ISSN 1213-6972 *Volume 19, Number 2, 2011*

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, Univerzitni 8
306 14 Plzen
Czech Republic
skala@kiv.zcu.cz

Managing Editor: Vaclav Skala

Published and printed by 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, University of West Bohemia
Centre for Computer Graphics and Visualization
Univerzitni 8, CZ 306 14 Plzen, Czech Republic
skala@kiv.zcu.cz http://herakles.zcu.cz
Journal of WSCG: URL: http://wscg.zcu.cz/jwscg

Direct Tel. +420-37-763-2473
Direct Fax. +420-37-763-2457
Fax Department: +420-37-763-2402

Editorial Advisory Board
MEMBERS

Journal of WSCG

Contents

Vol. 19, No.2

High-Quality Cartographic Roads on High-Resolution DEMs

Mikael Vaaraniemi
BMW Forschung und Technik GmbH
München, Germany
mikael.va.vaaraniemi@bmw.de

Marc Treib Rüdiger Westermann
Technische Universität München
München, Germany
{treib,westermann}@tum.de

Figure 1: Cartographic rendering of roads in the Vorarlberg region, Austria, and in central Munich, Germany.

ABSTRACT

The efficient and high quality rendering of complex road networks—given as vector data—and high-resolution digital elevation models (DEMs) poses a significant problem in 3D geographic information systems. As in paper maps, a cartographic representation of roads with rounded caps and accentuated clearly distinguishable colors is desirable. On the other hand, advances in the technology of remote sensing have led to an explosion of the size and resolution of DEMs, making the integration of cartographic roads very challenging. In this work we investigate techniques for integrating such roads into a terrain renderer capable of handling high-resolution data sets. We evaluate the suitability of existing methods for draping vector data onto DEMs, and we adapt two methods for the rendering of cartographic roads by adding analytically computed rounded caps at the ends of road segments. We compare both approaches with respect to performance and quality, and we outline application areas in which either approach is preferable.

Keywords

cartography, vector draping, shadow volume, GIS, roads, terrain.

1. INTRODUCTION

Geographic Information Systems (GIS) store, analyze and visualize geo-referenced data. Road networks, land usage regions and selected points of interest are usually stored as vector data. In urban planning, cartography, and for navigation purposes, the visualization of roads on digital terrain models plays an important role [Döl05]. GIS engines should be able to handle and display such vector data efficiently and at high quality. A bare and uncluttered visualization as in paper maps is desirable. This *cartographic* representation of roads requires vivid colors, dark edges, rounded caps and runtime scaling of road width [Kra01, RMM95]. Dynamic scaling allows the perception of roads at every distance. In a cartographic rendering, roads are tinted using vivid colors to distinguish them from the underlying terrain. Associating different colors to each type of road induces an automatic cognitive grouping of similar roads [Kra01]. In addition, dark edges around roads add visual contrast [RMM95]. Examples of such cartographic representations are shown in Fig. 1 and 2. An additional aspect of a cartographic representation are rounded caps at each road segment. This avoids the ap-

Figure 2: Cartographic rendering of road maps using vivid colors and dark edges to achieve a high visual contrast to the underlying terrain.

pearance of cracks between segments and makes the visualization more appealing by introducing smooth endings and avoiding undesirable angular corners.

Another important information layer in GIS is the digital elevation model (DEM). It is usually given as raster data defining a 2.5D height map. Since the resolution and size of these DEMs are increasing rapidly, rendering approaches must be capable of dealing with TBs of data and gigantic sets of primitives that have to be displayed at high frame rates. To cope with these requirements, visualization techniques employ adaptive Level-Of-Detail (LOD) surface triangulations [LKR+96] and data compression, combined with sophisticated streaming and pre-fetching strategies [DSW09]. In such scenarios, the combined visualization of roads and a high-resolution DEM in a single visualization engine becomes a challenging task.

The main contribution of this paper is a method for rendering cartographic roads with rounded caps on high-resolution DEMs. We extend existing vector draping methods by introducing the possibility to compute caps analytically, thus avoiding an explicit triangulation. In this way we achieve a high-quality appearance without increasing the number of geometric primitives to be rendered. Furthermore, we introduce screen-space road outlines, runtime width scaling, and correct treatment of road intersections.

We have integrated our method into a tile-based terrain rendering engine. During preprocessing, this engine builds an multiresolution pyramid for both the DEM and the photo texture. It then partitions each level into square tiles, creating a quad tree. Each tile stores a Triangulated Irregular Network (TIN) representation of the DEM along with the photo texture. During runtime, tiles are chosen based on a maximum allowed screen-space error. In combination, this enables interactive 3D browsing of high-resolution terrain data with superimposed cartographic roads.

2. RELATED WORK

Terrain Rendering. Terrain rendering approaches using rasterization have been studied extensively over the last years. They employ the GPU to render large sets of polygonal primitives, and they differ mainly in the hierarchical height field representation used. There is a vast body of literature related to this field and a comprehensive review is beyond the scope of this paper. However, Pajarola and Gobbetti [PG07] discuss the basic principles underlying such techniques and provide many useful algorithmic and implementation-specific details. A thorough discussion of terrain rendering issues that are specifically related to high resolution fields is given in [DSW09].

Vector Data. The mapping of vector data on DEMs is an active research subject. The existing methods can be broadly classified into geometry-based, texture-based and shadow volume-based approaches.

Geometry-based methods generate and render separate primitives for the vector data. As the sampling frequency of the vector data generally does not match the triangulation of the underlying terrain, an adaption to the terrain triangulation and its LOD scheme is necessary. Because of this preprocess, geometry-based algorithms are strongly tied to the terrain rendering system and usually only allow static vector data [ARJ06, SGK05, WKW+03].

Texture-based techniques map the vector data onto a DEM in two steps: first, the data is rendered into off-screen textures either at runtime or in a preprocess. Afterwards, these textures are overlaid onto the terrain using texture mapping [DBH00]. This approach does not produce any aliasing artifacts thanks to hardware-supported texture filtering. Additionally, these methods are independent of the underlying terrain triangulation algorithms.
Static texturing methods provide high performance, but do not allow runtime changes of rendering parameters. Further problems occur at large zoom factors, as only limited resolution textures can be precomputed—there is an inherent tradeoff between the memory requirements and the obtainable quality [DBH00]. Therefore, Kersting and Döllner [KD02] combine this approach with on-demand texture pyramids: associating each quadtree region with an equally sized texture allows on-the-fly generation of appropriate textures. Dynamic vector data can be visualized if these textures

are created in each frame. However, this severely impacts performance, as many render target switches are needed. To overcome this, Schneider et al. [SGK05] introduce an approach using a single reparameterized texture for the vector data, analogously to perspective shadow mapping (PSM) [SD02]. As in PSM, some aliasing artifacts occur.

Bruneton and Neyret [BN08] present an approach that adapts the terrain heightfield to constraints imposed by the vector data (e.g. to enforce locally planar roads). Their method works only on regular meshes and would be difficult to generalize to our TIN-based terrain system. It is also not feasible for high-resolution terrain data. Additionally, an adaption of the heightfield is only necessary if the terrain resolution is insufficient to resolve such constraints, or if real-time editing is desired.

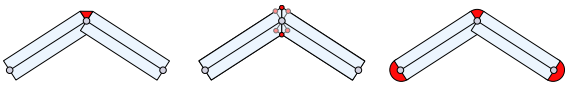
A **shadow volume-based** approach, recently introduced by Schneider and Klein [SK07], uses the stencil shadow volume algorithm to create pixel-exact draping of vector data onto terrain models. A stencil mask is created by extruding polygons along the nadir and computing the screen-space intersection between the created polyhedra and the terrain geometry. Using this mask, a single colored fullscreen quad is drawn. For each color, a separate stencil mask has to be generated. However, as the number of different vector data colors is typically small, this is not a major problem. The approach does not require any precomputations and is thus completely independent of the terrain rendering algorithm.

Our goal is to render *cartographic* roads on a *high-resolution* DEM. Continuous road scaling is a prerequisite, which makes texture-based approaches unsuitable. Likewise, a runtime triangulation of roads to match the DEM is not feasible, so most existing geometry-based approaches are not usable in our case.

We chose to use the shadow volume approach, as it does not require a preprocess and thus allows for runtime scaling of roads. It also provides pixel-exact projections. As a simpler and faster alternative, we also investigate a geometry-based approach where we adapt only the road centerlines to the DEM, so road scaling remains possible.

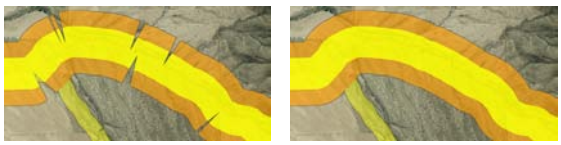
3. CARTOGRAPHIC ROADS

In GIS, roads are usually stored as vector data, i.e. as a collection of 2D polylines. One possibility to visualize such data is to convert the vector data into geometric primitives that are rendered on top of the terrain. However, a naive extrusion of each line segment to a quad results in the appearance of cracks between segments. The higher the curvature of a polyline, the more these cracks become visible. Two pragmatic solutions exist: filling the holes with additional triangles (see Fig. 3(a)) or connecting the corners of adjacent quads (see Fig. 3(b)). Both solutions are only possible if adjacency information is available. In real data sets, however, this information is commonly incomplete. Fig. 4 shows an example from a real data set where one continuous road is represented by several individual polylines, resulting in cracks between adjacent road segments where the polylines meet. We therefore choose a robust and elegant solution, which draws caps to avoid the appearance of cracks (see Fig. 3(c)) and does not require adjacency information. In addition to



(a) Extra triangle (b) Moving corners (c) Rounded caps

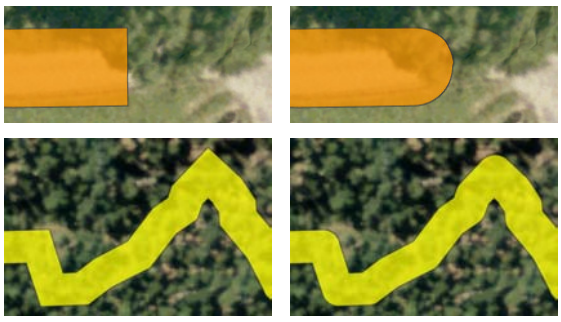
Figure 3: Methods for removing cracks between quads.



(a) Cracks (b) Fixed with caps

Figure 4: Cracks occur because of missing adjacency information.

filling cracks, this approach generates visually pleasant smooth road endings (see Fig. 5, top). It also naturally handles sharp turns in a road (Fig. 5, bottom). Many major navigation systems visualize roads using rounded caps, for example Nokia with Ovi Maps, Google with Google Maps, Navigon and TomTom. It has become a de-facto standard technique when rendering cartographic roads [Phy09]. A naive method for rendering caps is the triangulation of a half-circle, leading to a large number of additional triangles per segment. Furthermore, the discrete triangulation becomes visible at large zoom factors. In the following sections, we present two methods that allow using perfectly round caps while avoiding an increase of the triangle count.



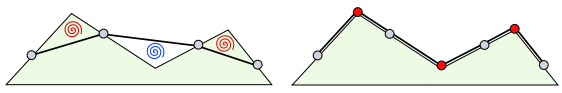
(a) No rounded caps (b) With rounded caps

Figure 5: Quality improvement with rounded caps.

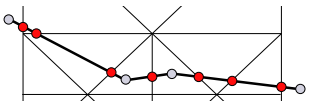
4. GEOMETRIC APPROACH

Our first method renders cartographic roads using a geometry-based approach. From the initial polyline

representation of a road, we individually process each line segment defined by successive vertices. In a preprocess, these lines are clipped against the terrain mesh in 2D, inserting additional vertices at each intersection (see Fig. 6). For more details on this preprocess, see section 6.1.



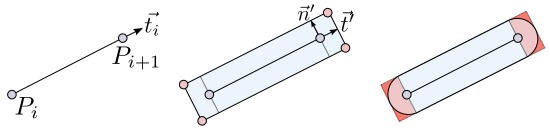
(a) Incorrect mapping of a road (grey vertices); problematic areas are marked by spirals (b) Correct mapping of the road using additional vertices (red)



(c) Top-down view with additional vertices

Figure 6: Geometry-based mapping of roads onto a terrain mesh.

To render rounded caps, we do not explicitly triangulate half-circles at the beginning and the end of each road segment. Instead, we render a single quad encompassing an entire road segment and evaluate the caps analytically in a shader program [Gum03] (see Fig. 7).



(a) Line segment i (b) Generated quad (c) Analytical caps

Figure 7: Analytical evaluation of rounded caps on a base quadrilateral.

We use the endpoints P_i and P_{i+1} of each line segment and the tangent $\vec{t}_i$ to generate a quad encompassing both capped ends (see Fig. 7(a) and (b)).

The caps are cut out of the generated quad in a pixel shader. We create a normalized local coordinate system inside both caps [RBE+06], which allows determining those fragments of a quad that are outside the cap and have to be discarded (see Fig. 7(c)).

Given points P_0, P_1, the ratio h between their distance $d = \overline{P_0P_1}$ and the cap radius $\frac{w}{2}$ is given by

$$h = \frac{w}{(d + 2 \cdot \frac{w}{2})} = \frac{w}{d+w}.$$

Equipped with h, we generate the local coordinates inside the caps with

$$x_{cap} = \frac{|x|-1}{h} + 1 \quad , \qquad y_{cap} = |y|.$$

If $x_{cap} > 0$, the fragment lies inside the cap area (the red area in Fig. 7(c)). If $x_{cap}^2 + y_{cap}^2 > 1.0$, it is outside of the half circle that builds the cap, and is discarded.

5. SHADOW VOLUME APPROACH

Our second algorithm is an extension of the shadow volume-based approach introduced by Schneider and Klein [SK07]. We extrude the road geometry along the nadir and apply a stencil shadow volume algorithm [Cro77, Hei91]. Thus, we compute the intersections between the extruded roads with the terrain geometry, resulting in per-pixel accurate projections onto the terrain. Similar to the approach described in section 4, we extend this algorithm by adding analytic rounded caps. We enlarge the geometry of each line segment to encompass the caps, and construct a local coordinate system that allows us to determine the fragments lying inside or outside the cap area. In the inside area, we analytically evaluate the caps via an intersection test between a ray and a cylinder and compute the depth value of the intersection point to be used during the depth test.

5.1. Intersection

From the camera position O, the fragment position F, and the view direction $\vec{v} = (F-O)/|F-O|$ we construct the view ray $R = O + t\vec{v}$. Given such a ray, the intersection of the ray with the cylinder spanned by the cap can be computed. Because the cylinder is always aligned with the z axis (the nadir), we can replace the 3D ray-cylinder test by a 2D ray-circle test in the xy plane (see Fig. 8).

