of the domain and one expressive stream surface we extracted; the top right picture depicts the flow behaviour on the $z = \pi$ plane.

The stream surface under consideration has two almost overlapping areas in the bottom part, one on the left and one on the right. If we do not take into account any DoI, we expect that the splitting procedure separates these areas of the surface. That is exactly what happens after the first cut in Figure 6. The situation is even more interesting if we set the DoI proportional to the velocity magnitude: as can be seen in Figure 7, bottom row, the first cut is made so that the high velocity areas at the bottom right are clearly visible.

4.2 Flow in a box

The second dataset we investigated using our framework is a CFD simulation of fluid flow in a box-like structure. As illustrated in Figure 8, left, the inlet is placed on the far upper side, while the outlet is situated on the front plane, adjacent to both the right and the bottom wall. Vortices and eddies are expected close to where the inlet connects to the box, so we seeded a stream surface in that area.

The surface adequately conveys the rotational behaviour, but, due to self occlusion, it is very difficult to understand what is actually happening in the inner part. After applying a first cut, the more stable piece of the surface is separated from the swirling one, effectively showing the inner vortex (Figure 8, second and third pictures from the left). Requesting an additional cut, the twisting piece is split again (Figure 8, fourth and fifth pictures). This exposes the inner part of the surface and let us analyze the swirling behaviour close to the core of the vortex. Achieving the same goals with traditional techniques, such as transparency or clipping, would have been substantially more difficult.

4.3 Gas leak simulation

The last dataset is a CFD simulation of a gas leak in a closed room on an oil platform. An overview of the architectural structure is given in Figure 2, top. The left and right walls are semi-permeable and, in normal condition, there is an almost constant flow of air in the room, from right to left. After the gas begins leaking, it mixes with air and affects the regular air flow.

The gas/air mixture is described by the *equivalence ratio* (ER), which roughly represents the ratio between fuel and oxidant. In our scenario, where ER is between 0.5 and 1.7 the mixture is flammable, while ER greater than 1.7 means that the mixture cannot burn but it is not breathable either. One of the aspects of interest in this dataset is identifying the locations where there is mixing between air and gas.

We seeded a stream surface in front of the gas leak and observed its behaviour. Two vortices can be easily identified in the top part of the spatial domain and, given their proximity to the leak, they may have a strong influence on the mixing process. Our splitting approach, already at the first cut, correctly separates the branch with the two vortices from the rest of the surface (Figure 2). Figure 9 shows the effect of subsequent cuts: the swirling areas of the surface are effectively

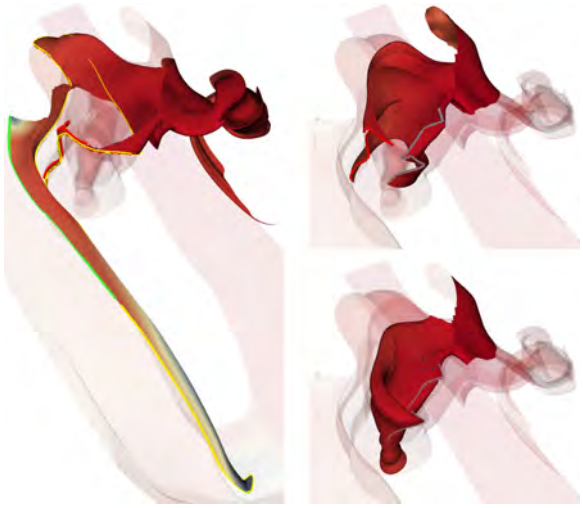


Figure 9: Pieces of stream surface extracted from the gas leak dataset. Iteratively cutting the surface with the proposed approach allows for an easy investigation of the inner areas of the vortices.

Dataset	Vertices	Triangles	Complexity Measure	Best Cut Search	Splitting
ABC flow	42 050	82 329	0.379 s	0.278 s	0.094 s
Box	166 499	322 931	1.466 s	0.582 s	0.362 s
Gas leak	151 320	286 874	1.438 s	0.475 s	0.301 s

Table 1: Summary of the execution time of every step of the pipeline.

subdivided, and the resulting pieces can be more easily investigated and analyzed.

We received positive feedback from a domain expert. Our splitting scheme is deemed effective in simplifying stream surfaces, easing the analysis phase. The approach is considered well suited for the validation of dispersion models and, in general, for the study of turbulence and small scale phenomena.

5 IMPLEMENTATION

The splitting algorithm can be briefly summarized as follows: when a cut is requested, for every current piece of the surface the complexity is computed, the cut space is generated, the best cut is identified and finally the corresponding piece is split. Notice that for every piece, the complexity, the cut space and the best cut can be stored and reused when another cut is requested. In order to maximize the efficiency of our system, the current implementation precomputes all these values for the existing pieces. Therefore, when a cut is requested, the previously computed best cut is used to split the corresponding piece of surface, then the two resulting pieces are analyzed and the next best cut is determined.

If the mesh used to represent the stream surface has a large number of vertices and triangles, determining the best cut can take a considerable time. We aim at supporting user interaction on, at least, surfaces of average size, thus, we introduced various optimizations. First of all, the computation of the complexity measure is based on a ray casting process in the three-dimensional space. This is known to be a highly expensive procedure. But we can exploit the fact that the rays we trace are always directed towards the pivot. We then compute the spherical coordinates (r, ϕ, θ) of every vertex with respect to the pivot: in the resulting spherical space, all the rays we need to trace are parallel to the r axis, which means we have one less dimension to take into account. Moreover, in this space we can use a simple quad-tree to speed up the process.

A similar idea is adopted to approximate the integration of complexity along the cuts. In the $2D$ parameter space, the surface is a flat plane and the cuts are straight lines parallel to the axis (see Section 3.2). Therefore we compute the parameter coordinates of the points and rasterize the transformed surface on a $n \times n$ grid. The parameter n is user specified and determines the size of the cut space. Every row and every column of the resulting image represents a possible cut: evaluating their overall complexity is now a simple image processing procedure.

The time needed to complete any of the steps of the pipeline is heavily dependent on the number of points and triangles of the mesh. This implies that, with the current implementation, the initial surface is the one that requires the most computational efforts to be analyzed. Table 1 summarizes the execution times of every step of the pipeline on the initial surface on a 2.8 GHz CPU. It is clear that the computation of the complexity measure is still the most expensive step despite the optimization. As a matter of fact, the complexity of a vertex is completely independent from the complexity of other vertices, so its computation can be easily performed on the GPU. This will be part of future developments.

6 CONCLUSION AND FUTURE WORK

We propose a novel illustrative flow visualization algorithm which can iteratively split an integral surface while preserving its semantic meaning. The subdivision effectively reduces the structural occlusion caused by the wrapping and twisting of the surface. The resulting pieces are presented in a focus+context fashion, and the relationships between different parts of the surface are conveyed through a subdivision hierarchy. We have applied our visualization system to study one synthetic dataset and two CFD simulations, obtaining meaningful results and receiving positive feedback from a domain expert.

We have already planned a series of changes which will improve different components of our framework. As mentioned in the previous section, we plan to rework the implementation, introducing additional optimizations and executing the parallelizable operations on the GPU. Regarding the visualization, many ideas are being evaluated: e.g., the subdivision tree can be modified in order to present both the hierarchical and the adjacency information between the surface pieces. Moreover, in the focus+context view, it can be useful to show a set of selected pieces instead of just one.

In this paper we have demonstrated our approach applied to stream surfaces, but its extension to path surfaces is straightforward. We believe that the general idea can be applied to many different kinds of surfaces once a suitable cut space has been determined.

ACKNOWLEDGEMENTS

Many thanks the anonymous reviewers for their feedback. We are grateful to Maik Schulze and Holger Theisel for providing the code for the generation of stream surfaces. Special thanks to Josué Quilliou and

GexCon AS for providing the gas leak dataset. Thanks also to AVL for provinding the dataset of the flow in a box. This report has been worked out within the scope of the SemSeg project and we acknowledge the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number 226042.

REFERENCES

[AS82] R. Abraham and C. D. Shaw. *Dynamics–the geometry of behavior*. Aerial Press, 1982.

[BCP+12] A. Brambilla, R. Carnecky, R. Peikert, I. Viola, and H. Hauser. Illustrative flow visualization: State of the art, trends and challenges. In *Eurographics 2012 State-of-the-Art Reports*, pages 75–94, 2012.

[BG05] S. Bruckner and M. E. Gröller. Volumeshop: an interactive system for direct volume illustration. In *IEEE Visualization 2005*, pages 671–678, 2005.

[BG06] S. Bruckner and M. E. Gröller. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1077–1084, 2006.

[BM10] S. Bruckner and T. Möller. Isosurface similarity maps. *Computer Graphics Forum*, 29(3):773–782, 2010.

[BPS97] C. L. Bajaj, V. Pascucci, and D. R. Schikore. The contour spectrum. In *IEEE Visualization '97*, pages 167–ff., 1997.

[BVG10] J.-P. Balabanian, I. Viola, and M. E. Gröller. Interactive illustrative visualization of hierarchical volume data. In *Proceedings of Graphics Interface 2010*, pages 137–144, 2010.

[BWF+10] S. Born, A. Wiebel, J. Friedrich, G. Scheuermann, and D. Bartz. Illustrative stream surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16:1329–1338, 2010.

[CSC07] C. D. Correa, D. Silver, and Min Chen. Illustrative deformation for data exploration. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1320–1327, 2007.

[DFH+86] T. Dombre, U. Frisch, M. Henon, J. M. Greene, and A. M. Soward. Chaotic streamlines in the abc flows. *Journal of Fluid Mechanics*, 167:353–391, 1986.

[dV11] L. da Vinci. The babe in the womb, c. 1511.

[FTAT00] I. Fujishiro, Y. Takeshima, T. Azuma, and S. Takahashi. Volume data mining using 3d field topology analysis. *IEEE Computer Graphics and Applications*, 20(5):46–51, 2000.

[HGH+10] M. Hummel, C. Garth, B. Hamann, H. Hagen, and K. I. Joy. Iris: Illustrative rendering for integral surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16:1319–1328, 2010.

[KLMA10] O. Karpenko, W. Li, N. Mitra, and M. Agrawala. Exploded view diagrams of mathematical surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1311–1318, 2010.

[LACS08] W. Li, M. Agrawala, B. Curless, and D. Salesin. Automated generation of interactive 3d exploded view diagrams. pages 101:1–101:7, 2008.

[LMGP97] H. Löffelmann, L. Mroz, M. E. Gröller, and W. Purgathofer. Stream arrows: Enhancing the use of streamsurfaces for the visualization of dynamical systems. *The Visual Computer*, 13:359–369, 1997.