A circle with center C and radius r is defined by the equation

$$(X-C)^2 = r^2.$$

Inserting the ray R into this equation with $\vec{c} := O - C$ yields

$$((O + t\vec{v}) - C)^2 = (\vec{c} + t\vec{v})^2 = r^2.$$

Expanding this results in the quadratic equation

$$(\vec{v}\cdot\vec{v})\, t^2 + 2\,(\vec{v}\cdot\vec{c})\, t + (\vec{c}\cdot\vec{c} - r^2) = 0.$$

Solving for t gives the discriminant

$$d = 4\,(\vec{v}\cdot\vec{c})^2 - 4\,(\vec{v}\cdot\vec{v})\,(\vec{c}\cdot\vec{c} - r^2).$$

If $d \leq 0$, there is none or only a single solution to the quadratic equation. This means that the ray does not hit the cap at all, or just grazes it. In this case, we discard the fragment. Otherwise, we get

$$t_{1/2} = \frac{-2\,(\vec{v}\cdot\vec{c}) \;\pm\; \sqrt{d}}{2\,(\vec{v}\cdot\vec{v})}.$$

For front faces, $\min(t_1, t_2)$ is the correct solution, for back faces it is $\max(t_1, t_2)$.

So far, we have assumed that the road geometry is extruded toward infinity to generate the shadow volumes. Since this is wasteful in terms of rasterization fill rate, we consider the height field for limiting the extent of

the shadow volumes. Assuming the terrain being partitioned into tiles, it is sufficient to extrude each line segment only within the extent of the bounding box of the tile it belongs to.
To accommodate this, the intersection algorithm has to be extended to handle the top and bottom sides of the extruded polyhedron: If the 2D distance between F and C is smaller than the cap radius (which can only happen for fragments belonging to the top or bottom side), F already gives the final intersection.

5.2. Numerical Precision

The algorithm as presented so far suffers from problems caused by limited numerical precision. One such problematic situation is depicted in Fig. 8: The intersection between each ray and the cylinder is computed twice, once for the front face of the bounding box (corresponding to F_0 in the figure) and once for the back face (corresponding to F_1). The ray direction is computed as $F_0 - O$ and $F_1 - O$, respectively. Because of small perturbations in F_0 and F_1, which are caused by the limited precision of the interpolation hardware, one of the intersection tests may report an intersection while the other one does not. This results in inconsistent output causing visible artifacts.

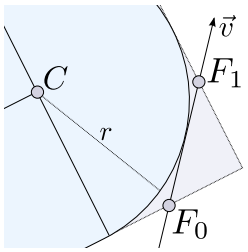


Figure 8: Numerically problematic ray-circle intersection.

In order to achieve consistent results, we compute both intersections in the same shader invocation: We render the geometry with front face culling enabled, and analytically compute the entry point into the bounding box of the extruded road. We then compute both intersections between the ray and the road as described above. This results in two depth values z_0, z_1 that need to be compared to the terrain depth z_t. We therefore replace the regular depth test with a custom two-sided test: z_t is read from a texture created as a secondary render target during the terrain rendering pass. If $z_0 < z_t < z_1$, then the road volume intersects the terrain geometry; otherwise, we discard the fragment.

Two beneficial side effects of this approach are that only half the amount of geometry needs to be rasterized compared to the naive approach, and that in contrast to the original shadow volume algorithm it does not require the rendering of full-screen quads to color the intersections.

6. IMPLEMENTATION DETAILS

In our proposed GIS engine, we visualize vector data e.g. from the OpenStreetMap project [Ope10]. Road networks are stored as a collection of polylines. Each polyline has a *functional road class* (FRC) [Tal96], defining a distinct width and color. For efficient data management at runtime, we partition the vector data into quadtree tiles, similar to the terrain data. Inside each tile, roads are stored sorted by their FRC.

6.1. Geometry Clipping

To avoid an incorrect mapping of roads onto the DEM in the geometric approach as in Fig. 6(a), we apply a preprocess where the centerline of each road segment is clipped against the terrain mesh in 2D. Additional vertices are inserted at each intersection (see Fig. 6(c)). However, finding the exit point of a line in a triangle by line-line intersection tests with the triangle edges provides poor numerical stability. We therefore perform these calculations in barycentric coordinates as illustrated in Fig. 9.

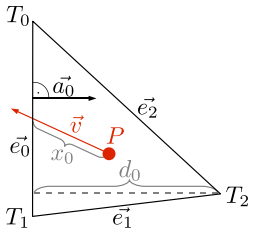


Figure 9: Computing line – triangle edge intersections.

We trace a line starting at point P along the normalized direction vector $\vec{v}$ in the triangle defined by the vertices T_0, T_1 and T_2. The change in the barycentric coordinate λ_2 of P with respect to T_2 is given by the signed distance moved along $\vec{a_0}$ divided by the distance d_0 of T_2 from $\vec{e_0}$, where $\vec{a_0}$ is a normalized vector perpendicular to $\vec{e_0}$ and pointing inside the triangle. When moving along $\vec{v}$, this becomes $(\vec{a_0} \cdot \vec{v})/d_0$. If this value is larger than zero, $\vec{v}$ is pointing away from $\vec{e_0}$ and we skip this edge. Otherwise, the maximum distance x_0 we can move along $\vec{v}$ before we hit $\vec{e_0}$ is given by

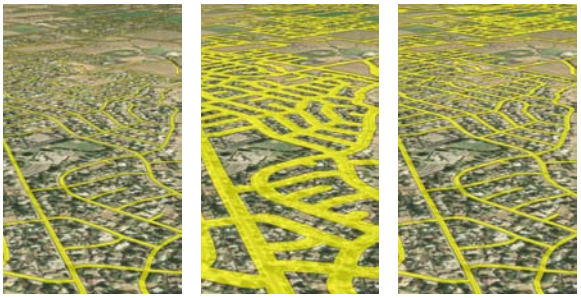
$$x_0 = \frac{\lambda_2 d_0}{\vec{a_0} \cdot \vec{v}}.$$

This can be done analogously for the other edges to compute x_1 and x_2; the smallest of these provides the actual exit point. At this point, an additional vertex is inserted into the polyline.

6.2. Cartographic Rendering

Scaling. In cartographic rendering, roads should be visible at all zoom levels. Therefore, while zooming out our system scales the roads' widths continuously.

The scaling factor is determined by the distance to the viewer. To avoid that roads close to the viewer become too wide, we only scale roads that are further away from the user than a given distance threshold (see Fig. 10).



(a) No scaling (b) Constant scaling (c) Distance-based scaling

Figure 10: Scaling of road width. Without scaling, distant roads become too narrow (left). A constant scale makes close roads too wide (middle). Distance-based width scaling gives satisfactory results (right).

Intersections. At crossroads or junctions, multiple roads of potentially different FRCs overlap, resulting in visible artifacts caused by additive blending. To resolve this problem, we draw roads into an offscreen render target without blending, in increasing order of importance.
The same approach allows for an easy integration of multi-colored roads by drawing a road multiple times with different widths and colors. This increases the geometry count proportionally to the number of colors, but since typically only a few important roads use multiple colors, this is acceptable. Fig. 11 demonstrates the correct handling of intersections of roads with different FRCs, including a two-color motorway.

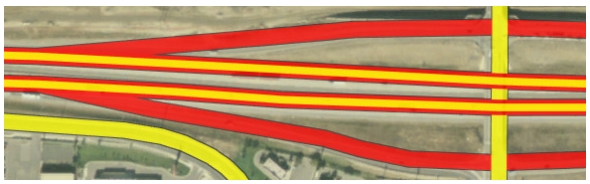


Figure 11: Correct handling of road intersections.

Outlines. To distinguish cartographic roads from the underlying terrain, we add dark edges around roads to increase contrast [RMM95]. To detect edges in screen space, we use a 3×3 or 5×5 kernel to find the local maximum road intensity α_{max} around each fragment. The road intensity is the road opacity for pixels which are covered by a road, and 0 otherwise. The difference $\alpha_{\text{max}} - \alpha_{\text{current}}$ defines the resulting edge intensity. Fig. 12 demonstrates the increase in visibility achieved by using outlines around roads.

(a) Without outlines (b) With outlines

Figure 12: Improving visibility by using dark outlines.

7. RESULTS

We have tested the proposed algorithms using three high-resolution data sets:

- A DEM of the US State of Utah, covering an area of about $276,000$ km^2 at a geometric resolution of 5 m. The road data set contains about $6,839,000$ vertices (216 MB).
- A DEM of Bavaria in Germany, covering an area of about $70,500$ km^2 at a geometric resolution of up to 80 cm. The road data set contains about $5,697,000$ vertices (151 MB).
- A DEM of the Vorarlberg region in Austria, covering an area of about $4,760$ km^2 at a geometric resolution of 1 m. The road data set contains about $213,000$ vertices (7 MB).

The size of the terrain data including photo textures is around 1 TB per data set. We therefore use an out-of-core visualization system capable of handling arbitrarily large data sets.
The preprocessing step for the geometric approach (see section 6.1) increased the size of the road data by about a factor of ten in all tested cases. Note that for the shadow volume approach, this step is not required.

Performance. All performance measurements were taken at a display resolution of 1600×1200 on a PC with Windows Vista, a 2.66 GHz Intel Core 2 Quad CPU, 8 GB of RAM and an ATI Radeon HD 5870 GPU (driver version 10.6).
The graph in Fig. 13 shows the frame rate during a recorded flight over the medium-resolution DEM of Utah at an average speed of about 1750 m/s. When rendering geometric roads (GEO), the maximum (average) performance drop is about 30% (26%) compared to rendering the terrain without roads. The highest performance impact occurs over Salt Lake City (far right in the graph). This area contains a dense road network and only a small amount of terrain geometry, as buildings are not included in the height field. The additional rendering of rounded caps does not significantly influence the performance.
For shadow volume-based roads (SV), the maximum (average) performance drop is around 40% (35%) without and 55% (42%) with rounded caps. Breaking the numbers down to the sole rendering of roads, SV with caps is about 1.4 times as expensive as without caps.

The visual quality produced by both techniques is identical at most locations in Utah. Therefore, GEO is preferable because of its higher performance. Fig. 14

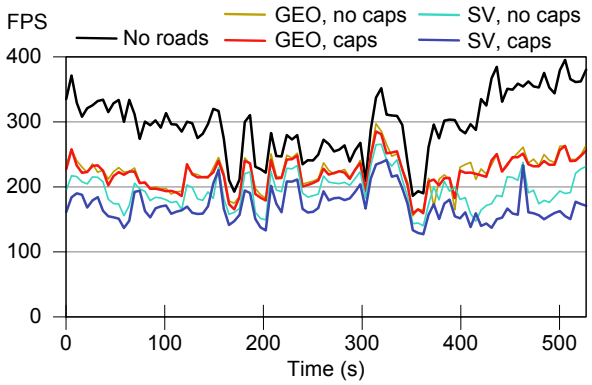


Figure 13: Performance - Utah

shows the frame rates during a flight over the high-resolution data set of Bavaria at an average speed of about 950 m/s. In this scenario, the performance of all approaches is very close; the average cost is between 33% and 43%. Even though GEO often requires many more triangles (up to about 3 million) than SV ($\leq$ 1M) because of the adaption to the terrain mesh (which itself uses up to about 35M triangles), GEO is still slightly faster. Thus, the GPU is more limited by shading computations than by the geometry throughput. However, GEO can often not provide an adequate mapping on such high-resolution terrain data (see Fig. 15). Therefore, SV is preferable for such fine-grained DEMs.

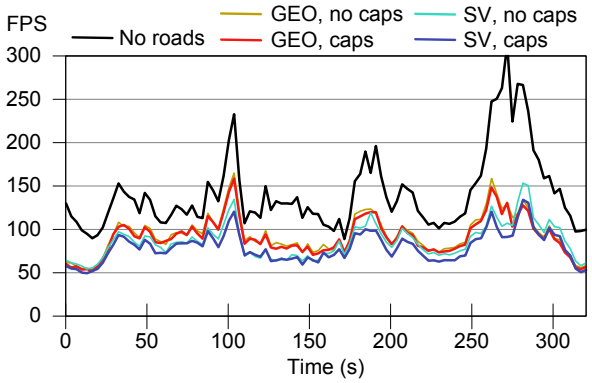
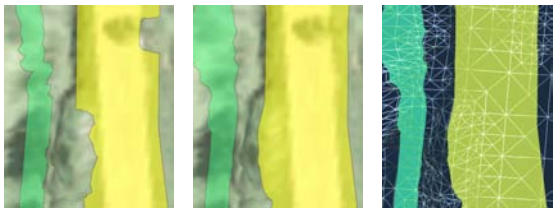


Figure 14: Performance - Bavaria



(a) Geometric (b) Shadow volume (c) Wire frame

Figure 15: Comparison of our draping algorithms on high-resolution terrain.

The Vorarlberg data set has a similar geometric resolution as the Bavaria data set, but the road network is much more sparse. Regardless of which algorithm is chosen for rendering roads, the highest performance impact amounts to only 15%. However, as in Bavaria, GEO can not provide adequate quality.

Matching. We should note that in many situations the vector data set did not exactly match the terrain data, i.e. there was a certain offset between the vector data roads and roads in the phototextures. These problems frequently occur in cities or forests, where even a slight offset causes a road to be projected onto a building or a tree. GEO fails to produce any reasonable results in this case (see Fig. 16(a)); SV produces a technically correct but not very useful projection (see Fig. 16(b)). This is a problem of the data rather than the draping algorithm. An additional preprocessing step could match the vector data to the terrain and its phototextures.

(a) Geometric (b) Shadow volume

Figure 16: Artifacts caused by a mismatch between terrain and vector data.

Comparison. Our method presents a marked improvement over several commercial GIS systems. For example, Google Earth 6.0 uses a simple geometric approach without adaption to the terrain and therefore does not achieve a correct projection of roads onto the DEM. It also does not provide correct road intersections and does not support multi-color roads or outlines. ArcGIS 10.0 rasterizes vector data into textures which are overlaid onto the terrain, similar to the orthophotos. This results in a correct projection and correct behavior at road intersections. However, a dynamic scaling of road widths is not possible, and multi-color roads or outlines are not supported.

8. CONCLUSION

In this paper, we have proposed and evaluated two approaches for rendering high-quality cartographic roads with rounded caps on high-resolution 3D terrain models. Both can be used on hardware platforms supporting Direct3D 10 or OpenGL 3.0. We have shown that a geometry-based approach provides high performance and good quality for low- to medium-resolution terrain data sets. However, it requires a moderately complex preprocessing step, and it can not provide an adequate visual quality with high-resolution terrain data. It is therefore a reasonable choice for low-end hardware, e.g. on mobile devices, where rendering of high-resolution terrain data is not feasible.

The shadow volume algorithm enables pixel-exact rendering of cartographic roads on 3D terrain. It is more expensive at runtime than the geometry-based approach; however, the rendering of high-resolution terrain remains the larger part. In low-resolution terrain data sets, on the other hand, its relative performance impact is large. The algorithm is easy to integrate into existing terrain rendering engines, as no adaption of roads to the terrain is required. It also extends naturally to polygonal vector data.

In further research, we plan to evaluate the use of tesselation shaders for the creation of geometric caps on Direct3D 11 or OpenGL 4.0 capable hardware.

9. ACKNOWLEDGEMENTS

The authors wish to thank the Landesvermessungsamt Feldkirch, Austria, the Landesamt für Vermessung und Geoinformation Bayern and the State of Utah for providing high-resolution geo data.
This publication is based on work supported by Award No. UK-C0020, made by King Abdullah University of Science and Technology (KAUST).

10. REFERENCES

[ARJ06] Agrawal, A., Radhakrishna, M., and Joshi, R. Geometry-based mapping and rendering of vector data over LOD phototextured 3D terrain models. In *Proceedings of WSCG*, pages 787–804, 2006.

[BN08] Bruneton, E. and Neyret, F. Real-time rendering and editing of vector-based terrains. In *Comput. Graph. Forum*, volume 27, pages 311–320, April 2008. Special Issue: Eurographics '08.

[Cro77] Crow, F. C. Shadow algorithms for computer graphics. *SIGGRAPH Comput. Graph.*, 11(2):242–248, 1977.

[DBH00] Döllner, J., Baumann, K., and Hinrichs, K. Texturing techniques for terrain visualization. In *VISUALIZATION '00: Proceedings of the 11th IEEE Visualization 2000 Conference (VIS 2000)*, Washington, DC, USA, 2000. IEEE Computer Society.

[Döl05] Döllner, J. Geovisualization and real-time 3d computer graphics. In E., Dykes, J., MacEachren, A., and Kraak, M., editors, *Exploring Geovisualization*, chapter 16, pages 325–343. Pergamon, 2005.

[DSW09] Dick, C., Schneider, J., and Westermann, R. Efficient geometry compression for GPU-based decoding in realtime terrain rendering. *Computer Graphics Forum*, 28(1):67–83, 2009.

[Gum03] Gumhold, S. Splatting illuminated ellipsoids with depth correction. In *Proceedings of 8th International Fall Workshop on Vision, Modelling and Visualization*, volume 2003, pages 245–252, 2003.

[Hei91] Heidmann, T. Real shadows, real time. *IRIS Universe*, 18:28–31, 1991.

[KD02] Kersting, O. and Döllner, J. Interactive 3d visualization of vector data in GIS. In *GIS '02: Proceedings of the 10th ACM international symposium on Advances in geographic information systems*, pages 107–112, New York, NY, USA, 2002. ACM.

[Kra01] Kraak, M. *Cartographic principles*. CRC, 2001.

[LKR+96] Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L. F., Faust, N., and Turner, G. A. Real-time, continuous level of detail rendering of height fields. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 109–118, New York, NY, USA, 1996. ACM.

[Ope10] OpenStreetMap. OpenStreetMap website, 2010.

[PG07] Pajarola, R. and Gobbetti, E. Survey of semi-regular multiresolution models for interactive terrain rendering. *Vis. Comput.*, 23(8):583–605, 2007.

[Phy09] Physical Storage Format Initiative. *Navigation Data Standard: Compiler Interoperability Specification*, 2009.

[RBE+06] Reina, G., Bidmon, K., Enders, F., Hastreiter, P., and Ertl, T. GPU-Based Hyperstreamlines for Diffusion Tensor Imaging. In *Proceedings of EUROGRAPHICS - IEEE VGTC Symposium on Visualization 2006*, pages 35–42, 2006.

[RMM95] Robinson, A., Morrison, J., and Muehrcke, P. *Elements of cartography*. John Wiley & Sons Inc, 1995.

[SD02] Stamminger, M. and Drettakis, G. Perspective shadow maps. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 557–562, New York, NY, USA, 2002. ACM.

[SGK05] Schneider, M., Guthe, M., and Klein, R. Real-time rendering of complex vector data on 3d terrain models. In Thwaites, H., editor, *The 11th International Conference on Virtual Systems and Multimedia (VSMM2005)*, pages 573–582. ARCHAEOLINGUA, October 2005.

[SK07] Schneider, M. and Klein, R. Efficient and accurate rendering of vector data on virtual landscapes. *Journal of WSCG*, 15(1-3), January 2007.

[Tal96] Talvitie, A. *Functional Classification of Roads*. Transportation Research Board, Washington, D.C., 1996.

[WKW+03] Wartell, Z., Kang, E., Wasilewski, T., Ribarsky, W., and Faust, N. Rendering vector data over global, multi-resolution 3d terrain. In *VISSYM '03: Proceedings of the symposium on Data visualisation 2003*, pages 213–222, Aire-la-Ville, Switzerland, 2003. Eurographics Association.

Pipeline Reconstruction from Fisheye Images

Yuhang Zhang
The Australian National University
yuhang.zhang@anu.edu.au

Richard Hartley
The Australian National University
richard.hartley@anu.edu.au

John Mashford
CSIRO
Australia
john.mashford@csiro.au

Lei Wang
The Australian National University
lei.wang@anu.edu.au

Stewart Burn
CSIRO
Australia
stewart.burn@csiro.au

ABSTRACT

Automatic inspection of pipelines has great potential to increase the efficiency and objectivity of pipeline condition assessment. 3-D pipeline reconstruction aims to reveal the deformation of the pipe surface caused by internal or external influences. We present a system which can reconstruct the inner surface of buried pipelines from multiple fisheye images captured inside the pipes. Whereas the pipelines are huge, a fatal defect can be as tiny as a fine crack. Therefore a reliable system demands both efficiency and accuracy. The repetitive patterns on the pipe surface and the poor illumination condition during photographing further increase the difficulty of the reconstruction. We combine several successful methods found in the literature as well as new methods proposed by ourselves. The proposed system can reconstruct pipe surface not only accurately but also quickly. Experiments have been carried out on real pipe images and show promising performance.

Keywords: 3D reconstruction, surface reconstruction, fisheye lens, water pipelines, pipe inspection, image processing.

1 INTRODUCTION

Water pipelines are indispensable facilities of modern urban systems. After serving for decades underground, the condition of the pipelines deteriorates to varying degrees. Timely inspection and repair is therefore required to prevent imminent collapse. Traditionally pipe inspection involves intensive manual effort. Manual image interpretation is an expensive process for which wrong decisions caused by fatigue and subjective bias are inevitable. Hence a computer-aided inspection system is of great value.

We present a system which can reconstruct the inner surface of buried water pipes based on a sequence of images captured inside the pipes (Figure 1). Deformation of the pipe surface which foreshadows the pipeline collapse can then be detected from the reconstructed model. Early work on similar applications relied on range cameras such as laser scanners, which is expensive. Later, due to the developments of computer vision, methods solely based on 2D images were proposed [3, 8, 9]. However, because of the limitation in computer vision at the time and the difficulty in this particular application, some of these works made restrictive assumptions such as that, the pipes are built from bricks which provide distinctive patterns; and the others terminate with reconstructing a small group of isolated points only. Some 3D reconstruction applications of general scenes [18] bear the same limitation as well. More recently, several large-scale 3D reconstruction applications of general scenes have been proposed [1, 20], which can reconstruct millions of points in a relatively short time. However, their implementation requires high-end parallel computers.

What distinguish the proposed system from all the previous ones are:

1. we intensively reconstruct the pipe surface, which is composed of millions of points, rather than a group of selected points;
2. our algorithm is fast and can be implemented on normal PCs;
3. we have proposed a number of specific mechanisms to increase the robustness of the system, so that it can work with pipe surface without distinctive patterns under poor illumination conditions.

We first give an overview of the reconstruction problem, as well as our method. When discussing each step in detail, experimental results will be provided accordingly. As will be seen, our method performs not only accurately but also quickly.

2 OVERVIEW

We make no particular assumption about the material of the pipelines. Actually, the pipes a civil engineer frequently confronts are made from concrete which gives

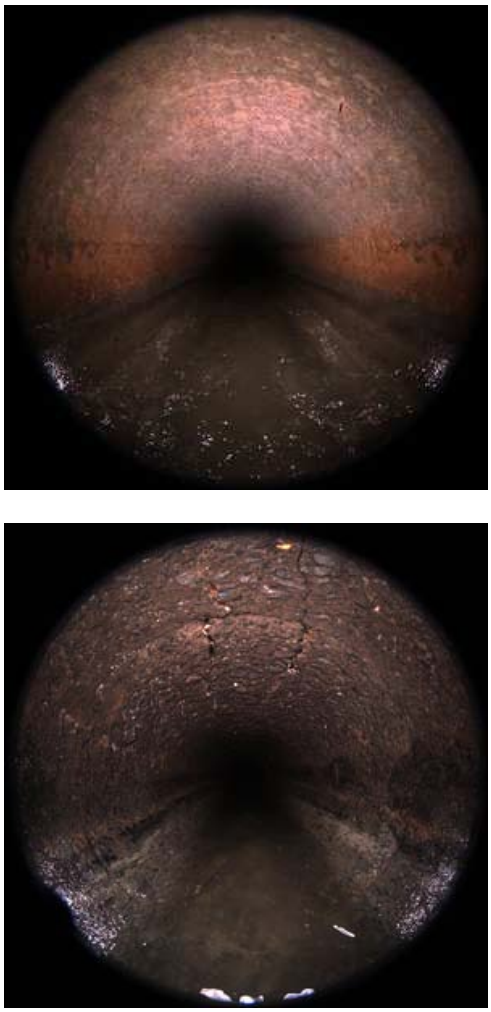


Figure 1: fisheye images captured inside of the pipelines. The pipe on the top is in relatively good condition, whereas the one on the bottom is in poor condition.

little reliable texture. We will therefore make our system capable of handling pipes of this type.

To assess the pipeline condition, images are collected by a mobile camera travelling through the pipelines. To capture more details from the pipe surface, the mobile camera is equipped with a wide view angle fisheye lens rather than an ordinary perspective lens. During photographing the illumination is provided by a light fixed to the camera, which can only illuminate a limited range in the pipe unevenly. Figure 1 shows two example images captured in different pipelines. As we can see, only the peripheral regions of the images contain clear pipe surface. The texture on the pipe surface is fine and weak. In the same pipe the surface appears to be similar everywhere. A sequence of images is captured as the camera moves. Two adjacent images share overlapping regions.

Our reconstruction follows a four-step paradigm.

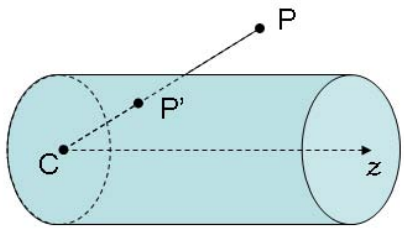


Figure 2: image cylinder: **C** is the camera center; *z* axis is the central axis of this image cylinder; **P** is an object point, with **P'** as its image.

1. Firstly, initial point matching is established between overlapping images;
2. Matched points will then be utilized to estimate the relative pose of calibrated cameras corresponding to different views.
3. With the obtained camera parameters, we implement dense matching between overlapping images while enforcing the epipolar constraints. This step was never included in the previous works [3, 8, 9], in which only those matched points detected in the first step were reconstructed. This step is also arguably the most sophisticated and time consuming step in the whole algorithm. Handling it efficiently is our major contribution.
4. Finally, the 3D location of each point in the image is densely determined through triangulation and a 3D model is built up.

As computer vision algorithms about perspective cameras have already been well studied [4], one might transform each fisheye image to a perspective image to simplify the subsequent process [8, 9]. However, such a transformation either produces an extremely large perspective view which significantly upsampled the peripheral region in the original image, or produces a perspective view of proper size but at the cost of cropping off the peripheral region. In either case, we will destroy the region which really contains the important information in the original image. Therefore, in our work we choose to process the images in their original form, or transform them, when necessary, onto an image cylinder (Figure 2) instead of an image plane.

We define an image cylinder by specifying its central axis. Its radius can be deliberately set to unity without affecting its functionality. The central axis of the image cylinder can be the optical axis of the camera or the baseline between two cameras, depending on the circumstances. The cylindrical image of each point in the 3D world is generated by the intersection of the image cylinder and the ray going from the point to the camera center. Each parallel line on the cylindrical surface

functions like a perspective camera by itself, however, altogether they receive an omnidirectional image more readily than a normal perspective camera does. This image cylinder is particularly useful during point matching and depth estimation, as will become clear soon.

In the remaining part of this paper, we discuss each step mentioned above in detail.

3 INITIAL POINT MATCHING

Due to the development of local invariant features [13, 16], finding corresponding points between overlapping images is much easier now than ever. Comprehensive surveys into the feature detectors and descriptors can be found in [14, 15]. However, point matching on a pipe surface is still difficult due to the faint and similar patterns everywhere. Moreover, whereas all the proposed local invariant features are approximately invariant under affine transformations, the transformation conducted by a fisheye lens is not even perspective, but nonlinear. Thus the corresponding points identified by local invariant features on pipe surface contain many false matches. Our experiments show that the number of false matches can easily exceed the number of true matches by an order of magnitude.

To improve the situation, besides enforcing loose geometry constraints, we transform each fisheye image onto the image cylinder we discussed in Section 2. The image cylinder here takes the optical axis of the camera as its central axis. The consequential advantage of such a transformation is obvious. Since the optical axis of all cameras are roughly parallel to each other as well as to the central axis of the pipe, the images generated on different image cylinders only differ from each other approximately by a simple translation. Comparing to the original fisheye images, we not only remove the scale difference between corresponding regions in different images, but also largely rectify the distortion caused by the nonlinear projection through a fisheye lens. Hence the corresponding points found by local invariant features on the cylindrical images are more reliable. Geometry consistency is also easier to enforce on the transformed images. All line segments connecting corresponding points in two cylindrical images should be roughly parallel and of almost the same length. After detecting corresponding points in the transformed images, we can easily back-project them onto the original fisheye images to facilitate camera pose estimation.

Figure 3 shows the matching results on the original images and the transformed images respectively. Particularly, Hessian-affine detector [16] and SIFT descriptor [13] are used for feature extraction. Matches are identified if two SIFT features share a Euclidean distance under a predefined threshold. Although point matching is between two images, we only present one of them here for clear presentation. We plot the matched points from two images onto one image and connect each pair with a yellow line segment. As we can expect, the lines in the original image should all roughly point to the image center, whereas those in the transformed images should all be roughly horizontal. We only present those matches that can be verified with these loose geometry constraints in Figure 3. On the original image 239 matches passed the verification, whereas on the transformed image 563 matches passed the verification. That justifies our earlier discussion that matching on the transformed images is more reliable.

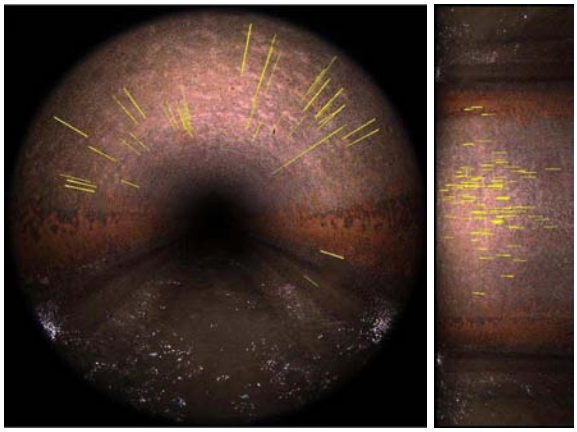


Figure 3: matches found on the image pair of original form and transformed form respectively. Only those matches that pass the loose geometry verification are presented. The rejected false matches are thousands in number. They happen so frequently because the pipe surface is similar everywhere.