[MLP+10] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen. Over two decades of integration-based, geometric flow visualization. *Computer Graphics Forum*, 29(6):1807–1829, 2010.

[PVH+02] F. H. Post, B. Vrolijk, H. Hauser, R. Laramee, and H. Doleisch. Feature extraction and visualization of flow fields. In *Eurographics 2002 State-of-the-Art Reports*, pages 69–100, 2002.

[RBGV08] P. Rautek, S. Bruckner, M. E. Gröller, and I. Viola. Illustrative visualization: new technology or useless tautology? *ACM SIGGRAPH Computer Graphics Quarterly*, 42:4:1–4:8, 2008.

[RVB+08] M. Ruiz, I. Viola, I. Boada, S. Bruckner, M. Feixas, and M. Sbert. Similarity-based exploded views. In *Smart Graphics*, volume 5166, pages 154–165. 2008.

[VKG05] I. Viola, A. Kanitsar, and M. E. Gröller. Importance-driven feature enhancement in volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):408–418, 2005.

[WS05] A. Wiebel and G. Scheuermann. Eyelet particle tracing - steady visualization of unsteady flow. In *IEEE Visualization 2005*, pages 607–614, 2005.

Morphometric Analysis of Mesh Asymmetry

Václav Krajíček[1,2] Ján Dupej[1] Jana Velemínská[2] Josef Pelikán[1]
vajicek@cgg.mff.cuni.cz jdupej@cgg.mff.cuni.cz velemins@natur.cuni.cz pepca@cgg.mff.cuni.cz

[1] Department of Software and Computer Science Education, Charles University in Prague, Malostranské náměstí 25, 11800, Prague, Czech Republic

[2] Department of Anthropology and Human Genetics, Faculty of Sciences, Charles University in Prague, Viničná 7, 12844, Prague, Czech Republic

ABSTRACT

New techniques of capturing shape geometry for the purpose of studying asymmetry in biological objects bring the need to develop new methods of analyzing such data. In this paper we propose a method of mesh asymmetry analysis and decomposition intended for use in geometric morphometry. In geometric morphometry the individual bilateral asymmetry is captured by aligning a specimen with its mirror image and analyzing the difference. This involves the construction of a dense correspondence mapping between the meshes. We tested our algorithm on real data consisting of a sample of 102 human faces as well as on artificially altered meshes to successfully prove its validity. The resulting algorithm is an important methodological improvement which has a potential to be widely used in a wide variety of morphological studies.

Keywords

mesh asymmetry, geometric morphometry, mesh correspondence

1. INTRODUCTION

The geometry of an object reflects many facts about its creation development and purpose. A significant property observed in many biological objects is symmetry. Specifically, bilateral symmetry can be defined as the existence of a plane that splits an object into two identical parts with respect to reflection. This kind of symmetry is exhibited by most living natural objects (see Figure 1). Deviation from bilateral symmetry, asymmetry, can result from various stresses in population or individual development. Evaluation of asymmetry in the human face is useful in various scientific fields like anthropology, plastic surgery, forensic medicine, orthodintics, surgery, anatomy and others. Notably, the quantitative analysis of asymmetry provides important information for treatment planning; e.g. it can be used to determine the target area or the surgical method to be applied [Dam11].

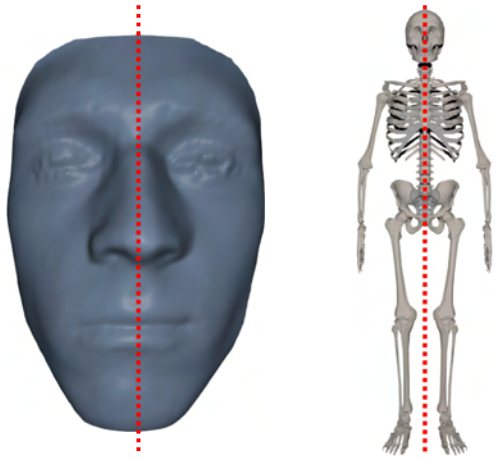


Figure 1: Examples of bilateral symmetry. It can be easily found in many higher and lower species.

Traditional and geometric morphometry, the tool of many fields of natural science such as botany, zoology or anthropology, offers ways to analyze the asymmetry of simple morphometric data such as lengths and landmark locations. However, with the advent of new measuring techniques such as surface and CT scanners to these fields new methods are required to analyze the full measure of information provided by these new data modalities.

Particularly we aim to develop an algorithm to analyze the asymmetry in triangular meshes in a way that is needed for geometric morphometry [Boo97],

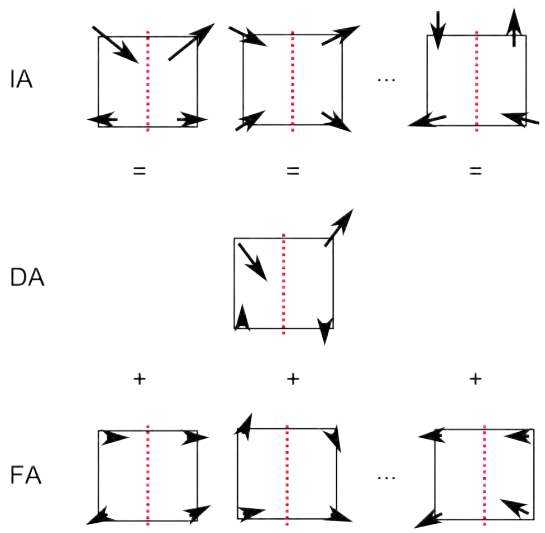


Figure 2: Decomposition of individual asymmetry (IA) into directional (DA) and fluctuating (FA) asymmetry components.

which includes its decomposition into directional and fluctuating asymmetry [Val62] [Pal94] [Kli02]. Before we analyze the group tendency to asymmetry we have to formulate how to interpret individual asymmetry.

Individual asymmetry is the quantified difference in a particular feature from its paired counterpart. Directional asymmetry is then the average of these differences across the studied sample. The statistical significance of this average can be evaluated using standard statistical tools such as the t-test. In other words, directional asymmetry describes the tendency to a certain deviation from the symmetry of the whole group or population. On the other hand, fluctuating asymmetry is defined as the difference between individual asymmetry and directional asymmetry and thus represents the random presence of asymmetry in the individual. Fluctuating asymmetry traits are normally distributed around the mid-sagittal plane. The overall magnitude of the fluctuating asymmetry is generally more significant than its spatial distribution. The process of such decomposition is visualized in the Figure 2.

In geometric morphometry, asymmetry is evaluated on paired features, i.e. paired landmarks and distances. The correspondence of particular features in traditional and geometric morphometry across the sample is known by definition from the homology of the features. Such correspondence is, however, not implicitly defined on triangular mesh data.

In the following chapter, some existing work related to mesh asymmetry analysis and extraction will be introduced. In Chapter 3 we will present the basis of our approach, namely the dense correspondence algorithm introduced by Hutton [Hut01]. Next, in Chapter 4 we thoroughly explain our procedure. We then we demonstrate its results in three scenarios in Chapter 5. Finally, Chapter 6 concludes our work and discusses future extensions.

2. RELATED WORK

Symmetry and asymmetry is an important feature of the human brain which was intensively studied in the past. In the modern era there have been many attempts [Fou11] to analyze MRI images and interpret brain asymmetry with respect to its connection to illnesses, functional abilities or genetics.

There have not been many attempts at automatic analysis of asymmetries in mesh data. One particular approach [Liu03] maps objects of interest onto a surrounding cylindrical surface. In the reference cylindrical coordinate system, the corresponding symmetric points are found with the help of manually placed landmarks. Asymmetry is then deduced from these pairs. This approach obviously works only for simple shapes that can be unambiguously projected onto a cylinder.

A different method [Ola07] assumes the existence of an ideally symmetric template and then maps each subject in the study onto this template using B-spline based non-rigid registration. Construction of the ideal or any symmetric template for a certain group is not trivial. Constructing that template requires the so called mid-sagittal plane about which the template is bilaterally symmetrical.

Another approach [Com08b] constructs a mid-sagittal plane using a modified EM algorithm [Com08a] and uses it to mirror the studied shapes. Asymmetry is then represented as the distance between the corresponding points on the original and mirror shape. Correspondence is found using non-rigid registration, bending the mirror shape to the original.

In geometric morphometry [Boo97], asymmetry was analyzed on landmark datasets [Sch06] by mirroring the landmark configuration and reordering the landmarks so that the mirror and the original can be realigned. The asymmetry is then defined as the difference of an ideally superimposed mirror and the original. More importantly, the approach decomposes the asymmetry in the traditional way studied in biological sciences [Val62] [Pal94].

We used some of these ideas in our proposed method.

3. DENSE CORRESPONDENCE

The individual asymmetry can be computed from the knowledge of matching counterpart features in a mirrored mesh. To calculate the directional and

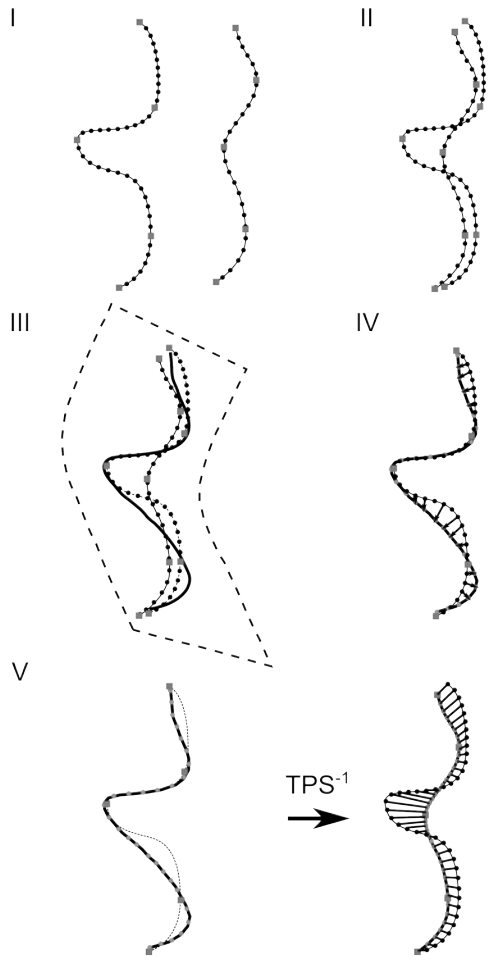


Figure 3: The steps of dense mesh correspondence construction in 2D. I-II) rigid alignment of input data, III) TPS deformation of the moving mesh, IV) nearest neighbor correspondence search, V) inverse TPS deformation of found points

fluctuating asymmetry, the matching of corresponding features in the group must be known.