Intuitively the matches from both cases are more than sufficient to implement subsequent estimation. One might therefore suspect the necessity of the cylindrical transform. However, as we can see, the lines presented in the image do not seem to match the numbers given above. That is because more than one match can happen intensively on neighbor pixels. Considering matches at the same location does not increase the estimation accuracy, more matches than sufficient is in fact necessary. The number of qualified matches also depends on the texture on the pipe surface. On some smooth surfaces, the number of matches will be much smaller as fewer local features can be detected. That is when the image transformation becomes more important.

Some false matches still remain in Figure 3 as their line segments are not of reasonable length. Again, it is more convenient to enforce this constraint on the transformed images rather than on the original images. On the transformed images, the length of all lines segments should be roughly equal. In the original images, their length should not be equal due to the nonlinearity of the fisheye lens, which is difficult to use as a loose constraint.

4 CAMERA POSE ESTIMATION

A calibration method for fisheye cameras can be found in [10]. Here we assume the camera is calibrated and only aim to estimate the external parameters of the camera. We have briefly discussed the reason why we do not transform the original image into perspective view in Section 2. Particularly on camera pose estimation, the nonlinear transformation between a fisheye view and a perspective view might significantly enlarge the matching error from one pixel to hundreds of pixels in the peripheral region of the image. Hence we need a pose estimation scheme that can be applied directly to fisheye images and is efficient.

We use a modified version of the direct estimation method initially designed for perspective cameras [6]. The main result of the original method is that, given a close enough initialization of the camera parameters as well as the point locations, a structure from motion problem can be solved directly using some iterative optimizing algorithms, e.g. the Levenberg-Marquardt algorithm [12]. The advantage of this method lies in the fact that it is one-stop. It requires no sophisticated operation on any interim variables, e.g. the fundamental matrix required in [5] or the measurement matrix required in [21]. The disadvantage of this algorithm is the requirement of a close initialization, which is usually impossible, especially when the number of unknown parameters is huge.

We discover that the advantage and the disadvantage of the direct estimation method can be magnified and reduced respectively in our problem. In particular, unlike the other algorithms of structure from motion applications, this algorithm bears no assumption on the camera model, neither perspective nor affine. That means it can be adapted to fisheye camera as well, as long as we change the cost function in the Levenberg-Marquardt minimization from the perspective projection to the fisheye projection. Furthermore, as we know the normal condition of the pipelines as well as the approximate location of the camera with respect to the pipe, we can initialize all the parameters accordingly. Obviously, many other inspection purposed applications share the same convenience.

Another important fact about the parameter initialization is that, the parameters are not independent. More precisely, from the parameters of two random cameras, we can accurately determine the 3D locations of all the matched points captured by the two cameras through triangulation. This observation largely reduces the number of variables we need to initialize, i.e. we only initialize the camera parameters, and then derive the location of the points. Besides dependence, obviously, there is also independence between different parameters. Whereas millions of points were captured from thousands of different locations, the camera pose for each image is only related to the dozens of points that have appeared in its image. The location of each point is only affected by the few cameras capturing it. This observation not only leads to the simplification within Levenberg-Marquardt optimization, i.e. the sparse-Levenberg-Marquardt [6], but also to the simplification of our reconstruction. We firstly estimate the camera parameters and points location locally between each pair of adjacent images with the direct estimation method. Although this estimation is local, it has already considered most of the information relevant to the two cameras. Hence the output should still be quite accurate. We then transform all the estimated points and cameras into the same frame of reference. That gives us the initialization of a global direct estimation. Indeed, when the global consistency is not compulsory, we can even terminate without a global estimation. Later we will see, at least for the purpose of pipe condition assessment, local estimation can already detect deformation and cracks on the pipe surface.

The error to be minimized with Levenberg-Marquardt algorithm is given by (1), where $\hat{x}_{ij}$ is the coordinates of point i observed in image j, and x_{ij} is the estimated coordinates of the corresponding point in the corresponding image. When $\hat{x}_{ij}$ is unknown, which really means point i is not observed in image j, we set $\hat{x}_{ij} = x_{ij}$, so that their difference is 0 and the total error will not be affected. During local estimation, as the numbers of points and cameras are limited, the sparse-Levenberg-Marquardt algorithm converges quickly. In our experiment, it takes about 0.5 seconds to estimate the relative pose between each pair of cameras, when 200 point matches are involved. The root mean square error is around one pixel upon converging.

$$e = \sum_j \sum_i ||\hat{x}_{ij} - x_{ij}||^2 \tag{1}$$

Further more, we might add the intrinsic camera parameters into the local estimation. That converts our problem to an uncalibrated reconstruction, requiring inputting three images each time. We do not recommend estimation based on three views. That is because the number of matching points that can survive three views are usually too small to facilitate reliable estimation.

5 INTENSIVE MATCHING

Whereas reconstructing a set of isolated points is sufficient to reveal the pipe deformation on large scale, intensive points reconstruction is required to reveal those cracks which are only several pixels wide on images. To intensively reconstruct the pipe surface, we need intensively match the points on the pipe surface.

Implementing intensive stereo matching between overlapping images is by nature a difficult problem, even though we can narrow the matching range using

the epipolar constraint. A good review of relevant algorithms can be found in [19]. The state of the art of intensive stereo matching lies in the α-expansion method proposed in [22], which approaches the problem by way of optimizing a multi-label Markov Random Field (MRF). However, when the size of the image is huge, optimizing a corresponding MRF requires heavy computation. Another method called FastPD [11] is faster but requires much more memory. More recently, a hierarchical mechanism is incorporated into MRF optimization [23], enabling optimizing large MRFs more efficiently with low memory occupancy.

However, due to the following reasons, our problem cannot be solved by these off-the-shelf methods. Firstly, since the light source is carried by the moving camera, corresponding points in different images are captured under significantly different illuminations, which obviously makes the matching tougher. Secondly, even the hierarchical mechanism [23] largely boosts the speed of solving an individual problem, intensively matching a large number of images is still a huge task. Therefore, we propose two mechanisms to improve the situation.

5.1 Illumination Regularization

Some illumination invariant description and comparison methods have been proposed in the literature, such as the Normalized Cross-correlation (NCC) and the SIFT descriptor [13]. They non-exclusively require more complex computation, which will significantly slow down the system. Here instead of using illumination invariant description, we make the illumination invariant.

Although the light source moves during photographing, its relative position to the camera center is fixed and the location of the camera center within each cross-section of the pipe is in general stable. That suggests, the pipe surface captured by the pixels on the same location within every image is illuminated by approximately the same light. From each pipeline, we have collected thousands of images. The average grey level of a pixel on the same location over thousands of images can be then regarded as the illumination intensity of this pixel or its corresponding points on the pipe surface.

Figure 4 shows the average illumination intensity on images captured in the two pipelines. They are different because the camera travelled at different height in the two pipes and the deterioration degree of the two pipes are different. Based on the illumination intensity images in Figure 4, we can regularize the illumination within each image through (2), where $I(i)$ is the pixel value of pixel i in the original image, $G(i)$ is the grey level of pixel i in the illumination intensity image, a is a positive constant controlling the brightness in the regularized image. Figure 5 compares the image before and after illumination regularization. Especially on the regularized cylindrical images, the obvious illumination variance is removed leaving all pixels under comparable illumination. After illumination regularization, we can easily measure the similarity between pixels by the absolute difference between their regularized pixel values.

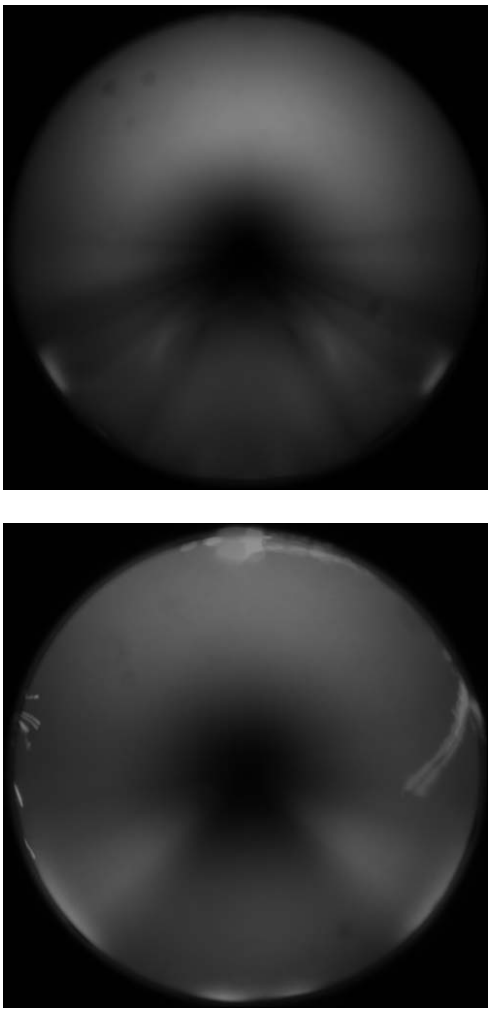


Figure 4: the average illumination intensity obtained from images of two pipelines: some dark blobs can be observed on the top image, which were caused by water drops spread onto the lens; the white threads on the bottom image are caused by some rubbish attached to the lens. However, their affect to the matching process is ignorable.

$$I_r(i) = \frac{aI(i)}{G(i)} \tag{2}$$

5.2 Sequential MRF Optimization

We first explain the design of α-expansion as well as its hierarchical version in our problem and then intro-

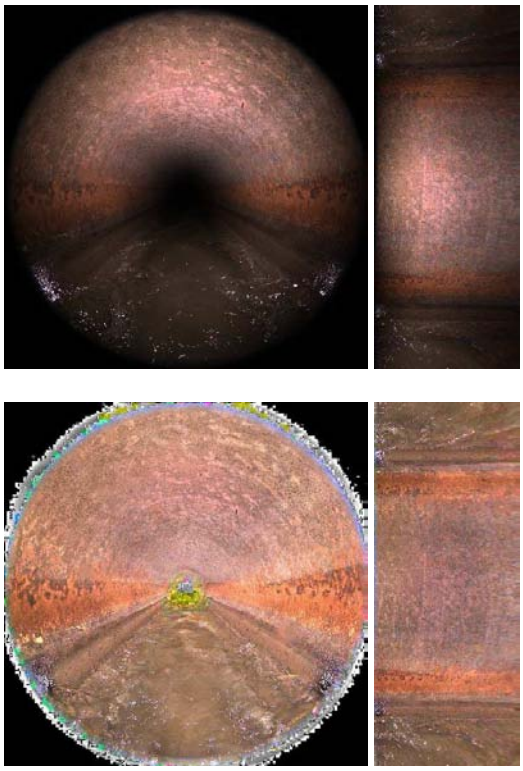


Figure 5: images before and after illumination regularization.

duce our sequential mechanism, which further boosts the speed of our system.

α-expansion After rectification [4], the corresponding points between two overlapping images all lie in corresponding scan lines. One of the two images will later be referred to as the reference. Localizing corresponding points along a scan line, namely estimating the disparity of each object point within the two images can be modelled as estimating the variables in a second order Markov Random Field.

In particular, each variable in the MRF corresponds to a pixel in the reference image. The value of each variable corresponds to the disparity of its corresponding pixel. The probability for each variable to have a particular value, or equivalently for a pixel to have a particular disparity, is subject to two factors. The first one, a function of the color difference between the two pixels related by this particular disparity, is usually referred to as the unary term or the data term. In our work, we use the following unary term:

$$U_i = ||I_1(i) - I_2(i')||_1 \tag{3}$$

where $I_1(i)$ is the color of pixel i in the reference image, $I_2(i')$ is the color of the pixel related to i by its current disparity in the other image, and U_i is computed as a L_1-norm difference between the two. The other one, a function of the disparity difference between the pixel and its neighbor, is usually referred to as the binary term or the smooth term. Each pixel usually has four neighbors, hence there are four binary terms. Binary terms are used to enforce the smooth constraint, i.e. the disparity of points in a scene should be smooth almost everywhere [17]. In our work, we use the following binary term:

$$B_{ij} = |L(i) - L(j)| \tag{4}$$

where $L(i)$ and $L(j)$ is the disparity of two neighbor pixel i and j in the reference image, and B_{ij} is computed as their absolute difference.

The unary term and the binary term really play the role of likelihood and prior in the Bayesian theory. Therefore, through maximizing the probability of a MRF, one really globally maximizes the posterior of each variable and obtains the most probable disparity of each pixel. Due to the Hammersley-Clifford theorem, maximizing the joint probability of variables in the above MRF is equivalent to minimizing the following cost-function:

$$E = \sum_i U_i + \lambda \sum_{ij} B_{ij} \tag{5}$$

where λ is a positive constant balancing the weight between the unary term and the binary term. An effective way of perceiving (5) is through constructing a weighted graph. As shown by Figure 6, each vertex in the graph corresponds to a pixel in the reference image or a disparity value. Edges are created between each disparity vertex and all the pixel vertices. Each edge of this type can be represented by a term U_i in (3). Pixel vertices which are neighbors in the image are connected by edges as well. Each edge of this type corresponds to a B_{ij} in (4). Then, minimizing (5) is equivalent to finding the minimal cut on its graph after which each subgraph contains one and only one disparity vertex.

If the graph contains only 2 disparity vertices, the minimal cut can be found using the max-flow algorithm, regarding the two disparity vertices as the source and the sink respectively. When the number of disparity vertices is larger than two, minimizing (5) is in general NP-hard [2]. α-expansion can provide a high quality suboptimal solution in polynomial time.

Starting from a random initial state, α-expansion sequentially examines the applicability of each disparity, represented by α, to all the pixels. In particular, for each α, a new graph is created. In the new graph, the source node corresponds to the current disparity of each pixel; the sink node corresponds to the α disparity. Those pixels, whose current disparity is α are not included into the new graph. A bi-cut is then implemented using max-flow algorithm to determine whether the pixels currently having other disparities should change their disparities to α. After each

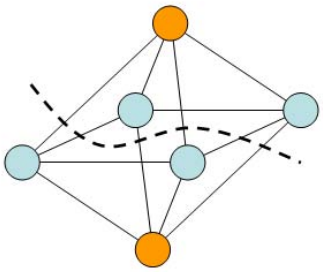


Figure 6: a graphic explanation of minimizing (5): the four blue vertices each correspond to a pixel in the reference image; the two orange vertices correspond to two possible disparities respectively; minimizing (5) is equivalent to a minimal cut to the graph after which each subgraph contains one and only one disparity vertex. The dashed line in the figure shows a possible cut.

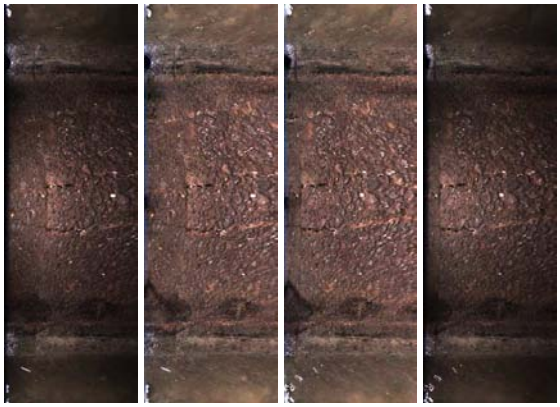


Figure 7: two adjacent images mapped onto the same image cylinder. Images before and after illumination regularization are both provided for comparison.

round of bi-cut, only the subgraph containing α vertex can be increased. That is why the algorithm is named as α-expansion. To compensate the loss in optimality, multiple outer iterations are usually implemented.

Denote the number of outer iterations as m, the number of disparity vertices as n, the processing time of max-flow algorithm as f. The processing time of α-expansion is mnf. When running on images of small size, *e.g.* 300×300, α-expansion can usually terminate quickly in 20 seconds on a normal PC. However, when dealing with a pair of images in large size, whose disparity range is usually large as well, the max-flow algorithm needs to be implemented on a huge graph many times. The processing time of α-expansion expands significantly. Our experiments show that when dealing with a stereo pair in the size of 1000×1000, α-expansion needs more than 30 minutes to converge. That is by definition too slow for practical use. The alternative method, FastPD, cannot be applied either, because a normal PC cannot provide sufficient memory space.

Hierarchical α-expansion The idea of hierarchical α-expansion can be explained as solving the problem with α-expansion under a low resolution first, and then fine tuning the low resolution solution onto higher resolution through optimizing another MRF. More details can be found in [23]. As these two steps can be implemented recursively, the original problem is really solved in a coarse-to-fine manner. Besides, since the MRFs being optimized in the two steps are both much smaller than the original one, the processing speed is largely improved. With the hierarchical α-expansion, processing a stereo pair in the size of 1000×1000 requires only around 10 seconds on a normal PC, and the optimality is comparable to the original α-expansion.

Figure 7 shows two sample images on which we have implemented hierarchical α-expansion. This time, the central axis of the image cylinder is the baseline connecting the two camera centers. As we have already obtained the external parameters of the cameras, we can accurately generate the cylindrical image through back-projection. Although this image cylinder is different from an image plane in shape, it can parallelize the epipolar lines as well. It takes minor effort to snip and unwind that cylindrical image into a planar image. Just make sure to snip the two cylinders along the same epipolar line. So we obtain an image pair in the form people usually deal with during intensive matching, namely corresponding points always lie on the same scan line. The pipe surface presented in these two images contains a vertical connection line and two horizontal narrow cracks, which will test our algorithm's capability in detecting small defects on the pipe surface. We crop off the region submerged by water before implementing graph cuts. That is because we are only interested in the pipe surface, and that dropping the water region can help saving processing time. Figure 8 shows the interim and final results of the hierarchical graph cuts. We can see how the final disparity map is reached through a coarse-to-fine procedure. The disparity value is larger in the center of the image, which corresponds to the top region in the pipe. That suggests that the camera is closer to the top of the pipe compared to the left and right sides of the pipe. The vertical connection line and the horizontal cracks can be clearly observed in the final result as well.

Sequential α-expansion To further boost the processing speed, we propose a sequential mechanism in MRF optimization, the key idea of which lies in better label initializations and smaller label range. The time cost by the max-flow algorithm which is a subroutine in α-expansion depends on the flows needed to be pushed before reaching the optimal state. The number of necessary flows depends on the initial state of the network. That really suggests, if the initial state of the network is

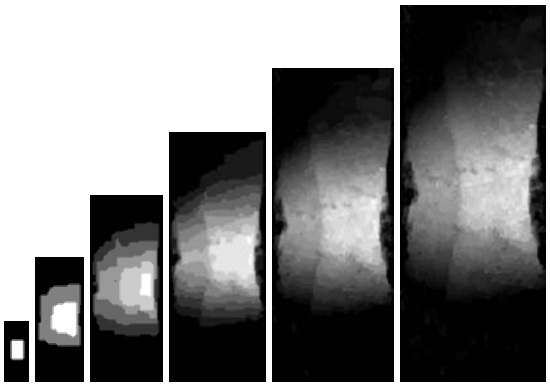


Figure 8: the interim and final results of the hierarchical graph cuts.

more similar to the optimal state, fewer flow, and hence less time, will be needed in optimization. Moreover, starting from an initial state close to the optimal also reduces the number of outer iterations in the α-expansion algorithm. A smaller label range will reduce the number of max-flow implementations in a single iteration.

Whereas for a contextless image pair one can only initialize all labels to be zero or arbitrary values, for sequential pipe images in our case we can largely predict the label configuration. The disparity of each point on the pipe surface is determined by two factors: firstly, its deterioration degree; secondly, and more importantly the location of the camera center. If the camera center travels along the central axis of the pipe, the disparity of different points will only differ slightly due to deterioration. However, if the camera center travels along some line far away from the central axis, the disparity of different points on the pipe will vary significantly. Although the deterioration degree of different regions on the pipe surface is arbitrary, the location of the camera center within each cross section of the pipe is generally stable. Therefore, we only use a large label set during the intensive matching for the first few image pairs. We can then acquire the relative location of the camera within the cross-section of the pipe, or more directly the average disparity along each scan line in the image. On subsequent image pairs, the pixels on each scan line are initialized with the corresponding average disparity. A smaller label set will then be used to estimate their disparities accurately. The smaller label set only needs to cover the variety caused by deterioration, which will be significantly narrower than that caused by the camera location. The MRF optimizing speed is hence boosted.

6 BUILDING A 3D MODEL

Through dense matching on the image cylinder, we have acquired the depth information related to each pixel on the cylindrical image. Together with the camera parameters estimated earlier, we can easily determine the 3D location of each point on the pipe surface through triangulation. The scale ambiguity is removed by setting the length of the baseline between two camera centers as unity. From each pair of adjacent images, we can obtain several millions of isolated 3D points. For better visualization, we might reconstruct a continuous surface with these isolated points using the algorithm proposed in [7]. However, a model containing millions of independent points is too huge for a normal PC to render.

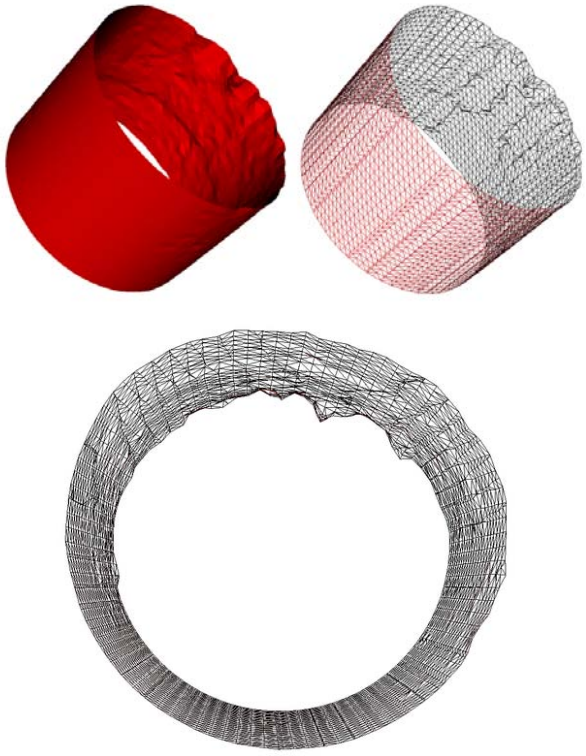


Figure 9: reconstructed pipe surface from the point cloud together with its triangulation state.

Figure 9 only shows the surface reconstructed from one hundredth of all the points. However, even after this significant downsampling, the connection line is still clearly presented, so is the pipe deformation on the large scale. The two cracks are missing because they are both less then ten pixels wide, which can not be preserved during this one hundredth downsampling. However, their existence and state have been represented by the point cloud containing millions of independent points, which will be assessed by civil engineers during force analysis. Note that we can only reconstruct the pipe surface above the water. We observe a complete cylinder here because the missing part has been manually complemented with ideal cylindrical surface.

7 CONCLUSION

We successfully reconstruct the inner surface of buried pipelines from a sequence of fisheye images. The obtained point cloud can be used to generate a virtual surface for visualization, as well as to facilitate other algorithms for pipe condition analysis. We used various efficient and reliable schemes over the four-step reconstruction. We paid particular attention to the process of intensive matching, which is generally slow and memory demanding based on previous algorithms. Our new method overcomes the obstacle of illumination variance and largely boosts the speed. More improvement on 3D model generation is still necessary. One possible development lies in automatically detecting regions of interest and unevenly downsampling the point cloud accordingly. This will be a direction of future work.

ACKNOWLEDGEMENTS

This work was supported by CSIRO, Water for a Healthy Country Flagship.

REFERENCES

[1] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building rome in a day. In *ICCV*, pages 72 –79, 2009.

[2] Endre Boros and Peter L. Hammer. Pseudo-boolean optimization. *Discrete Appl. Math.*, 123:155–225, November 2002.

[3] D. Cooper, T.P. Pridmore, and N. Taylor. Towards the recovery of extrinsic camera parameters from video records of sewer surveys. In *MVA*, pages 53–63, 1998.

[4] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, March 2004.

[5] Richard I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *ECCV '92: Proceedings of the Second European Conference on Computer Vision*, pages 579–587, London, UK, 1992. Springer-Verlag.

[6] Richard I. Hartley. Euclidean reconstruction from uncalibrated views. In *Applications of Invariance in Computer Vision*, pages 237–256, 1993.

[7] Hugues Hoppe. *PhD Thesis: Surface Reconstruction from Unorgnized Points*. 1994.

[8] J.H. Kannala, S.S. Brandt, and J. Heikkila. Measuring and modelling sewer pipes from video. In *MVA*, volume 19, pages 73–83, March 2008.

[9] Juho Kannala and Sami S. Brandt. Measuring the shape of sewer pipes from video. In *MVA*, pages 237–240, 2005.

[10] Juho Kannala and Sami S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fisheye lenses. *PAMI*, 28(8):1335–1340, 2006.

[11] N. Komodakis and G. Tziritas. Approximate labeling via graph cuts based on linear programming. *PAMI*, 29(8):1436–1453, August 2007.

[12] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathmatics*, II(2):164–168, 1944.

[13] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[14] K Mikolajczyk and C Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, October 2005.

[15] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *Int. J. Comput. Vision*, 65:43–72, November 2005.

[16] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *IJCV*, 60(1):63–86, 2004.

[17] Tomaso Poggio, Vincent Torre, and Christof Koch. Computational vision and regularization theory. *Nature*, 317:314–319, September 1985.

[18] Radka Pospíšilová. Occlusion detection and surface completion in 3d reconstruction of man-made environments. In *WSCG*, 2007.

[19] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47:7–42, April 2002.

[20] N. Snavely, S.M. Seitz, and R. Szeliski. Skeletal graphs for efficient structure from motion. In *CVPR*, pages 1 –8, 2008.

[21] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *Int. J. Comput. Vision*, 9(2):137–154, 1992.

[22] R. Zabih, O. Veksler, and Y.Y. Boykov. Fast approximate energy minimization via graph cuts. In *ICCV*, pages 377–384, 1999.

[23] Yuhang Zhang, Richard Hartley, and Lei Wang. Fast multi-labelling for stereo matching. In *ECCV 2010*, volume 6313, pages 524–537. Springer, 2010.

Deducing Explicit from Implicit Visibility for Global Illumination with Antiradiance

Gregor Mückl
University of Stuttgart
Allmandring 19
70569 Stuttgart, Germany
Gregor.Mueckl@visus.uni-stuttgart.de

Carsten Dachsbacher
Karlsruhe Institute of Technology
Am Fasanengarten 5
76131 Karlsruhe, Germany
dachsbacher@kit.edu

Abstract

The antiradiance method, a variant of radiosity, allows the computation of global illumination solutions without determining visibility between surface patches explicitly, unlike the original radiosity method. However, this method creates excessively many links between patches, since virtually all elements exchange positive and negative energy whose interplay replaces the visibility tests. In this paper we study how and if explicit visibility information can be recovered only by analyzing the link mesh to identify chains of links whose light transport cancels out. We describe heuristics to reduce the number of links by extracting visibility information, still without resorting to explicit visibility tests, e.g. using ray casting, and use that in combination with the remaining implicit visibility information for rendering. Further, to prevent the link mesh from growing excessively in large scenes in the beginning, we also propose a simple means to let graphic artists define blocking planes as a way to support our algorithm with coarse explicit visibility information. Lastly, we propose a simple yet efficient image-space approach for displaying radiosity solutions without any meshing for interpolation.

Keywords: global illumination, radiosity, antiradiance

1 INTRODUCTION

In the 1990s, radiosity methods have been significantly improved, but after a period of large interest research essentially ceased for several years. Mainly three difficulties with radiosity caused the degrading interest: meshing of the input geometry and computing the (hierarchical) link mesh, the necessity of storing all patches and radiosity values in memory for computing a view-independent solution, and above all, the expensive visibility computation that typically consumes most of the computation time [8]. However, some of these problems have recently been successfully tackled: the antiradiance method reformulates the rendering equation such that visibility computation for form factors is no longer necessary, and by this enables a simple and fast GPU implementation of radiosity methods. Dong et al. [4] also demonstrated a GPU-radiosity algorithm by coarsely discretizing visibility that can be computed without ray casting. Motivated by this progress, Meyer et al. [15] introduced a data-parallel method for meshing and hierarchical linking, and demonstrate a CUDA implementation. In combination, these methods allow for interactive radiosity in dynamic scenes. Fig. 1 illustrates the fundamental differences of traditional radiosity, and the two aforementioned improvements. While Dong et al.'s [4] method produces small link meshes – actually smaller than traditional radiosity – it affects the global illumination result negatively due to the coarse discretization. In this paper, we focus our study on the antiradiance method which matches traditional radiosity in terms of quality. However, it replaces costly visibility computation for form factors by creating excessively many links to transport negative energy to compensate for missing occlusion (see Fig. 2).