Finding these matches equates to the construction of correspondence mapping either between a mesh and its mirror image of any two meshes in the group. Generally, any non-rigid mesh registration algorithm could be used for this task.

In the case of biological data we are also able to exploit homology – a unique one-to-one correspondence of certain features of interest. We therefore employ the following mesh correspondence construction.

The dense correspondence construction algorithm by [Hut01] [Hut03] uses sparsely landmarked surfaces. The more landmarks are used, the better the results. If possible, landmarks should be spread evenly over the area of interest. Placing all landmarks in one plane should be avoided as it will reduce the quality of correspondence. This is due to the fact that the spatial deformation describing the correspondences would

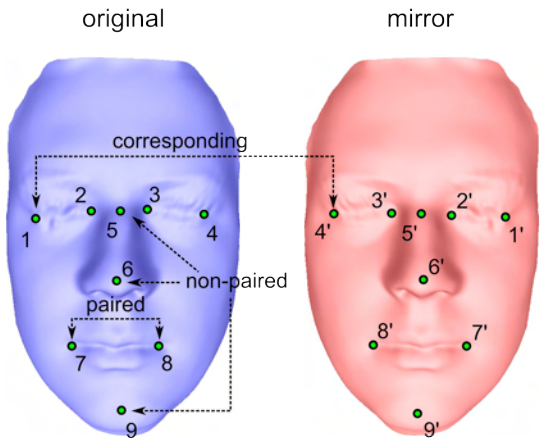


Figure 4: Original and mirror mesh.

not be defined well outside this plane. The mesh against which the correspondences are sought is designated the reference mesh, while the others are referred to as the moving meshes.

First, we rigidly align the moving mesh to the reference mesh by minimizing the squared distance of their respective landmarks using ordinary Procrustes superimposition, a method of rigid registration [Boo97] while preserving unit centroid size.

Second, the aligned moving mesh is bent to fit the reference mesh using thin-plate splines (TPS) interpolation

$$TPS(\vec{x}) = \vec{a}_0 + \vec{a}_1 x + \vec{a}_2 y + \vec{a}_3 z + \sum_{i=0}^{n} \vec{w}_i \varphi(\|\vec{x} - \vec{p}_i\|)$$

where $\vec{p}_i$ are landmark locations on the reference and φ is the radial basis function $\varphi(r) = r^3$. The TPS is fitted to the superimposed landmark pairs from the previous step. This way the meshes are brought as close as possible to each other so that the correspondences can be constructed.

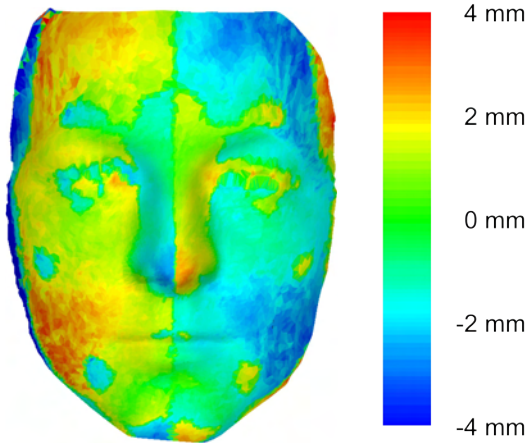


Figure 5: Symmetric face and individual asymmetry captured by the algorithm

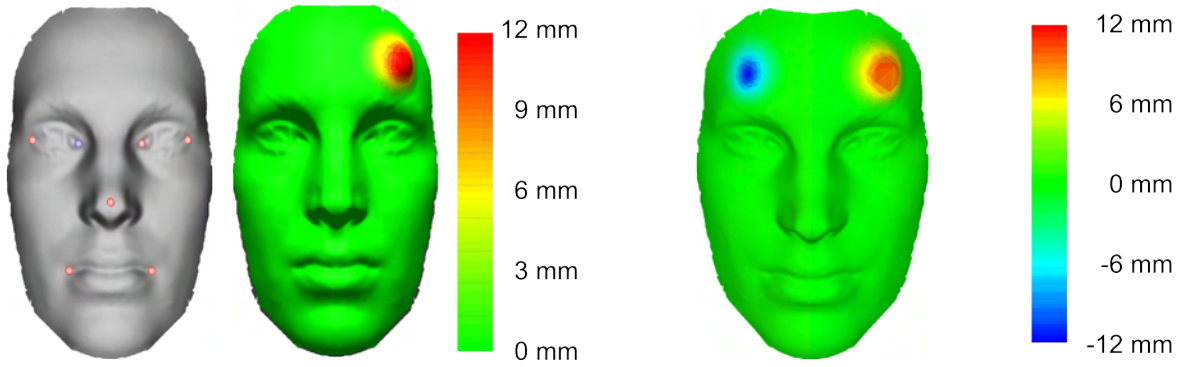


Figure 6: An ideally symmetric face and the artificially added asymmetry (left). Individual asymmetry of artificial test subject detected by the algorithm (right).

In the last step the reference mesh is used to construct a new mesh with the same topology as the reference but the shape of the moving mesh, which is referred to as the correspondence mesh. This is done by finding the closest point (not necessarily vertex) in the reference mesh to each vertex of the moving mesh. The search efficiency can be enhanced with acceleration structures. A kd-tree on all mesh triangles has proven very effective. This process yields the correspondence between the reference mesh's points and the deformed moving mesh. We now need to transform these vertex locations to the space of the original moving mesh. Because the deformed mesh was created with a TPS the original is obtained by inverting that TPS. Since TPS has no analytically defined inversion it must be computed numerically as the solution of the following minimization problem.

$$\underset{\vec{x}\in R^3}{\boldsymbol{arg\ min}} \|TPS(\vec{x})-\vec{y}\|^2$$

This could be solved by any optimization procedure. Because in this case the second derivatives can be easily analytically expressed, the Newton's method is suitable for the problem.

An alternative to the numeric TPS inversion computation is an approximation using barycentric coordinates. The correspondence points in the deformed moving mesh are expressed in barycentric coordinates inside their respective triangles. The same barycentric coordinates are then used to compute the point in the corresponding face of the original moving mesh. A scheme of the correspondence construction procedure is described in Figure 3.

Sometimes the corresponding point is located too far from the vertex in the reference. Then it is not likely that it is a proper correspondence. We can define a threshold distance beyond which the correspondence is not accepted. In that case we simply omit this vertex from the mesh with identical topology.

In order to analyze a group of meshes the correspondences across the whole sample must be constructed. One mesh from the sample is chosen as the reference and the remaining meshes are used as moving meshes to construct correspondences to the reference.

The choice of the reference mesh may have a significant influence on the result. It should be chosen so that it is not too different from the rest of the group. Ideally, it lies in the middle of the group, hence the other moving shapes need to be bent less to align with the reference. The authors of the original algorithm [Hut01] claim that if the input data is random, choosing the first mesh as the reference for the group is as good as choosing any other.

4. MESH ASYMMETRY ANALYSIS

In our approach to capture a group mesh asymmetry we use the concept of decomposing the asymmetry into directional and fluctuating components. Concurrently, dense triangular meshes allow us to express the asymmetry on a very localized level.

Before we can compare the local asymmetries in all meshes, we need to have them transformed into a common coordinate space. This is done by applying a group-wise rigid landmark-based registration, specifically generalized Procrustes superimposition has been used. The meshes' vertices are transformed the same way as the landmarks in the Procrustes superimposition.

The next step is to recompute the meshes to the same number of vertices and the same topology. We used the dense correspondence construction algorithm from the previous section.

Now, the individual mesh asymmetry is computed. The result is the list of directions for every vertex. If every vertex were moved by their respective displacement the mesh would become ideally symmetric. The mirror mesh must be constructed by

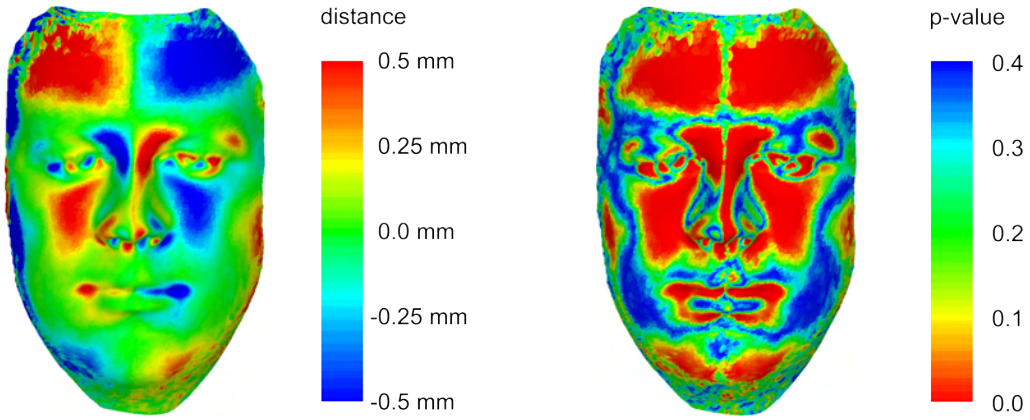


Figure 7: Directional asymmetry (left) and significance map of the asymmetry (right) on 102 human faces.

negating one of the vertex coordinates (see Figure 4). The same is done with landmarks placed on the mesh.

The landmarks are then used to align the mirror mesh back to the original mesh. In order to do that the landmarks must be reordered as they have changed their homologous meaning after the mirroring. e.g. some landmarks on the left side of a bilaterally symmetric mesh became the landmarks on the right side of the mirror mesh. These are the so called paired landmarks; they swap their positions with their mirror counterparts, while the others, the non-paired landmarks are not affected by mirroring.

After mirroring and landmark reordering the mirror meshes are realigned by ordinary Procrustes superimposition and deformed by TPS in order to get mirror meshes closer to the original ones and allow for subsequent correspondence searching. The closest points on the mirror meshes to each vertex of the original correspondence meshes are found using search acceleration structures. Again, we opted for a kd-tree.

The vectors defined by the difference of the original mesh vertices and their closest mirror mesh points is the local measures of asymmetry. Completely symmetrical shapes have identical mirrors and when aligned the distances between the original mesh vertices and the closest points are zero. If asymmetries occur on the mesh the difference between the left and right part of the mesh appears and the distance between a part of the mesh and its mirrored counterpart becomes non-zero. Furthermore, the associated vector holds the information about the direction of the asymmetry, i.e. how the part of the mesh was moved to form the asymmetry. This information is exhibited on either side of a bilaterally symmetrical mesh in the opposite directions. Hence it cannot be said which part of the mesh originated from a symmetric shape and which was altered, if this is the way the analyzed asymmetric shape was created. It can be said that bilateral asymmetry is a symmetric feature. The vector field that represents the displacement of a point on a mesh from where it would lie if the mesh were ideally symmetric is called individual asymmetry.