That is, two solutions at the opposite ends of the spectrum exist for handling visibility in the radiosity method: either fully explicit or fully implicit. This paper bridges the gap in between by presenting a method to deduce explicit visibility information from the link mesh that is generated for the antiradiance (AR) method. By this we can reduce the number of links, however, still without computing form factors with explicit visibility, e.g. using ray casting. Obviously, the visibility information of a scene must be encoded in the AR link mesh: otherwise it would not be possible to compute the correct global illumination result with implicit visibility. Our method starts at exactly this point: we analyze the link mesh to identify chains of links whose light transport cancels out. Once such a chain has been identified, we can remove it without changing the result or reducing the rendering quality noticeably. However, reducing the number of links saves memory and computation time during light prop-

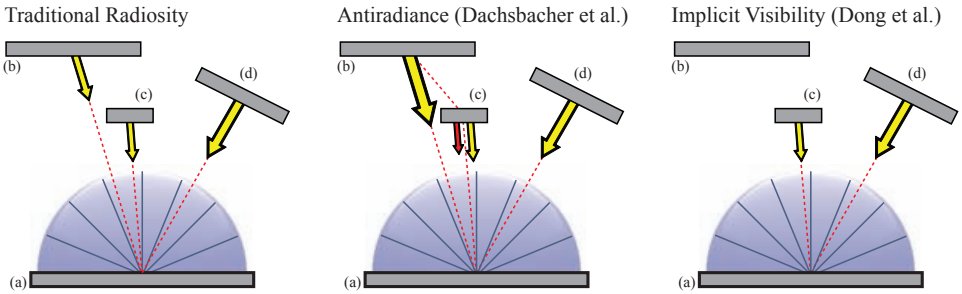


Figure 1: Light transport in different radiosity variants between a patch (a), and three other patches (transport links to (a) shown as dashed lines; sizes of yellow arrows indicate transported intensity): *Left:* radiosity computes visibility for each pair of patches; the transport from (b) is partly blocked. *Center:* the antiradiance method transports energy as if there is no blocking, but compensates for this by propagating negative light that originates from (b) via (c) to (a) (red arrow). *Right:* Dong et al. [4] discretize visibility to one link per direction bin eliminating transport from (b) to (a). In case of full visibility, as for patch (d), all three methods yield the same result.

agation. We consider our contribution being a principal study of deducing explicit from implicit visibility, and report statistical as well as visual results from our prototypal implementation. We also incorporate an effective way to let graphics artists define during the modeling stage where light transport cannot take place. For example, if there is no light exchange between two wings of a building, then we indicate this by simply placing a polygon somewhere in between. Our algorithm will use this coarse explicit visibility information to prevent the AR link mesh from growing large even prior to our reduction. Lastly, we describe a novel way for rendering high-quality antiradiance solutions without meshing or interpolating the final radiance across neighboring elements. At render time, we light the scene using every element as an area light source with an (anti)radiance distribution obtained from the global illumination solution. This allows the rendering of images with high quality interpolation and better contact shadows.

2 RELATED WORK

The importance of global illumination (GI) for computer graphics can be seen from the vast body of research papers in this field. Two main directions have been studied intensively in the last decades: ray tracing and radiosity methods; we refer the reader to excellent text books on these topics, e.g. Dutré et al. [5].

With the increasing computational power of graphics hardware, there have been many attempts to use GPUs to speed up global illumination. In recent years, research in ray tracing made a great leap forwards and there exist algorithms for real-time, parallel kd-tree construction [24], BVH construction [12], and fully interactive GI based on photon mapping [22].

Ray tracing based methods often cache information about the lighting in a scene, e.g. irradiance caching [23], photon mapping [9, 7], or instant radiosity [11]. In particular instant radiosity gained much attraction as represent the lighting of a scene by a set of virtual point lights, and thus easily maps to GPUs [18]. The light cuts method [21] clusters point lights into a hierarchy to speed up rendering. Final gathering is an essential step for computing high-quality images and a parallel algorithm therefor has recently be demonstrated [17].

In the following, we discuss work which is more closely related to our approach. There have been several attempts to compute radiosity solutions on the GPU. The main cost factor is the evaluation of the mutual visibilities between surfaces patches. Either rasterization together with the hemicube method [2, 1], or ray tracing on the GPU [19] have been used to compute form factors. The antiradiance reformulation [3] of the rendering equation [10] replaces explicit visibility by a recursive computation with negative light ("antiradiance"). Dong et al. [4] use a directional discretization and store only one link to the respective closest patch per direction. Thus the visibility is constructed implicitly with the link hierarchy. Although both methods are fundamentally different, both rely on a hierarchical link structure which initially had to be generated sequentially on the CPU. Meyer et al. [15] present a

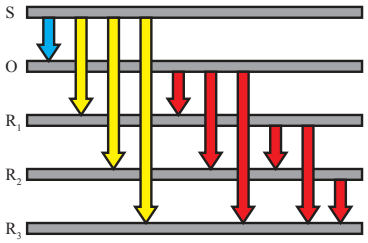


Figure 2: Radiance and antiradiance links for a 1D example. The normal of patch S is pointing downwards, the other surface normals are pointing towards S. There is only one unoccluded radiance link $S \xrightarrow{(+)} O$ (blue). The other radiance links $S \xrightarrow{(+)} R_n$ (yellow) and antiradiance links $R_n \xrightarrow{(-)} R_m$ (red) only exist for implicit occlusion handling.

data-parallel algorithm for link and patch creation, implemented in CUDA, which allows interactive radiosity methods with fully dynamic scenes.

Typically, the per-patch radiosity values that represent the GI solution are interpolated across adjacent patches. This can be done by generating an accordingly tessellated triangle mesh together with Gouraud shading [5] and improved using discontinuity meshing [14]. Dachsbacher et al. [3] used an image-space splatting approach that does not require a special mesh. Lehtinen et al. [13] directly compute global illumination using a meshless hierarchical representation and display the solution similarly.

3 ANTIRADIANCE

In this section we briefly review the antiradiance method [3], also following the notation of this work. The rendering equation describes the equilibrium of light transport in a scene and the radiance at a surface point x in direction ω_o is:

$$\begin{aligned} L(x,\omega_o) =& E(x,\omega_o) + \int_{\Omega^+} f(x,\omega_o,\omega_i) L_{in}(x,\omega_i) \cos\theta \, d\omega_i \\ =& E(x,\omega_o) + (\mathbf{K}L_{in})(x,\omega_o) \; . \end{aligned}$$

The incoming radiance at position x from direction ω_i originates from the closest surface in that direction. It is determined using the ray casting operator $ray(x,\omega_i)$ and part of the transport operator $\mathbf{G}$:

$$L_{in}(x,\omega_i) = L(ray(x,\omega_i),\omega_i) = (\mathbf{G}L)(x,\omega_i) \; . \quad (1)$$

As the computation of $\mathbf{G}$ is a very costly, the AR method strives to replace $\mathbf{G}$ by another transport operator that is cheaper to compute. Instead of resolving visibility explicitly by finding the nearest surface, the radiance is gathered from all surfaces along a ray yielding the transport operator $\mathbf{U}$. Extraneous light is then propagated and must be compensated. A pass-through operator $\mathbf{J}$ is defined that lets incoming radiance at a patch pass through without changing its magnitude or direction. The operators are related as follows:

$$\mathbf{G}L = \mathbf{U}L - \mathbf{U}\mathbf{J}\mathbf{G}L = \mathbf{U}(L-A) \quad (2)$$

with $A = \mathbf{J}\mathbf{G}L$ being the *antiradiance*. With this reformulation of the standard transport operator, the antiradiance rendering equation is obtained as:

$$L = E + \mathbf{K}\mathbf{U}(L-A) \quad (3)$$

$$A = \mathbf{J}\mathbf{U}(L-A) \; . \quad (4)$$

When L, A and E are projected into a suitable Hilbert base over the scene surface with finite dimensionality, these functions can be expressed as vectors of their components in that Hilbert space. Likewise, the operators $\mathbf{U}$, $\mathbf{K}$ and $\mathbf{J}$ become matrices:

$$\begin{pmatrix} L \\ A \end{pmatrix} = \begin{pmatrix} E \\ 0 \end{pmatrix} + \begin{pmatrix} \mathbf{K}\mathbf{U} & 0 \\ 0 & \mathbf{J}\mathbf{U} \end{pmatrix} \begin{pmatrix} L-A \\ L-A \end{pmatrix} \; . \quad (5)$$

We can then separate K out of this matrix and get:

$$\begin{pmatrix} L \\ A \end{pmatrix} = \begin{pmatrix} E \\ 0 \end{pmatrix} + \begin{pmatrix} \mathbf{K} & 0 \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{U}^L & 0 \\ 0 & \mathbf{J}\mathbf{U}^A \end{pmatrix} \begin{pmatrix} L-A \\ L-A \end{pmatrix} \; . \quad (6)$$

The thus remaining matrix describes the occluded light transport by means of the transport operator $\mathbf{U}$. Replacing it with $\left(\begin{smallmatrix} \mathbf{G} & 0 \\ 0 & 0 \end{smallmatrix}\right)$ yields the discretized equation for occluded light transport (as in standard radiosity) again. This comparison shows that the upper left part $\mathbf{U}^L$ of the matrix describes radiance transport while the lower right part $\mathbf{U}^A$ describes antiradiance transport.

4 REMOVING OCCLUDED LINKS

Eq. 6 shows that once we separated out the transport operator matrix we can interpret antiradiance and traditional radiosity as two extremes of how light is propagated between elements in the scene. However, the transport matrix does not have to take the form of either the fully occluded or the fully unoccluded transport. Within limits that we will discuss in the following, it is possible to create intermediate matrices $\mathbf{U}'^L$ and $\mathbf{U}'^A$ that contain transport with and without explicit visibility, i.e. essentially a mixture of entries from $\mathbf{G}$ and $\mathbf{U}$.

Let us first assume the case where we replace one unoccluded transport by a transport with explicit visibility. For this we define the matrix $\mathbf{U}'^L$ which contains one entry of $\mathbf{G}$, i.e. $\mathbf{U}'^L_{kl} = \mathbf{G}'^A_{kl}$, and all other entries are equal to $\mathbf{U}^L$. If the resulting light transport is correct, then the solutions L_{ij} and L'_{ij} for both matrices are equal. The equation for the k-th patch, L_k, becomes:

$$L_k = \sum_{i \neq l} \mathbf{K}_{ki}(\mathbf{U}^L_{ki}L_i - \mathbf{U}'^A_{ki}A_i) + \mathbf{K}_{kl}(\mathbf{G}_{kl}L_{kl} - \mathbf{U}'^A_{kl}A_{kl}) \; . \quad (7)$$

An entry $\mathbf{G}_{ij}$ is always less or equal to the respective entry $\mathbf{U}_{ij}$, and thus the sum over all $\mathbf{U}'^A_{ki}A_i$ in this equation must either be equal or less than the sum over all $\mathbf{U}^A_{ki}A_i$. This means, that *at least* one of the entries in this particular row of $\mathbf{U}'^A$ must be decreased in value. In other words, if the radiance transport between two patches is performed with proper occlusion, the receiving patch must no longer receive the same amount of antiradiance that was previously transported to it (to account for the occlusion along this transport path that we now consider explicitly). Note that although this shows that unoccluded and occluded light transport can be performed at the same time, no rules for the adjustments to $\mathbf{U}'^A$ can be derived from these equations alone. If we assume for now that we can replace values of $\mathbf{U}^L_{ij}$ one after another, then we can repeat this until $\mathbf{U}'^L$ equals $\mathbf{G}$, and in this case $\mathbf{U}'^A$ vanishes.

4.1 Link Removal in 1D

We have shown that mixing occluded and unoccluded light transport operations is possible under the restriction that antiradiance must only be transported to

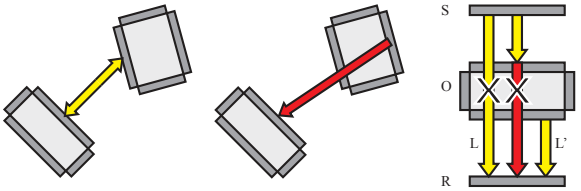


Figure 3: Convention for creating links: radiance links are created between patches facing each other (left); antiradiance links are in the negative hemisphere of a patch (center). Right: For every incoming link (here: L) we search if there are shorter incoming links (L'). Then we can remove L. Note that the antiradiance link shown in red is removed because of the same fact and thus the result is again correct at last.

patches that are the target of unoccluded transport operations. To introduce our algorithm we start with the instructional example of patches along one line. We will discuss rules for removing transport links for the case where a target patch is fully occluded from the source. Note that this information is solely based on the patch positions and extents, and the link mesh. Detecting partial occlusion, i.e. not only removing links for fully occluded patches, requires visibility computation, e.g. using ray casting that we still want to avoid. Also, we only consider opaque surfaces as potential occluders.

The motivation for the link removal is that the implicit handling of visibility in antiradiance generates a number of light transporting links that rises disproportionally with the depth complexity of the scene.

A simple set of surfaces in a 1D example as seen in Fig. 2 motivates the removal of unnecessary links. With unoccluded light transport, the topmost surface S illuminates all other surfaces. Therefore, it is linked to all of these surfaces, although the nearest surface O already fully occludes the light, and the light transported across the links to the surfaces further away needs to be compensated for. This is achieved by the antiradiance links from O to all other patches R_n below, and the pattern repeats analogously for every surface. When explicit visibility is taken into account, only the link $S \xrightarrow{(+)} O$ is required to illuminate this example scene correctly. With the implicit handling of occlusion as described above, $n+n!-1$ excess links are generated for n patches.

The first observation here is that all occluded radiance links in this example intersect the sole unoccluded patch O. Removing them along with all antiradiance links that originate from O itself effectively results in the occluded transport again. However, the remaining antiradiance links do not transport energy anymore since the patches where they start from do not receive any. To optimally reduce the link mesh, we also want to remove those from the transport matrix.

It is now possible to formulate two rules to find

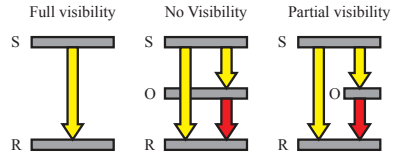


Figure 4: Classification of visibility in 3D used for our heuristics. A sender patch S emits radiance towards a receiving patch R, which may be blocked by a potential occluding patch O. Three situations need to be distinguished: full (left), none/fully occluded (middle) and none (right).

groups of links that can be removed from the link mesh without changing the result:

Rule 1 Search for a pattern of three links: from the sender S to the occluding receiver O, from S to the occluded receiver R, and from O to R. When such a pattern is found, both links to the occluded receiver $S \xrightarrow{(+)} R$ and $O \xrightarrow{(-)} R$ may be removed. We distinguish radiance and antiradiance links by the sign over the arrow.

Rule 2 Check for every *incoming link* of a patch L if there is another, shorter incoming link L'. If such a link is found, we can remove L as it is occluded (Fig. 3). Note that this rule is only valid if the objects in the scenes are manifolds with closed surfaces as also assumed in [3].

4.2 Heuristics for Link Removal in 3D

Obviously, the aforementioned removal in 1D retains validity in two or three dimensions only if it is applied to infinitely small patches along a single ray through the scene. For patches with finite size and only a finite number of discrete directions (*directional bins*), as used in the antiradiance method [3], we can derive heuristics from these rules that still work well when the scene discretization is reasonably fine. In Section 6 we evaluate the validity of both heuristics described in this section.

The modification compared to the 1D case is necessary due to the fact that the visibility function V between two patches is no longer either 0 or 1, but can take any value in-between (see Fig. 4). Furthermore, a special treatment is required to respect peculiarities in a hierarchical link mesh: to determine if the light transport between two patches is blocked, we might have to search across different levels of the hierarchy.

Heuristic 1 The first rule from Section 4.1 transforms into Algorithm 1. Again we find a combination of sender S, occluder O, and receiver R. However, we only remove the links, if O or one of its parents (in the patch hierarchy) subtends a large enough solid angle such that the light transport from a sender to another surface is completely blocked (see Fig. 5). In addition, the radiance link $S \xrightarrow{(+)} R$ must have the same (discrete) direction as the antiradiance link $O \xrightarrow{(-)} R$. To test this,

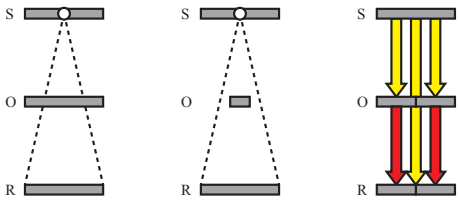


Figure 5: Left: the occluder O, as seen from S (note that in the antiradiance method links are established between patch centers), subtends a larger solid angle than the receiver patch R. Center: the subtended solid angle of O is too small and the link from S to R cannot be removed. Right: the radiance link $S \xrightarrow{(+)} R$ is on a different level of the hierarchical link mesh than the antiradiance links $O \xrightarrow{(-)} R$. Thus we also consider parents in the patch hierarchy when searching for blockers.

we construct two infinite circular cones with their tips in S: the first one, C_{Link} connects the centers of S and O with its axis and has an opening angle so that the solid angle subtended by this cone equals that of O as seen from S. The second cone, C_{bin} has the discrete direction of the link's source bin as its axis and its opening angle is chosen so that C_{bin} covers a solid angle of ω_{bin}. Then, C_{Link} must enclose C_{bin} (cf. Alg. 1 lines 6-10). This ensures that all patches lie on one line, analogous to the 1D example before. When using hierarchical link meshes, an occluder can be linked to the receiver on a finer level than the sender-receiver link, in which case the patch at the finer level takes on the role of R (Fig. 5, right, Alg. 1 line 5). Also, the full extent of the occluding geometry can be better estimated by checking the solid angles subtended by any parents of the suspected occluding patch.

Heuristic 2 The second removal rule transfers to the 3D case analogous to the previous one (see also Algorithm 2). The aforementioned restriction to scenes consisting entirely of closed manifolds stays valid. When

Algorithm 1 Find and disable all links between senders, occluders and receivers.

```
1  for each radiance link L
2    required[L] = true
3
4  for each radiance link L
5    for all target patch P_j of link L and its parents
6      C_Link ← cone(center(P_i), actualdir(L), ω_{i←j})
7      // sourcedir(L) is discreet source direction of Link L
8      C_bin ← cone(center(P_i), sourcedir(L), ω_bin)
9      if ω_{i←j} > ω_bin and C_Link encloses C_bin
10       binOccluded ← true
11
12   if binOccluded
13     for all antriadiance links L' starting from P_j
14       P_k ← target_patch(L)
15       if link P_i → P_k exists
16         if sourcedir(L)=sourcedir(L')
17           if targetdir(L)=targetdir(L')
18             required[L] ← false
19             required[L'] ← false
20
21 remove all links where required[L] is false
```

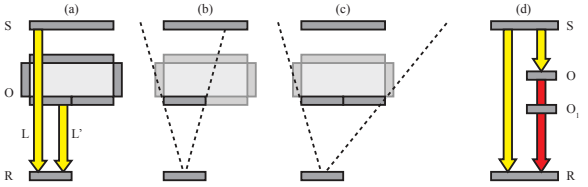


Figure 6: (a) Two incoming links L and L' at R have the same direction. To determine if we can remove L, we need to test if the transport from S to R is blocked. Testing this blocking with the patch O where L' emanates from does not reveal this information. Ascending the hierarchy and testing with the parent of O determines full blocking (c) and L can be faithfully removed. (d) A situation where a fully occluded antiradiance link must not be removed: The link $S \xrightarrow{(+)} R$ is partially occluded by O and O_1 and the link $O \xrightarrow{(-)} R$ is fully occluded by O_1, but the antiradiance produced at O must arrive at R to correctly reproduce the partial occlusion. Therefore, the link $O \xrightarrow{(-)} R$ must not be removed.

we test if a link $S \xrightarrow{(+)} R$ can be removed, we need to find a shorter link from a potential occluder O to R. However, these two links must have the same (discretized) direction (Alg. 2 line 4). If we detect such a situation, then we need to determine if O, or the surface to which is belongs, is large enough to block the transport from S to R by ascending the hierarchy from O to its largest parent. Next, we project S onto the plane of this parent patch and only if the projection is fully on that surface the link can be removed (Fig. 6, Alg. 2 lines 9-15). Note that patches that share a common parent with S will never be considered as occluders (Alg. 2 line 8).

However, not all links that are detected as fully occluded may actually be removed. As illustrated in Fig. 6 antiradiance links might be necessary to capture partial occlusion although the above heuristic marks them as occluded. Algorithm 3 detects these links and prevents them from being removed. For each partially visible link $S \rightarrow R$ it searches all other antiradiance links

Algorithm 2 Classify Link visibility as seen from target patch.

```
1  for each radiance link L
2    visibility[L] = full
3    for each link L' ending at target patch of L
4      if target_bin(L)=target_bin(L')
5        R ← target_patch(L)
6        S ← source_patch(L)
7        O ← source_patch(L')
8        if S and O do not have same parent patch
9          // If S or O are clusters, the following is
10         // checked on every pair of patches in these clusters
11         O' ← biggest parent of O
12         S' ← perspective projection of S onto O'
13         if S on far side of O' and S' fully inside O'
14           visibility[L] ← none
15           continue
16         else
17           visibility[L] ← partial
```

Algorithm 3 Finding required occluded links.

```
1  for each link L
2    if visibility[L] = none
3      required[L] = false
4    else
5      required[L] = true
6
7  for each link L with visibility[L]=partial
8    S ← source_patch(L)
9    R ← target_patch(L)
10   for R and each child patch of R
11     for each antiradiance link L'
12       S' ← source_patch(L')
13       R' ← target_patch(L')
14       if R=R'
15         for each link L'' connecting S to S'
16           if visibility[L''] != none and L'' is radiance link
17             required[L'] ← true
```

$S' \xrightarrow{(-)} R$ to the same target patch and checks if a non-occluded link $S \rightarrow S'$ exists. If so, $S' \xrightarrow{(-)} R$ is required to keep the partial occlusion of S' from R by S intact and is marked as required. Note that this heuristic cannot be used after heuristic 1 because links $S' \xrightarrow{(-)} R$ may already have been removed, resulting in required links getting missed and removed.

Patches without incoming links In case of static direct lighting in the scene we can also remove all links emanating from patches that are neither light sources nor have any incoming links, as these links will never transport energy. However, similar to the original antiradiance implementation [3] we typically use shadow maps for computing direct illumination and thus potentially every patch can receive energy that has to be propagated further.

4.3 User-Defined Link Removal

In addition to the link removal heuristics described above, we can optionally perform link removal based on user-defined (invisible) blocking geometry which strictly cuts all links that intersect it. This additional geometry can be a simple polygon generated by the user along with the scene to separate parts of the scene that obviously do not directly exchange light with each other, e.g. two rooms separated by solid walls (see Fig. 8). Since this geometry consists of few polygons only, we test for every link if it intersects the blocking geometry without generating high cost, and remove the link if this is the case. Note that this is similar to Fradin et. al. [6] who used manually placed portals to section large scenes.

5 FINAL SHOOTING

Dachsbacher et al. [3] used a splatting approach, similar to point-based rendering, to render an image with interpolated patch colors. Instead we propose to use a "final shooting" approach: we treat each patch in the scene as a *patch light source* (PLS) with a directional intensity distribution according to the total exitant intensity determined by the antiradiance solver. For rendering the final solution, we simply light the scene only using the PLSs. This approach can be seen as a variant of instant radiosity [11], where light sources emit radiance and antiradiance (computed using the antiradiance method) and thus account for shadowing implicitly. Note that the resulting number of virtual light sources in our approach is typically orders of magnitudes higher. Furthermore, every PLS is an area light source from which lighting computation is more intricate than from point lights. To this end, when computing the lighting of a fragment due to a PLS, we replace the PLS by 8 point light sources randomly placed on the PLS. This can be seen as a Monte-Carlo sampling of the area light sources; no noise is visible in the images, as the number of PLS is very high (it equals the number of patches in the scene). Note that by lighting the scene with all PLSs we obtain not only a smooth interpolation, but also one (additional) indirect bounce at the same time with no additional cost.

When using a full link mesh, we efficiently accumulate the contributions of all PLSs using deferred shading and interleaved sampling [20]. However, care has to be taken when using final shooting together with the link removal heuristics: in this case, only those patches are to be lit by a PLS that are still linked to it after reducing the link mesh. To account for this, instead of using deferred shading, we have to render every receiver patch for every PLS, compute per-pixel lighting, and accumulate the contributions. Note that this process benefits from the GPU's early-z culling automatically omitting occluded receivers. For lighting computing from a PLS, we look up the interpolated intensity towards a fragment using precomputed interpolation weights in a cube map (6×512^2 resolution in our examples).

6 RESULTS AND DISCUSSION

In this section we compare the heuristics and the user-defined link removal. To determine the impact of the heuristics' approximation regarding the link removal, we modified both to perform explicit visibility checks using ray casting and Monte Carlo sampling instead of the link mesh based checks. Note that no heuristic *should* remove more links than those removed with explicit visibility testing. We have implemented an Antiradiance solver similar to the one described in [3], using the same algorithm for building the hierarchy. The preprocessing, including the previously discussed link removal heuristics, is initially implemented and executed on the CPU (running on multiple cores), while OpenGL is used for the simulation of the light transport and for displaying the result. While our current implementation can be seen as an experimental prototype, we plan to integrate our heuristics into the data-parallel link generation method by Meyer et al. [15] in the future.

			heuristic 1				heuristic 2			
scene	patches	links	explicit test	heuristic removal	incorrect	not removed	explicit test	heuristic removal	incorrect	not removed
Japan	12745	629665	157449	129313	54872 (42%)	83058 (52%)	71007	34408	11732 (34%)	48331 (68%)
Office	14246	1470440	581768	395592	124232 (31%)	310408 (53%)	260475	85075	28739 (34%)	204139 (78%)
Desks	14396	1465632	759669	280705	7752 (10%)	486716 (40%)	690145	300316	11577 (4%)	401406 (58%)
Soda Hall	25023	2774452	2076609	1621749	73998 (5%)	528858 (25%)	1888014	1016923	21026 (2%)	892117 (47%)

Table 1: Results of applying our heuristics to the test scenes with: the total number of radiance and antiradiance links, the number of links removed by the explicit test, the links removed by the heuristic, the number of incorrectly removed links compared to the explicit test, and the number of links that have not been removed by the heuristic but by the explicit test.

scene	patches	links	heuristic 1	heuristic 2
Japan	12745	629665	22.1s	26.0s
Office	14246	1470440	248.2s	224.9s
Desks	14396	1465632	80.5s	127.0s
Soda Hall	25023	2774452	87.9s	280.0s

Table 2: Measured run time for our multithreaded CPU implementation of the heuristics running 8 concurrent threads (averaged over 10 runs)

6.1 Link Removal Heuristics

We tested our link removal heuristics against the explicit visibility oracle on various scenes: the Japanese room and office from [3], the "desks" shown in Fig. 8 and the model of the fifth floor of the Soda Hall (see Fig. 12). The results are summarized in Table 1 and the run times are given in Table 2. All measurements were taken on an Intel Core i7 CPU with 2.66MHz, 6GB RAM, and an NVIDIA GeForce GTX 465 with 1024MB RAM, running Linux.

Our results show that the heuristics also remove links that are not totally occluded. In the Japanese room and office scene, which both consist of a single room with many objects inside, the error rate is particularly high while it is low for the "desks" scene and the Soda Hall with their walls as main occluders. This shows that while both heuristics perform well with big solid occluders, they are prone to inaccuracy with correctly estimating visiblity along silhouettes of objects. The first heuristic relies on the subtended solid angle for testing blocking. However, this gives no indication about the actual patch shapes: for instance, elongated patches can be visible although nearly quadratic patches with larger solid angles are in front of them. The second heuristic erroneously removes links for which unblocked paths between the patches may still exist although the projection test succeeds. The runtime for heuristic 1 depends on the number of occluders in the scene: the algorithm only needs one occluder per link and thus has to search less links and terminates quicker when many large occluders are present. Heuristic 2 spends most time in algorithm 3, whose complexity is dominated by the number of links that have to be searched.

Although the resulting error due to the link removal heuristics seems significant for the affected scenes, the impact on the rendered images is hardly perceivable (see Fig. 7 and Fig. 10a). Obviously, errors are only introduced for links whose contribution (either radiance or antiradiance) is negligible. The first heuristic creates generally brighter shadow regions. It discards $O \xrightarrow{(-)} R$

(a) heuristic 1 (b) heuristic 2

(c) 4× difference heuristic 1 (d) 4× difference heuristic 2

Figure 7: The Japan scene rendered with reduced link meshes using final shooting (top), and then luminance differences to an image computing with the full link mesh (red is less, green is greater than original). The difference images have been scaled by a factor of 4. Compare Fig. 9, 10a for references.

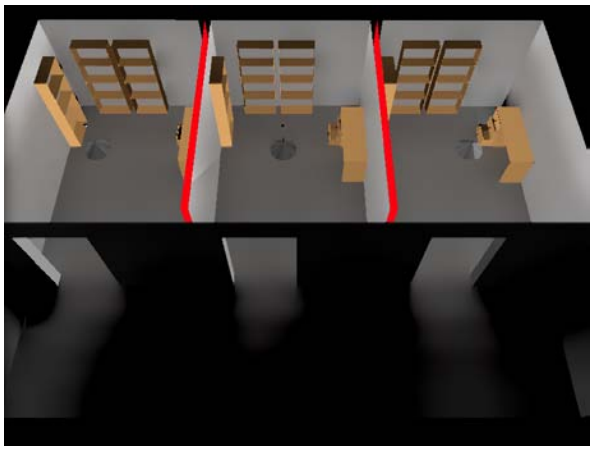


Figure 8: Desk scene with blocking geometry (red) after 6 antiradiance iterations, rendered with splatting.

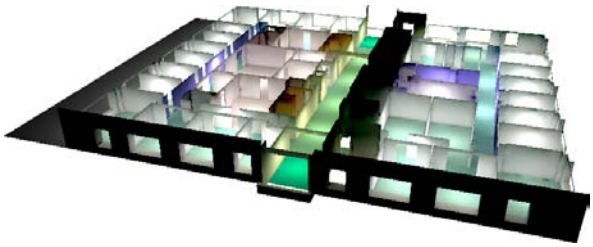


Figure 12: Soda Hall level 5 lit by diffuse emitting area light sources in the rooms and on the hallways. It was rendered using final shooting and the reduced link mesh produced using heursitic 1 (see Table 1).

links even if there exists another link $S' \xrightarrow{(+)} R$ that is only partially occluded by O and requires the antiradiance generated at O to correctly light R. The second algorithm of heuristic 2, which is run to preserve partial occlusion, results in a stronger tendency to keep antiradiance links. Thus, the corresponding difference image shows lesser brightening of shadow regions. Applying heuristic 2 followed by heuristic 1 on the Japanese room and Soda Hall scenes results in 144983 removed links (45.5% error) and 1628987 removed links (4.7% error), respectively, and visual quality comparable to Fig. 7.