We visualize the aforementioned vector field on the original mesh with color-coded signed distances (see Figure 5). The sign is the same as that of the dot product of the mesh normal and the vector of individual asymmetry in that point. The color images could be simply interpreted in the following way: red areas lie in the front of the corresponding mirrored counterpart which means that they are larger than the corresponding paired counterpart while blue areas are smaller and lie behind the aligned mirrored counterpart. The areas that are close to green are not significantly larger or smaller. This interpretation does not include any information about the direction

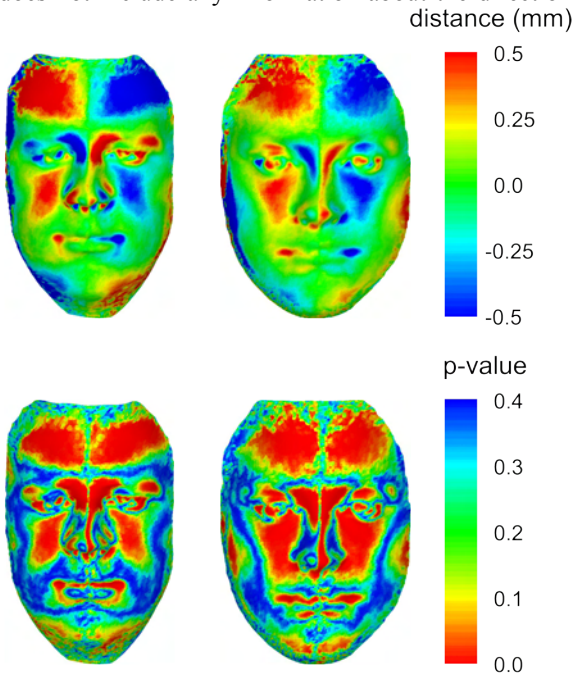


Figure 8: Directional asymmetry of male (left) and female (right) subgroups.

of the asymmetry. This sort of visualization is also known as clearance vector mapping and is useful in quantifying the facial surface asymmetries in the areas where anthropometric landmarks are scarce. The volume of detected asymmetry is potentially significant in patients who may have their unilateral facial deficiencies corrected using injections or implants [OGr99].

All individual asymmetries are already aligned group-wise, therefore, directional asymmetry is computed as the average of all corresponding individual asymmetry vectors. The length of the respective individual asymmetry vectors is the same for all the meshes and they correspond to each other per element as the meshes were reconstructed to have the same topology. The directional asymmetry is visualized the same way as individual asymmetry.

Fluctuating asymmetry is computed as the difference of individual asymmetry and directional asymmetry. Its visualization is also based on color coded distances without the consideration of the sign for direction as was done for individual and directional asymmetry above. As stated in the previous chapter, we are more interested in the overall magnitude of the fluctuating asymmetry which is computed as the sum of squared distances of the fluctuating asymmetry vectors. It can be compared across the group and if normalized by the number of vectors, or between groups just as well.

To prove that the directional asymmetry reflects the global trend of the group, and is not the result of randomness in the group, it has to be tested statistically. A standard t-test is performed on the signed lengths of corresponding individual asymmetry vectors. The significance map can then also be visualized. This way of interpretation is especially important for particular research in biological sciences.

The direction of the individual asymmetry vector is also important property that should be taken into account. In order to do so we define the local orientation difference asymmetry measure which is equal to cosine of angle between corresponding individual asymmetry vectors.

The lengths of individual asymmetry vectors and local orientation difference asymmetry measure can be summarized into total asymmetry and total orientation asymmetry.

$$TA = \frac{1}{N}\sum_{i=0}^{N} \left\| \vec{a}_i \right\|^2$$

$$TOA = \frac{1}{N}\sum_{i=0}^{N} \vec{n}_i \cdot \vec{m}_i$$

5. RESULTS

Our semiautomatic landmark-based approach reflects the needs of scientists from fields like archaeology and anthropology where the homology of certain feature points is advantageous. It is often the case that mesh datasets together with their respective landmark configurations already exist and the results of landmark-only based studies have already been published so some comparison of results can be done. All these facts are the motivation supporting our solution over completely autonomous algorithms, e.g. employing nonrigid mesh registration.

In the first test it will be demonstrated how the proposed algorithm approximates individual asymmetry on an artificial dataset where the ground truth is well known. A symmetric face model was created by cutting a real face in half approximately along its medial axis mirroring one of the halves and stitching it to its original. Then the symmetric face was locally deformed, bulged on the left side of the forehead. Shell distance measured by a software tool, RapidForm XOS in our case, was used as the ground truth (see the left side of Figure 6). In this case, if the deformation does not affect the areas with landmarks, the algorithm can uncover the asymmetry perfectly (see the right side of Figure 6). If the location of a landmark is distorted by a nearby asymmetry the error of alignment by the generalized Procrustes superimposition will be distributed among all

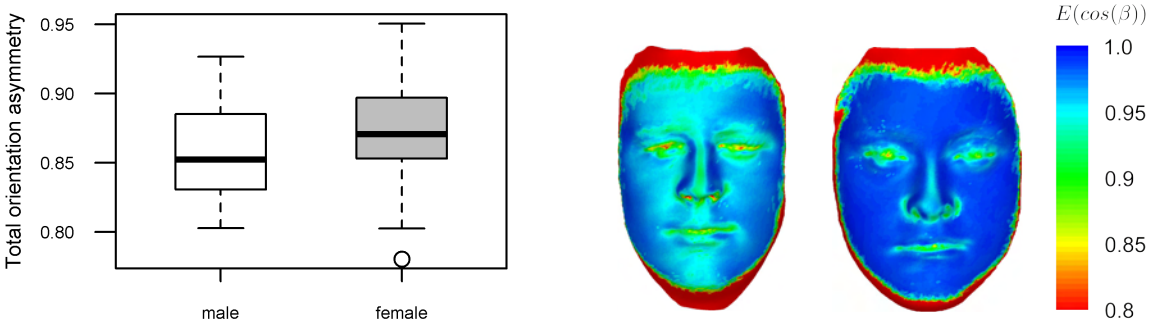


Figure 9: Difference in total orientation asymmetry of male and female subgroups is statistically significant ($p < 0.05$). The orientation the asymmetry is locally unevenly distributed on the male faces while being uniform almost everywhere on female faces.

landmarks. Even in this case the algorithm still reveals the asymmetry relatively well. In case of large asymmetries all across the mesh it would be difficult for any method to recover results close to ground truth as in this kind of problem it is highly ambiguous what the original symmetric shape is. Therefore in practice, landmarks are usually placed on stable locations. These locations may exhibit asymmetry as well, it is however assumed that the user will choose a landmark configuration whose own asymmetry will have the least impact on the results.

In the second test a group of 102 real faces were analyzed. The faces were captured by a *InSpeck 3D MegaCapturor II* scanner with 0.4 mm resolution in the direction of its optical axis. The resulting meshes with approximately 100k-200k triangles were cleaned and trimmed from border areas and finally decimated to approximately 20k-30k triangles. Our landmark configuration contains nine landmarks situated in the corners of mouth, eyes, and on the tip and around the nose. The landmarks were placed by an expert. The resulting directional asymmetry can be seen in the left side of Figure 7. The significance map of the asymmetry is in right side of Figure 7. The group includes both male and female subjects. Each sex can be analyzed separately and compared (see Figure 8). Local asymmetry information can be summarized into a single value per specimen called total asymmetry. Neither individual asymmetry nor total asymmetry discriminates between sexes in our sample. On the other hand, the total orientation asymmetry shows the difference rather clearly (see Figure 9). This finding is confirmed by results from [Liu03] which indicates that the orientation asymmetry is important in comparative studies. Our method is new in ability to capture asymmetry and correspondence in more complex shapes.

In the third (Figure 10) test we used 50 scans of human palate (a convex surface surrounded by dental arc). The shape of this particular structure is studied in order to describe impact of the therapy on patients with cleft of the lip and palate. Specifically the palate has an altered shape and its further development is influenced by inadequate growth of maxilla. The shape of the palate has great individual variation. The above-described methodology is useful for the comparison of the palatal shape depending on the type and extent of the cleft, the utilized surgical method and the surgeon's proficiency, as well as numerous other factors associated with surgical and orthodontic treatment [Sma03]. The shape of the palate is a problematic object from the view of geometric morphometry as the only apparent landmarks can be placed at the locations of teeth.

The whole algorithm is not very computationally demanding. The analysis of all 102 faces took approximately 283 seconds on Intel Core i7 machine. The computation consists of many independent parts that can be computed in parallel. For instance the group correspondence construction in fact involves N independent processes, N being the count of meshes in the group. Similarly, individual mesh asymmetry can be computed independently for each mesh.

6. CONCLUSIONS

The dense correspondence algorithm has been used in many geometric morphometry studies, e.g. [Ham04] [Bej12] [Vel12]. It extends the ability of GMM methodology to capture shapes from simple primitives such as homologous landmark to triangular meshes representing the whole surface of the object. We continued in this trend to study asymmetry of groups of shapes in a novel way but using the traditional approach of decomposition into directional and fluctuating asymmetry.

The most significant contribution of our work is the application of dense correspondence mapping, as introduced in [Hut01], to map asymmetries onto complex geometry, as opposed to [Liu03] that only used cylindrical surfaces. Furthermore, our approach utilizes landmark-based registration, which makes it more adjustable in certain scenarios than most automatic algorithms [Com08b].

From a practical point of view and in comparison to commercially available tools implementing dense correspondence modeling algorithms such as

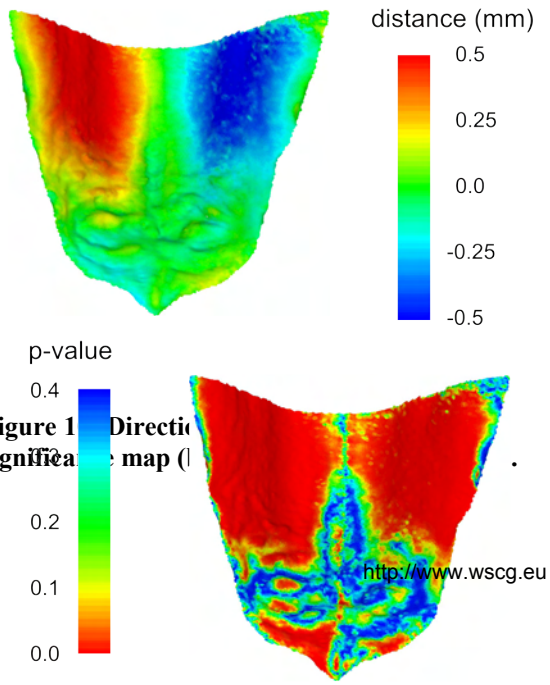


Figure 1 Directi
significa map (

MorphoStudio 3.0 (from BioModeling Solutions, 2006) we sped up the correspondence matching by employing a kd-tree search data structure.

7. ACKNOWLEDGEMENTS

Our thanks goes to the members of Laboratory of 3D Analytical and Visualization methods, Charles University in Prague for providing us with real data. This research has been supported by GAUK 309611 research grants from the Grant Agency of Charles University in Prague and MSM 0021620843 from the Ministry of Education, Youth and Sports of the Czech Republic.

8. REFERENCES

[Bej12] Bejdová, Š., Krajíček, V., Peterka, M., Trefný, P., and Velemínská, J. Variability in Palatal Shape and Size in Patients with Bilateral Complete Cleft Lip and Palate Assessed Using Dense Surface Model Construction and 3D Geometric Morphometrics, J Cranio Maxill Surg, 40 [3]:201–208, 2012.

[Boo97] Bookstein, F.L. Morphometric Tools for Landmark Data : Geometry and Biology, Cambridge University Press, 1997.

[Com08a] Combes, B., Hennessy, R., Waddington, J., Roberts, N., and Prima, S. Automatic Symmetry Plane Estimation of Bilateral Objects in Point Clouds, In IEEE Conference on Computer Vision and Pattern Recognition (CVPR'2008), Anchorage, USA, IEEE, 2008.

[Com08b] Combes, B. and Prima, S. New Algorithms to Map Asymmetries of 3D Surfaces, In Proceedings of the 11th International Conference on Medical Image Computing and Computer-Assisted Intervention - Part I, Berlin, Heidelberg, Springer-Verlag, pp.17–25, 2008.

[Dam11] Damstra, J., Oosterkamp, B.C.M., Jansma, J., and Ren, Y. Combined 3-dimensional and Mirror-image Analysis for the Diagnosis of Asymmetry, Am J Orthod Dentofac, 140 [6]:886–894, 2011.

[Fou11] Fournier, M., Combes, B., Roberts, N., Keller, S., Crow, T.J., Hopkins, W.D., and Prima, S. Surface-based Method to Evaluate Global Brain Shape Asymmetries in Human and Chimpanzee Brains, In Proceedings of the 8th IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Chicago, USA, IEEE, pp.310–316, 2011.

[Ham04] Hammond, P., Hutton, T.J., Allanson, J.E., Campbell, L.E., Hennekam, R.C.M., Holden, S., Patton, M.A., et al. 3D Analysis of Facial Morphology, Am J Med Genet, 126A [4]:339–348, 2004.

[Hut01] Hutton, T.J., Buxton, B.F., and Hammond, P. Dense Surface Point Distribution Models of the Human Face, In Proceedings IEEE Workshop on Mathematical Methods in Biomedical Image Analysis, Kauai, Hawaii, IEEE, pp.153–160, 2001.

[Hut03] Hutton, T.J., Buxton, B.F., Hammond, P., and Potts, H.W.W. Estimating Average Growth Trajectories in Shape-space Using Kernel Smoothing, IEEE Trans Med Imaging, 22 [6]:747–753, 2003.

[Kli02] Klingenberg, C.P., Barluenga, M., and Meyer, A. Shape Analysis of Symmetric Structures: Quantifying Variation Among Individuals and Asymmetry, Evolution, 56 [10]:1909–1920, 2002.

[Liu03] Liu, Y. and Palmer, J. A Quantified Study of Facial Asymmetry in 3D Faces, In Proceedings of the IEEE International Workshop on Analysis and Modeling of Faces and Gestures, Nice, France, IEEE, pp.222–229, 2003.

[OGr99] O'Grady, K.F. and Antonyshyn, O.M. Facial Asymmetry: Three-Dimensional Analysis Using Laser Surface Scanning, Plast Reconstr Surg, 104 [4]:928–937, 1999.

[Ola07] Ólafsdóttir, H., Lanche, S., Darvann, T.A., Hermann, N.V., Larsen, R., Ersbøll, B.K., Oubel, E., et al. A Point-wise Quantification of Asymmetry Using Deformation Fields: Application to the Study of the Crouzon Mouse Model, In Proceedings of the 10th International Conference on Medical Image Computing and Computer-assisted Intervention, Berlin, Heidelberg, Springer-Verlag, pp.452–459, 2007.

[Pal94] Palmer, A.R. Fluctuating Asymmetry Analyses: A Primer, In Proceedings of the International Conference on Developmental Instability: Its Origins and Evolutionary Implications, Dordrecht, The Netherlands, Kluwer, pp.335–364, 1994.

[Sch06] Schaefer, K., Lauc, T., Mitteroecker, P., Gunz, P., and Bookstein, F.L. Dental Arch Asymmetry in an Isolated Adriatic Community, Am J Phys Anthropol, 129(1):132–42, 2006.

[Sma03] Šmahel, Z., Trefný, P., Formánek, P., Müllerová, Ž., and Peterka, M. Three-Dimensional Morphology of the Palate in Subjects With Isolated Cleft Palate at the Stage of Permanent Dentition, Cleft Palate-Cran J, 40 [6]:577–584, 2003.

[Val62] Van Valen, L. A Study of Fluctuating Asymmetry, Evolution, 16 [2]:125–142, 1962.

[Vel12] Velemínská, J., Bigoni, L., Krajíček, V., Borský, J., Šmahelová, D., Cagáňová, V., and Peterka, M. Surface Facial Modelling and Allometry in Relation to Sexual Dimorphism, Homo, 63 [2]:81–93, 2012.

Preprocessing for Quantitative Statistical Noise Analysis of MDCT Brain Images Reconstructed Using Hybrid Iterative (iDose) Algorithm

Petr Walek
Dept. of Biomedical Engineering
Brno University of Technology
612 00, Brno, Czech republic
walek@feec.vutbr.cz

Jiri Jan
Dept. of Biomedical Engineering
Brno University of Technology
612 00, Brno, Czech republic
jan@feec.vutbr.cz

Petr Ourednicek
Philips Czech Republic
155 00 Prague, Czech Republic
petr.ourednicek@philips.com

Jarmila Skotakova
Children's Hospital - Faculty Hospital
Masaryk University
625 00, Brno, Czech Republic
j.skotakova@post.cz

Igor Jira
Children's Hospital - Faculty Hospital
Masaryk University
625 00, Brno, Czech Republic
igor.jira@volny.cz

ABSTRACT

Radiation dose reduction is a very topical problem in medical X-ray CT imaging and plenty of strategies have been introduced recently. Hybrid iterative reconstruction algorithms are one of them enabling dose reduction up to 70 %. The paper describes data preprocessing and feature extraction from iteratively reconstructed images in order to assess their quality in terms of image noise and compare it with quality of images reconstructed from the same data by the conventional filtered back projection. The preprocessing stage consists in correction of a stair-step artifact and in fast, precise bone and soft tissue segmentation. Noise patterns of differently reconstructed images can therefore be examined separately in these tissue types. In order to remove anatomical structures and to obtain the pure noise, subtraction of images reconstructed by the iterative iDose algorithm from images reconstructed by the filtered back projection is performed. The results of these subtractions called here residual noise images and are the used to further extract parameters of the noise. The noise parameters, which are intended to serve as input data for consequent multidimensional statistical analysis, are the standard deviation and power spectrum of the residual noise. This approach enables evaluation of noise properties in the whole volume of real patient data, in contrast to noise analysis performed in small regions of interest or in images of phantoms.

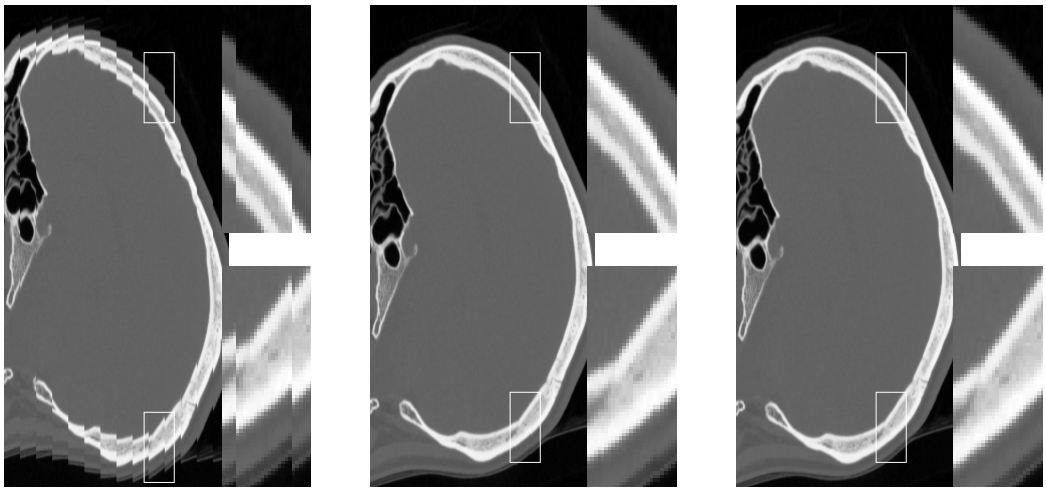
Keywords

X-ray computed tomography, dose reduction, skull segmentation, noise power spectrum.

1 INTRODUCTION

Multidetector X-ray computed tomography (MDCT) imaging is very important for medical diagnostics and quantity of pathological states diagnosed by a MDCT is steadily increasing. This fact causes, in conjunction with rising accessibility of MDCT examinations, increase of the average population radiation exposure which, in spite of unexceptionable diagnostics outcome, constitutes certain health risk especially for pediatrics patients. Effective dose introduced by each MDCT scan depends on many factors and nowadays usually falls within range from 1 to 14 mSv which can be considered as a high value in comparison with the annual dose received from natural sources in the Czech Republic (2.5 mSv).

In order to be compliant with the ALARA principle each of the major MDCT manufacturers have focused their research on as large radiation dose reduction as possible. As a result of this increased effort there have been introduced new strategies for reducing radiation dose, for example tube current modulation (in both angular and longitudinal directions), elimination of over-ranging effect, dual energy scanning, bowtie filtering and replacement of a filtered back projection (FBP) by iterative reconstruction algorithm [MPB+09], [Goo12]. Among a range of mentioned methods the iterative re-



(a) (b) (c)

Figure 1: Saggital slice of brain image (with magnified sections): (a) original slice, (b) slice after registration by phase correlation, (c) slice after registration by gradient descent optimization.

construction one takes exceptional position by producing quality images, even when drastic radiation dose reduction (up to 70%) is applied [FB11]. Such a dose reduction is alowed by inclusion of photon counting statistics and models of acquisition process into reconstruction. So far, each of the available iterative reconstructions are vendor specific and further details about used algorithms are unknown. General description of iterative reconstruction methods can be found in [BKK12] and references therein.

Many studies dealing with problem of quality evaluation of iteratively reconstructed images have been proposed recently. These studies are targeted either to assessment of image quality in small regions of interest in real patient data [MNS+10] or to evaluation of images acquired by scanning of phantoms [MGB+12]. Former approach utilizes information only from spatially limited range and thus can not affect whole complexity of image noise, e.g. differences between noise in anatomical structures. The phantom approach analyzes noise properties in homogeneous regions of artificial images and there is difficulty to relate results obtained by this approach to clinical practice. In order to overcome previously mentioned drawbacks a new way of extraction of noise parameters from whole volume of real patient data is presented in this paper.

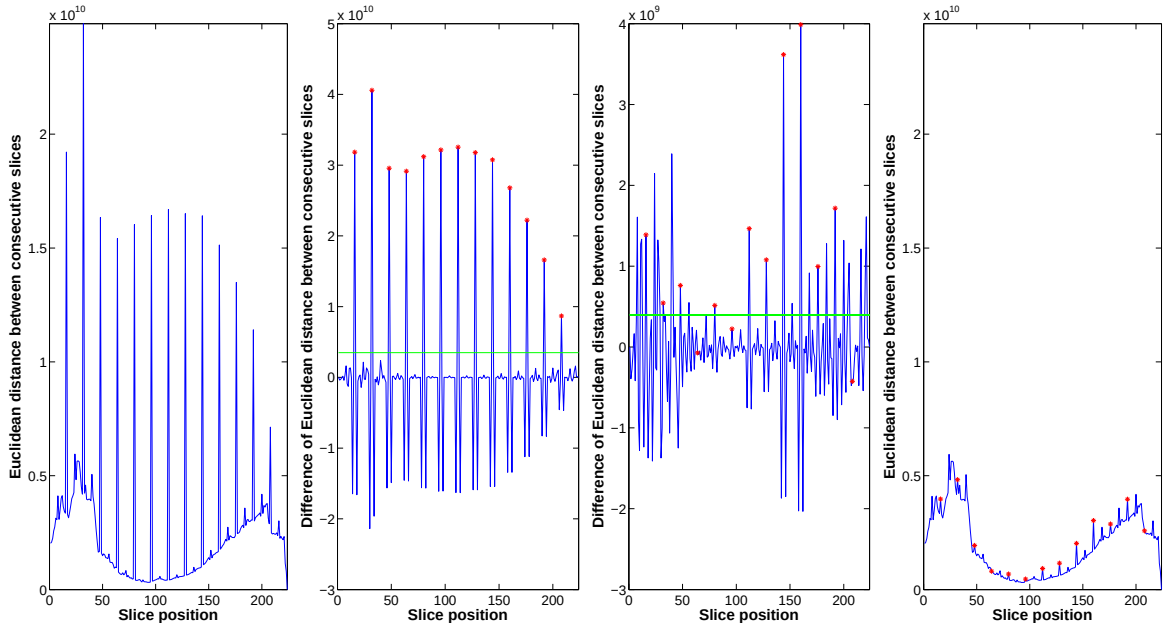
2 DATA ACQUISITION

Our study is targeted to head MDCT images, acquired by the Philips Brilliance CT 64-channel scanner and reconstructed by a prototype of the Philips iterative reconstruction method called iDose during ordinary operation of radiological center in Children's hospital in Brno. Acquired raw data were once reconstructed by the conventional filtered back projection and four times using the iDose algorithm, every time with differently adjusted parameters. The parameters adjusted before each reconstruction are inclusion level of iDose reconstruction expressed in percents (chosen to be 30 %, 50 % and 70 % and in this paper labeled as an ID30, ID50 and ID70), and Multi Resolution which was turned on only together with the iDose level ID70 (in this paper labeled as an ID70MR). Meaning of iDose levels is folowing, images reconstructed by FBP have equal standard deviation of noise as images acquired with 30 % less dose and reconstructed by ID30.

A statistical data set contains forty patients uniformly divided into male and female, aged in range from three months to sixty years. A certain group of patients was scanned with regular dose according to a scanning protocol, other group also with regular dose but in a high quality imaging mode (HQ) and the last group in a high quality mode with radiation dose reduced about 30 % (30HQ). Note that dose reduction was obtained by a uniform reduction of tube current.

3 CORRECTION OF A STAIR-STEP ARTIFACT

Acquired images suffer from the very severe stair-step artifact especially after three-dimensional reformatting to a sagittal plane as can be clearly seen in Fig. 1a. In general a stair-step artifact is caused by using wide collimation and a non-overlapping reconstruction interval especially using multisection scanning [BK04]. Artifact introduces translational shift into sub-volumes located identically according to sections acquired in one gantry rotation during multisection scanning. Such a shift cause many artificial edges in a sagittal plane which are able to harm further noise power spectra analysis by introducing artificial high frequencies.



(a) (b) (c) (d)

Figure 2: Euclidean distance between consecutive slices (EDCS): (a) original image, (b) difference of EDCS with detected marginal slices (red stars) and threshold (green line), (c) difference of EDCS after registration using phase correlation, (d) EDCS of finally corrected image

Positional detection of translated sub-volumes margins

The first step in correction of the stair-step artifact is positional detection of margins of displaced sub-volumes. Positions of marginal slices are detected by evaluation of the Euclidean distance similarity function (1), computed between consecutive slices. Variables **a** and **b** in this equation means pixel intensities rearranged to a vector and N is a number of pixels in images.

$$C_E(\mathbf{a},\mathbf{b}) = \sqrt{\sum_{i=0}^{N}(a_i - b_i)^2} \tag{1}$$

Resulting vector of Euclidean distances as a function of slice positions can be seen in Fig. 2a. Despite of clearly visible peaks in the similarity function there is also a slow and strong trend which can possibly preclude the detection, and is removed by differentiation of this curve. Resulting difference of the similarity function can be seen in Fig. 2b and a peaks detection algorithm is applied (note that as a peak is labeled each position in the vector with a bigger value than their neighbors). Detected peaks are thresholded, threshold is determined as the mean of absolute values of the vector (depicted as a green line in Fig. 2b), thereby the most significant peaks, representing positions of the most dissimilar consecutive slices, are obtained. The last step is a determination of a patient table translational increment after one rotation of a gantry which corresponds with sizes of mutually translated sub-volumes. Translational incerement of patient table is computed as the median of vector containing distances between the neighboring detected peaks. Finally detected margins of sub-volumes are plotted on Fig. 2b as red stars.

Registration of displaced sub-volumes

Once positions of mutually translated sub-volumes margins are known registration of sub-volumes is performed. Taking into account a character of the stair-step artifact (a simple translation of sub-volumes) phase correlation technique, originally proposed in [KH75], is chosen as a basis for registration. This method is based on a Fourier shift property stating that a planar shift between two functions is expressed in a Fourier domain as a linear phase difference. Let us take two functions $f_1(x,y)$, $f_2(x,y)$ and suppose that they vary only by a translation about Δx and Δy

$$f_2(x,y) = f_1(x-\Delta x, y-\Delta y). \tag{2}$$

Using Fourier shift property equation (2) can be restated to

$$F_2(u,v) = F_1(u,v).e^{(-i(u.\Delta x + v.\Delta y))} \tag{3}$$

where

$$F_i(u,v) = DFT_{2D}(f_i(x,y)). \tag{4}$$

According to equation (3) shifting of image does not influence its amplitude spectrum. Phase correlation can

be calculated as a inverse Fourier transform of a normalized cross power spectrum

$$p(x,y) = DFT_{2D}^{-1}\left[\frac{F_2(u,v).F_1(u,v)^*}{|F_2(u,v).F_1(u,v)|}\right]. \qquad (5)$$

This phase correlation matrix contains a strong impulse in position $[\Delta x, \Delta y]$ which is detected as the strongest peak. Vector of translation parameters $[\Delta x, \Delta y]'$ for each sub-volume is known and alignment can be performed in a very simple manner as $[x,y]' + [\Delta x, \Delta y]'$. Phase correlation, in its basic form, cannot determine sub-pixel shifts and registration therefore cannot be sufficient, see Fig. 1b. After registration of sub-volumes by phase correlation, difference of Euclidean distance between consecutive slices is computed and thresholded again, see Fig. 2c. Euclidean distances between sub-volumes margins (labeled as red stars) above threshold are then minimized by gradient descent optimization method, providing final correction of stair-step artifact, see Fig. 1c and Fig. 2d.

4 SEGMENTATION OF SKULL AND SOFT TISSUE

Very interesting findings may be done if noise properties of differently reconstructed images are examined separately for hyperdense and hypodense structures (i.e. bones and soft tissue). An automatic, reliable and fast segmentation algorithm is therefore needed which should be capable to distinguish between bones and soft tissue even in a complex structure of a basis cranii and segment not only cortical however also trabecular parts of bones. Distinction between soft tissue and bones is carried out in the following manner:

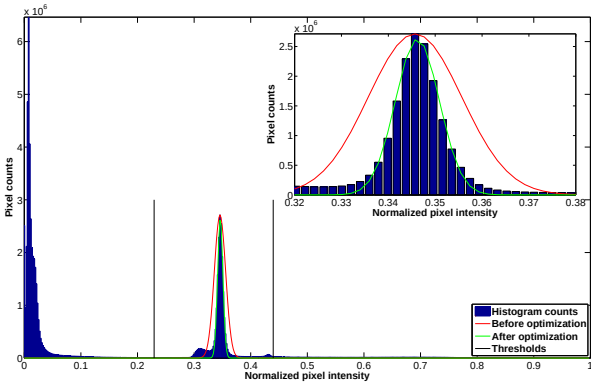


Figure 3: Brightness histogram of the whole brain volume (blue bar graph). Initial Gaussian curve (red plot) is fitted on soft tissue peak (green plot) and final thresholds are depicted as the black lines

- Segmentation of cortical bones.
- Adding trabecular bones parts into segmentation.
- Segmentation of surrounding air and sinuses.
- Segmentation of soft tissue by calculating a complement to segmented bones and surrounding air.

Only the first and the second steps deserves closer attention and are in detail discussed in next two subsections, on the other hand the third step is very similar to the first and the fourth is simple computation of a complement to two binary images (bones and surrounding air segmentations).

Segmentation of cortical bones

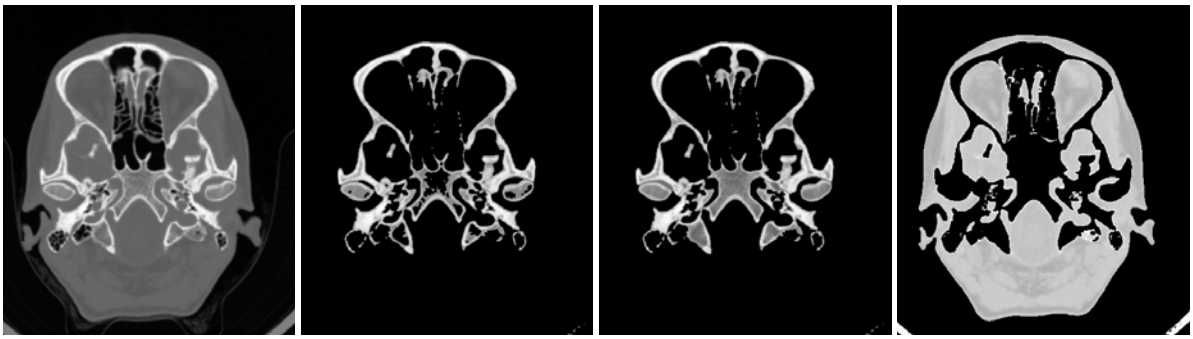
The simplest and fastest method for segmenting cortical bones parts is the intensity thresholding. A threshold is needed for this operation and probably the best way for its automatic determination is evaluation of the image histogram. A typical brightness histogram of the whole brain volume comprises only two distinct peaks, and can be seen in Fig. 3 plotted as a blue bar graph, note that pixel intensities are normalized to be in interval $\langle 0,1 \rangle$. The peak situated at lower intensities belongs to representation of surrounding air and sinuses, while second significant peak belongs to a representation of soft tissue. Intensities belonging to bones are spread over a wide range hence there is no distinct detectable peak. Threshold for cortical bones segmentation is therefore derived from a position of soft tissue peak which is detected in similar way as peaks in chapter 3. Peak with the second highest value is considered to be representation of soft tissue. Detected position of the soft tissue peak serves as a mean μ and magnitude as parameter a of initial Gaussian function (6) used to approximate properties of soft tissue lobe (variance σ is initially selected as 0.01).

$$f(x) = a.e^{\left(-0.5\left(\frac{x-\mu}{\sigma}\right)^2\right)} \qquad (6)$$

The initial Gaussian curve is depicted in Fig. 3 (please notice detailed plot) as a red curve and is optimized by a least-squares curve fitting algorithm in order to find optimal parameters μ and σ (green curve in Fig. 3). Thresholds for segmentation are empirically determined as $\mu - 25.\sigma$ for surrounding air and $\mu + 20.\sigma$ for bones (black lines in Fig. 3). In this way thresholds for bones and surrounding air segmentation are determined automatically and independently on the input data.

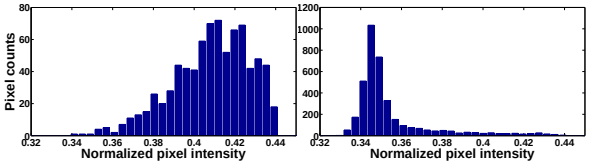
Classification of trabecular bones

Intensities (i.e. Hounsfield units or tissue density) in trabecular parts of bones are partially overlapped with intensities of soft tissue, therefore simple thresholding is only capable to segment cortical parts of bones as can be seen in Fig. 4b (note that resulting binary masks 4b,



(a) Original slice (b) After intensity thresholding (c) Final bones segmentation (d) Soft tissue segmentation

Figure 4: Example of skull and soft tissue segmentation



(a) Histogram of trabecular bone (b) Histogram of Soft tissue

Figure 5: Typical histograms of holes manually classified as soft tissue and trabecular bone

4c and 4d are in this view multiplied with the original slice 4a) and because of that areas being zero fully surrounded by values of one (in this paper called "holes") appears instead trabecular parts of bones. Separation of holes by a boundary tracking technique is therefore next step followed by decision if particular hole represents soft tissue or trabecular bone. As stated before intensities of soft tissue and trabecular bones are partially overlapping, nevertheless their histograms differ in shapes, typical histograms of soft tissue hole and trabecular bone are depicted in Fig. 5. Histograms of trabecular bones parts are, in comparison with soft tissue ones, more compact (histogram counts are smoother) and skewed towards higher intensities. Shape of a particular histogram is objectified by five parameters: entropy (7), compactness (8), relative position of the histogram mean according to position of soft tissue peak in histogram of whole volume (9), skewness (10) and kurtosis (11). In each of the following equations N means sum of all counts in bins (i.e. number of pixels in hole), i is a bin mark and x_i means counts in the bin marked as i.

$$S = -\frac{1}{N}\sum_{i=0}^{n-1} x_i \log(x_i). \tag{7}$$

$$C = \frac{1}{N}\sum_{i=0}^{n-1} \frac{x_i}{\max(x)} \tag{8}$$

$$P_{rel} = \frac{P_{pos}}{\mu}; \quad \mu = \frac{1}{N}\sum_{i=0}^{n-1} x_i i. \tag{9}$$

$$\gamma_1 = \frac{\frac{1}{N}\sum_{i=0}^{n-1}(i-\mu)^3}{\left[\frac{1}{N}\sum_{i=0}^{n-1}(x_i i)^2 - \mu^2\right]^{\frac{3}{2}}}. \tag{10}$$

$$\gamma_2 = \frac{\frac{1}{N}\sum_{i=0}^{n-1}(i-\mu)^4}{\left[\frac{1}{N}\sum_{i=0}^{n-1}(x_i i)^2 - \mu^2\right]^{2}} - 3. \tag{11}$$

Classification of the holes is done by a simple neural network, trained by set of 300 exemplary vectors, each vector is composed of histogram parameters resulting from equations 7 - 11. Each exemplary vector is manually classified and this classification is verified by an experienced radiologist. Final segmentation of bones in complex structure of basis cranii can be seen in Fig. 4c.

5 EXTRACTION OF PARAMETERS FOR STATISTICAL ANALYSIS

By means of the segmentation algorithm proposed in section 4 two binary masks is obtained representing bones and soft tissue. In order to compare noise properties of images reconstructed by the iDose with respect to the conventional FBP technique, anatomical structures must be removed. Removing of anatomical structures is done by subtraction of image reconstructed by the FBP from images reconstructed by the iDose (i.e. images marked by ID30, ID50, ID70 and ID70MR) using binary masks for bones and soft tissue. Results of these subtractions are called residual noise images which can be seen in Fig. 6.

Standard deviation of residual noise

First group of parameters extracted from residual noise images are standard deviations computed from a whole image volume (results can be seen in Fig. 6). Meaning of this parameter is explained considering following thought. A region of interest (ROI) is selected from each reconstructed image (before subtraction) in order

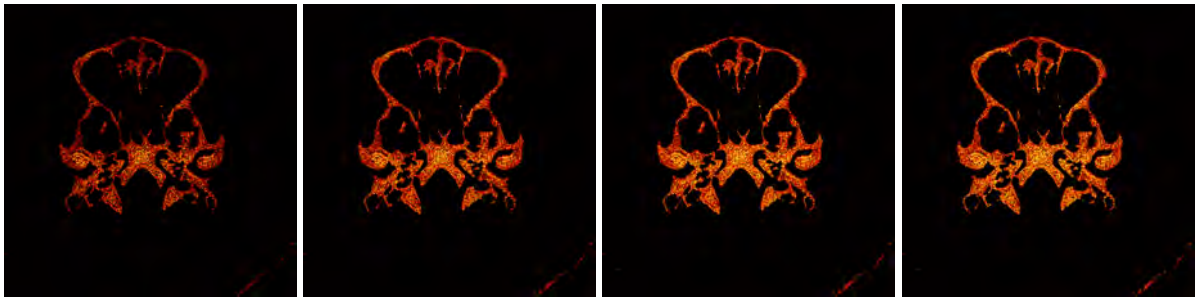


Figure 6: Residual noise images depicted separately for bones (upper row) and soft tissue (bottom row)

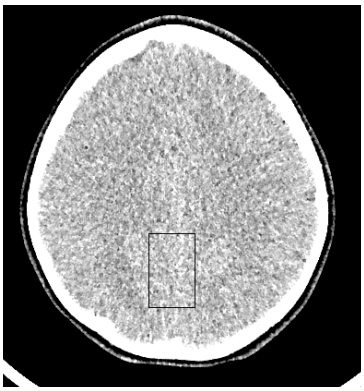


Figure 7: ROI selected from image reconstructed by ID70

to select an area of brain, which should be considered homogeneous. Therefore intensity changes in these ROIs are considered being only random noise patterns, see sample in Fig. 7.

Two parameters are computed from each ROI, a cross covariance with the ROI selected form the FBP reconstruction ($C_{X,FBP}$) and a standard deviation in each ROI (σ_X^2), see Tab. 1 and note that both parameters decreases with increasing iDose level. Subtracting of two random variables FBP and X (X is meant as a particular reconstruction), having variances σ_{FBP}^2 and σ_X^2 and cross-covariance $C_{X,FBP}$, results in new random variable with a standard deviation equal to equation (12).

$$\sigma_{(X-FBP)} = \sqrt{\sigma_{FBP}^2 + \sigma_X^2 - 2.C_{X,FBP}} \qquad (12)$$

Recon. (X)	$C_{X,FBP}$	σ_X^2	$\sigma_{(X-FBP)}$	$\sigma_{E(X-FBP)}$
FBP	276.64	16.63	0.41	0
ID30	234.09	14.12	2.78	2.77
ID50	200.31	12.19	4.95	4.95
ID70	158.81	9.97	7.64	7.64
ID70MR	155.11	9.46	7.47	7.48

Table 1: Noise parameters of selected ROI

A standard deviation of residual noise in investigated ROI $\sigma_{E(X-FBP)}$ is computed from real data and compared with value obtained from equation (12), see Tab. 1. Assuming that anatomical structures are identical in reconstructed images and completely suppressed by subtraction, the standard deviation of residual noise, therefore depends only on a standard deviation of noise in image X and a cross-covariance function between noises in images FBP and X. The standard deviation of residual noise increases with decreasing cross-covariance $C_{FBP,X}$ and increasing difference between σ_{FBP}^2 and σ_X^2 and thus can be considered as a valuable measure indicating improvement of noise properties in images reconstructed by the iDose according to images reconstructed by the filtered back projection. Advantage of this parameter lies in independence on an imaged object and therefore can be directly applicable to real patient data not only to phantoms. On the other hand it provides only relative improvement of noise properties according to the filtered back projection.

By means of the segmentation algorithm proposed in section 4 two binary masks is obtained representing bones and soft tissue. In order to compare noise properties of images reconstructed by the iDose with respect to the conventional FBP technique, anatomical structures must be removed. Removing of anatomical structures is done by subtraction of image reconstructed by the FBP from images reconstructed by the iDose (i.e. images marked by ID30, ID50, ID70 and ID70MR) using binary masks for bones and soft tissue. Results of these subtractions are called residual noise images which can be seen in Fig. 6

Power spectrum of residual noise

Standard deviation provides information only on noise magnitude, however no less important is knowledge about its frequency content. Such an information may be obtained by a noise power spectrum, routinely used as quality measure of MDCT imaging systems [YKH+08], [YKH+08] and [BMG07]. In this study residual noise images (Fig. 6), both for segmented bones and soft tissue, serves as input images for a noise power spectral analysis. Determination of a noise power spectra (NPS) is carried out by a direct digital technique as proposed in [SCJ02] and is computed according to equation (13).

$$S(f_x, f_y) = \frac{b_x b_y}{L_x L_y} \cdot \left\langle \left| DFT_{2D}\left\{ D(x,y) - D_{filt}(x,y) \right\} \right|^2 \right\rangle \tag{13}$$

Each slice is considered to be the one realization of a random noise and is denoted as $D(x,y)$. Individual realizations must be zero mean detrended before NPS calculation, therefore an image filtered by a lowpass Gaussian filter ($D_{filt}(x,y)$) is subtracted from each slice. Applying a two-dimensional Fourier transform and squaring an absolute value of a result ($|\circ|^2$), individual noise power spectrum is obtained. Individual noise power spectra suffer from a very large variance between realizations, power spectrum of a stochastic field (i.e. a process generating random noise) is therefore calculated as a mean value of individual power spectra (in equation (13) outlined by $\langle\circ\rangle$ operator). Fraction in this equation is a normalization term consisting of b_x b_y representing sampling periods and L_y L_x representing sizes in directions x and y, respectively.

Residual noise power spectra are determined in transverse $S(f_x, f_y)$, coronal $S(f_x, f_z)$ and sagittal plane $S(f_y, f_z)$ as can be seen in Fig. 8. A set of an annular sector shaped binary frequency filters, covering in piecewise sense the whole spectrum, is used to extract the final parameters from residual noise power spectra. Filters are used to select a segment of a NPS and the mean of this segment is the sought noise pattern, 36 parameters are extracted from each single residual noise power spectrum.

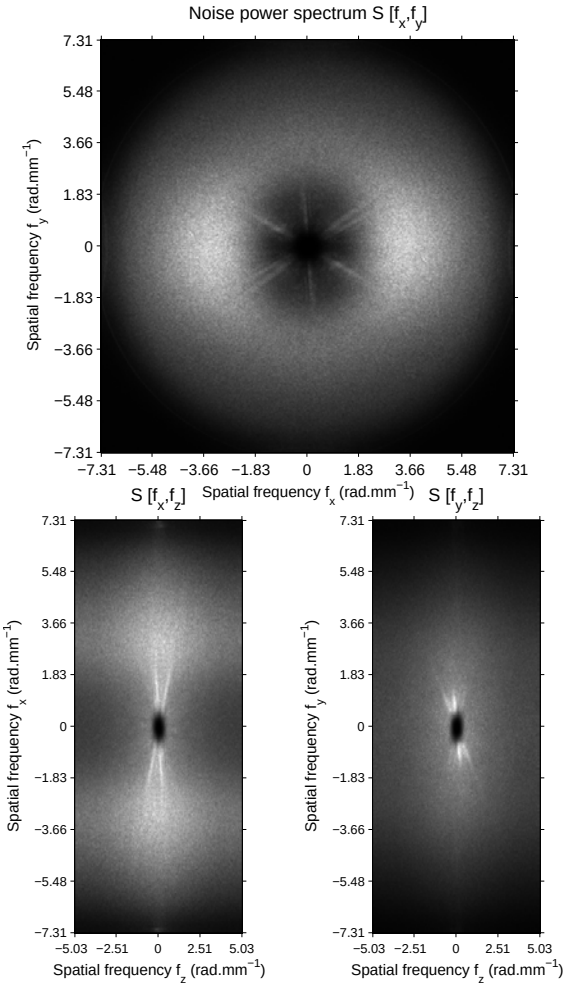


Figure 8: Example of power spectra of residual noise for transverse (upper), coronal (bottom left) and sagittal (bottom right) planes

6 CONCLUSIONS AND FUTURE WORK

Preprocessing of reconstructed image data and extraction of parameters for further statistical analysis of noise in MDCT images reconstructed by the iDose iterative algorithm is proposed in this paper. The preprocessing includes a correction of the stair-step artifact, which may harm further noise feature extraction and segmentation of bones and soft tissue. The proposed algorithm for segmenting bones in head images is fast and reliable even in complex structure of basis cranii, however there are certain drawbacks of this method. Segmentation of cortical bones, especially the ones with weak borders, may not result in areas of zeros fully surrounded by ones in locations of trabecular bones. Therefore boundary tracking algorithm can not label them as a "holes" and such a trabecular

bones remains unsegmented. Another difficulty is the lack of a trabecular structure in bones, especially in images of pediatrics patients. Considering that a trabecular structure in bones causes difference in the shapes of histograms of holes, the lack of this structure can negatively influence reliability of the resulting segmentation.

The parameters used for further statistical analysis are the standard deviation and noise power spectrum of the residual noise. The images formed by the residual noise are obtained by subtracting the images reconstructed by the filtered back projection from the images reconstructed by the iDose algorithm. When obtained by this subtraction, the noise properties can be evaluated in the whole volume of real patient data, on the other hand, the obtained parameters do not reflect the absolute level of the image noise but only the relative improvement with respect to image reconstructed by the FBP.

In order to assess different nature of noise and prove different behavior of the iterative reconstruction in soft tissue and bones the images of residual noise are multiplied with the binary masks obtained by segmentation. According to a convolution property of Fourier transform multiplying of signals results in convolution of their spectra therefore each of the residual noise power spectrum is affected by spectrum of the used binary mask which is moreover varying with respect to the slice position. Statistical inference of general results from such modified spectra may be incorrect and our future goal will be to analyze how strong is this influence and how to overcome this problem.

Considering the separate evaluation of the noise parameters from bones and soft tissue, taking into account the number of iDose reconstructions and the count of parameters extracted from three residual noise power spectra (for transverse, coronal and sagittal plane), we obtain 392 noise parameters per patient. The vectors of the parameters for forty patients can be arranged to matrix of size forty rows and 392 columns where each row can be considered as a single realization of a random process. Multidimensional statistical analysis such as principal component analysis or factor analysis can be used to reveal hidden relations in this matrix. Statistical analysis of the whole matrix can be rather complicated due to high number of the extracted parameters in comparison with quantity of scanned patients therefore selections of groups of parameters must be done (for example selection of low frequency noise). Results of future statistical analysis are expected to clarify relation between dose reduction, iDose level and quantity of image noise and differences between noise properties in soft tissue and bones. In future research proposed algorithms will be adapted to abdominal and thoracic images and typical noise properties of these body parts will be analyzed.

7 ACKNOWLEDGMENTS

Lending of the iDose reconstructor prototype programme package from Philips Healthcare is highly acknowledged as well as the long term data acquisition enabled by the Brno Faculty Hospital, Children's Hospital.

8 REFERENCES

[BK04] Julia F. Barrett and Nicholas Keat. Artifacts in CT: recognition and avoidance. *Radiographics : a review publication of the Radiological Society of North America, Inc*, 24(6):1679–91, 2004.

[BKK12] Marcel Beister, Daniel Kolditz, and Willi A. Kalender. Iterative reconstruction methods in X-ray CT. *Physica medica : PM : an international journal devoted to the applications of physics to medicine and biology : official journal of the Italian Association of Biomedical Physics (AIFB)*, 28(2):94–108, April 2012.

[BMG07] K. L. Boedeker and M. F. McNitt-Gray. Application of the noise power spectrum in modern diagnostic MDCT: part II. Noise power spectra and signal to noise. *Physics in medicine and biology*, 52(14):4047–61, 2007.

[FB11] Dominik Fleischmann and F. Edward Boas. Computed tomography–old ideas and new technology. *European radiology*, 21(3):510–7, March 2011.

[Goo12] Hyun Woo Goo. CT Radiation Dose Optimization and Estimation: an Update for Radiologists. *Korean journal of radiology : official journal of the Korean Radiological Society*, 13(1):1–11, January 2012.

[KH75] C. D. Kuglin and D. C. Hines. The phase correlation image alignment method. *IEEE Conference on Cybernetics and Society*, pages 163–165, 1975.

[MGB+12] Frédéric A. Miéville, François Gudinchet, Francis Brunelle, François O. Bochud, and Francis R. Verdun. Iterative reconstruction methods in two different MDCT scanners: Physical metrics and 4-alternative forced-choice detectability experiments - A phantom approach. *Physica Medica*, January 2012.

[MNS+10] Daniele Marin, Rendon C. Nelson, Sebastian T. Schindera, Samuel Richard, Richard S. Youngblood, Terry T. Yoshizumi, and Ehsan Samei. Low-tube-voltage, high-tube-current multidetector abdominal CT: improved image quality and decreased radiation dose with adaptive statistical iterative reconstruction algorithm–initial clinical experience. *Radiology*, 254(1):145–53, January 2010.

[MPB+09] Cynthia H. McCollough, Andrew N. Primak, Natalie Braun, James Kofler, Lifeng Yu, and Jodie Christner. Strategies for reducing radiation dose in CT. *Radiologic clinics of North America*, 47(1):27–40, January 2009.

[SCJ02] J. H. Siewerdsen, I. A. Cunningham, and D. A. Jaffray. A framework for noise-power spectrum analysis of multidimensional images. *Medical Physics*, 29(11):2655, 2002.

[YKH+08] Kai Yang, Alexander L. C. Kwan, Shih-Ying Huang, Nathan J. Packard, and John M. Boone. Noise power properties of a cone-beam CT system for breast cancer detection. *Medical Physics*, 35(12):5317, 2008.