We tested the blocking planes by adding two such polygons between the rooms in the "desks" scene (see Fig. 8). These two quads alone caused a removal of almost 76% of all occluded links in the scene, demonstrating that this mechanism is not only cheap to compute, but also highly efficient.

6.2 Rendering Quality

Fig. 9 shows resulting images for the Japanese room without interpolation, with splatting, and final shooting. The global illumination solution has been computed with 4 antiradiance iterations and 128 direction bins for all of our test scenes.

At a resolution of 800×600 pixels the rendering speed was 8.5 frames per second with no interpolation, 7.7 fps with splatting, and 0.04 fps with shooting. Although final shooting has a considerable impact on performance due to the high number PLSs, the resulting images capture finer details, e.g. contact shadows, due to patches emitting antiradiance. Interleaved sampling for final shooting with 4×4 interleaved sampling runs at 0.44 fps, and the performance increases further when using larger interleaving patterns yielding 1.22 fps at 8×8, and 2.88 fps at 16×16. However, at some point the wider Gaussian blur filter again removes details (see Fig. 10). Nichols et al. [16] report tremendous speedups with multi-resolution splatting compared to interleaved sampling, however, there is another way to speed up shooting. So far, we used the leaf nodes in the hierarchy as PLSs, but we also can use interior nodes therefor. Using the leaves' parents yields about 75% less PLSs, but only slightly reduces the amount of detail in the lighting as we treat them as area lights (see Fig. 11), but already yields a significant speedup: shooting at full resolution runs at 0.14 fps, at 1.29 fps with 4×4, at 2.99 fps with 8×8, and at 5.09 fps with 16×16 interleaving, respectively. A further optimization, and direction for future work, is to exploit the patch hierarchy also for selecting the PLSs, in spirit of [21].

7 CONCLUSIONS

In this paper we studied heuristics operating directly on the hierarchical link mesh of the antiradiance method to deduce explicit visibility information and thus to reduce the number of links. The energy propagation is the most time consuming step in antiradiance methods and greatly benefits from link removal as its cost is proportional to the number of links. Our heuristics remove links more faithfully than Dong et al.'s method [4] yielding results of similar quality as the antiradiance method. The user-defined blocker geometry can further be used to remove links with little computation. The final shooting simplifies the rendering of a high-quality final solution and yields better results and requires less tweaking than Dachsbacher et al.'s splatting approach.

Obviously, our heuristics will have to be incorporated into a data-parallel link generation method, such as Meyer et al.'s work [15] to actually speed up antiradiance in fully dynamic scenes. This data-parallel algorithm is about two orders of magnitude faster than our (and Dachsbacher et al.'s) CPU-based link generation algorithm (45 million links/s versus 140 thousand links/s), and we would expect a similar speedup for the link removal. Another challenge is to create blocking geometry automatically by analyzing the scene geometry. We believe that these two steps will enable our link removal in fully dynamic scenes at interactive speed.

ACKNOWLEDGEMENTS

The authors would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart.

REFERENCES

[1] Attila Barsi, László Szirmay-Kalos, and Gábor Szijártó. Stochastic glossy global illumination on the gpu. In *Spring Conference on Computer Graphics 2005*, pages 187–193, 2005.

[2] Greg Coombe, Mark J. Harris, and Anselmo Lastra. Radiosity on graphics hardware. In *Graphics Interface 2004*, pages 161–168, 2004.

[3] Carsten Dachsbacher, Marc Stamminger, George Drettakis, and Frédo Durand. Implicit visibility and antiradiance for interactive global illumination. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 26(3):61, 2007.

Figure 9: The Japanese room with different rendering methods. The room is lit from outside the window on the left. The images were rendered with 4 iterations at a resolution of 800×600 pixels with a full link mesh.

(a) PBRT reference (b) 4×4 interleaved sampling (c) 16×16 interleaved sampling

Figure 10: Results with different interleaved sampling patterns: larger patterns result in stronger blurring and loss of small scale features.

(a) final shooting (b) 4×4 interleaved sampling (c) 16×16 interleaved sampling

Figure 11: Final shooting using not the leafs of the patch hierarchy, but its parents as PLSs. This results in significantly faster rendering speed, sacrificing only little details in the lighting (e.g. visible in shadowed areas around the bookshelf).

[4] Zhao Dong, Jan Kautz, Christian Theobalt, and Hans-Peter Seidel. Interactive global illumination using implicit visibility. In *Proc. of Pacific Graphics*, pages 77–86, 2007.

[5] P. Dutré, K. Bala, and P. Bekaert. *Advanced Global Illumination*. AK Peters, 2006.

[6] D. Fradin, D. Meneveaux, and S. Horna. Out-of-core photon-mapping for large buldings. In *Proceedings of Eurographics symposium on Rendering*, June 2005.

[7] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, pages 1–8, 2008.

[8] Nicolas Holzschuch, François X. Sillion, and George Drettakis. An efficient progressive refinement strategy for hierarchical radiosity. In *Photorealistic Rendering Techniques (Eurographics Workshop on Rendering)*, pages 357–372, 1994.

[9] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., 2001.

[10] James T. Kajiya. The rendering equation. *Computer Graphics (Proc. of SIGGRAPH '86)*, 20(4):143–150, 1986.

[11] Alexander Keller. Instant radiosity. In *SIGGRAPH '97*, pages 49–56, 1997.

[12] Christian Lauterbach, Michael Garland, Shubhabrata Sengupta, David Luebke, and Dinesh Manocha. Fast bvh construction on gpus. *Computer Graphics Forum*, 28(2), 2009.

[13] Jaakko Lehtinen, Matthias Zwicker, Emmanuel Turquin, Janne Kontkanen, Frédo Durand, François X. Sillion, and Timo Aila. A meshless hierarchical representation for light transport. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 27(3):1–9, 2008.

[14] Dani Lischinski, Filippo Tampieri, and Donald P. Greenberg. Discontinuity meshing for accurate radiosity. *IEEE Computer Graphics and Applications*, 12(6):25–39, 1992.

[15] Quirin Meyer, Christian Eisenacher, Marc Stamminger, and Carsten Dachsbacher. Data-Parallel, Hierarchical Link Creation. In *Proceedings of the Eurographics Symposium on Parallel Graphics and Visualization*, 2009.

[16] Greg Nichols, Jeremy Shopf, and Chris Wyman. Hierarchical image-space radiosity for interactive global illumination. In *Computer Graphics Forum 28(4)*, pages 1141–1149, 2009.

[17] Tobias Ritschel, Thomas Engelhardt, Thorsten Grosch, Hans-Peter Seidel, Jan Kautz, and Carsten Dachsbacher. Micro-rendering for scalable, parallel final gathering. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)*, 28(5), 2009.

[18] Tobias Ritschel, Thorsten Grosch, Min H. Kim, Hans-Peter Seidel, Carsten Dachsbacher, and Jan Kautz. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)*, 27(5), 2008.

[19] Arne Schmitz, Markus Tavenrath, and Leif Kobbelt. Interactive global illumination for deformable geometry in cuda. *Computer Graphics Forum (Proc. of Pacific Graphics 2008)*, 27(7), 2008.

[20] B. Segovia, J. C. Iehl, R. Mitanchey, and B. Péroche. Non-interleaved deferred shading of interleaved sample patterns. In *Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics Hardware*, pages 53–60, 2006.

[21] Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. Lightcuts: a scalable approach to illumination. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 24(3):1098–1107, 2005.

[22] Rui Wang, Rui Wang, Kun Zhou, Minghao Pan, and Hujun Bao. An efficient gpu-based approach for interactive global illumination. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 28(3):1–8, 2009.

[23] Gregory J. Ward and Paul S. Heckbert. Irradiance gradients. In *Eurographics Workshop on Rendering*, pages 85–98, 1992.

[24] Kun Zhou, Qiming Hou, Rui Wang, and Baining Guo. Real-time kd-tree construction on graphics hardware. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)*, 27(5):1–11, 2008.

Analysis and design of the dynamical stability of collective behavior in crowds

Albert Mukovskiy
Section for Computational Sensomotorics, Department of Cognitive Neurology, Hertie Institute for Clinical Brain Research & Centre for Integrative Neuroscience, University Clinic, Tübingen, Germany
albert.mukovskiy@medizin.uni-tuebingen.de

Jean-Jacques E. Slotine
Nonlinear Systems Laboratory, Department of Mechanical Engineering, MIT; Cambridge, MA, USA
jjs@mit.edu

Martin A. Giese
Section for Computational Sensomotorics, Department of Cognitive Neurology, Hertie Institute for Clinical Brain Research & Centre for Integrative Neuroscience, University Clinic, Tübingen, Germany
martin.giese@uni-tuebingen.de

ABSTRACT

The modeling of the dynamics of the collective behavior of multiple characters is a key problem in crowd animation. Collective behavior can be described by the solutions of large-scale nonlinear dynamical systems that describe the dynamical interaction of locomoting characters with highly nonlinear articulation dynamics. The design of the stability properties of such complex multi-component systems has been rarely studied in computer animation. We present an approach for the solution of this problem that is based on Contraction Theory, a novel framework for the analysis of the stability complex nonlinear dynamical systems. Using a learning-based realtime-capable architecture for the animation of crowds, we demonstrate the application of this novel approach for the stability design for the groups of characters that interact in various ways. The underlying dynamics specifies control rules for propagation speed and direction, and for the synchronization of the gait phases. Contraction theory is not only suitable for the derivation of conditions that guarantee global asymptotic stability, but also of minimal convergence rates. Such bounds permit to guarantee the temporal constraints for the order formation in self-organizing interactive crowds.

Keywords: computer animation, crowd animation, coordination, distributed control, stability.

1 INTRODUCTION

Dynamical systems are frequently applied in crowd animation for the simulation of autonomous and collective behavior of many characters [MT01], [TCP06]. Some of this work has been inspired by observations in biology, showing that coordinated behavior of large groups of agents, such as flocks of birds, can be modelled as emergent behavior that arises from the dynamical coupling between interacting agents, without requiring an external central mechanism that ensures coordination [CS07, Cou09], [CDF+01]. Such models can be analyzed by application of methods from nonlinear dynamics [PRK03]. The simulation of collective behavior by self-organization in systems of dynamically coupled agents is interesting because it might reduce the computational costs of traditional computer animation techniques, such as scripting or path planning [TCP06, Rey87]. In addition, the generation of collective behavior by self-organization allows to implement spontaneous adaptation to external perturbations or changes in the system architecture, such as the variation of the number of characters. However, due to the complexity of the models describing individual characters the mathematical analysis of the underlying dynamical systems is typically quite complicated.

In crowd animation, some recent studies have tried to learn interaction rules from the behavior of real human crowds [DH03], [PPS07], [LFCCO09]. Other work has tried to optimize interaction behavior in crowds by exhaustive search of the parameter space exploiting computer simulations by definition of appropriate cost functions (e.g. [HMFB01]). However, most of the existing approaches for the control of group motion in computer graphics have not taken into account the effects of the articulation during locomotion on the control dynamics [PAB07], [NGCL09], [KLLT08]. Consequently, the convergence and stability properties of such dynamical animation systems have rarely been addressed. Distributed control theory has started to study the temporal and spatial self-organization of crowds of agents, and the design of appropriate dynamic interactions, typically assuming rather simple and often even linear agent models (e.g. [SS06], [PLS+07], [SDE95]). However, human-like characters are characterized by highly complex kinematic and even dynamic properties, c.f. [BH97]. Consequently, approaches for a systematic analysis and design of the dynamical properties

of crowd animation systems are largely lacking. However, such methods seem highly desirable, since they permit one to guarantee desired system properties and to ensure the robustness of the generated behavior under variations of system inputs and the system parameters.

In this paper we introduce Contraction Theory ([LS98], [PS07]) as a framework that makes such stability problems tractable, even for characters with multiple coupled levels of control. Contraction Theory provides a useful tool specifically for modularity-based stability analysis and design [Slo03], [WS05]. This framework is applied to a simple learning-based animation architecture for the real-time synthesis of the movements of interacting characters, which is based on a method that approximates complex human behavior by relatively simple nonlinear dynamical systems [GMP$^+$09], [PMSA09]. Consistent with related approaches in robotics [RI06], [BC89], [GRIL08], [Ijs08], [BRI06], [GTH98], [CS93], this method generates complex movements by the combination of the learned movement primitives [OG06], [GMP$^+$09]. The resulting system architecture is rather simple, making it suitable for a mathematical treatment of dynamical stability properties.

The paper is structured as follows: The structure of the animation system is sketched in section 2. The dynamics underlying navigation control is described in section 3. Subsequently, in section 4 we introduce some basic ideas from Contraction Theory. The major results of our stability analysis and some demos of their applications to the control of crowds are described in section 5, followed by the conclusions.

2 SYSTEM ARCHITECTURE

Our investigation of the collective dynamics of crowds was based on a learning-based animation system, described in details in [GMP$^+$09] (see Fig. 1). By applying anechoic demixing [OG06] to motion capture data, we learned spatio-temporal components. These source components were generated online by nonlinear dynamical systems, Andronov-Hopf oscillators. The mappings σ_j between the stable solutions of the nonlinear oscillators and the required source functions were learned by application of kernel methods [GMP$^+$09]. Each character is modelled by a single limit cycle oscillator, whose solution is mapped by support vector regression (SVR) onto three source signals. These signals were then superimposed with different linear weights w_{ij} and phase delays τ_{ij} in order to generate the joint angle trajectories $\xi_i(t)$ (see Fig. 1). By blending of the mixing weights and the phase delays, intermediate gait styles were generated. This allowed us to simulate specifically walking along paths with different curvatures, changes in step length and walking style. Interactive behavior of multiple agents can be modelled by making the states of the oscillators and the mixing

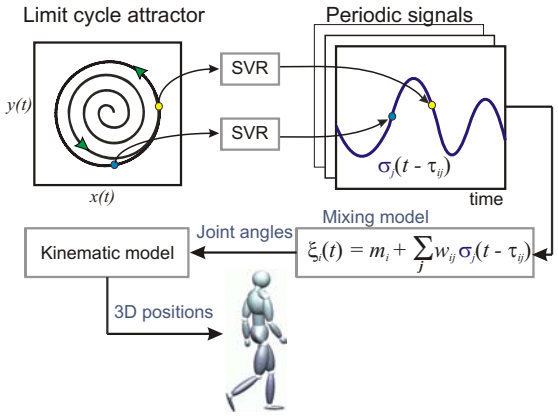


Figure 1: Architecture of the simulation system.

weights dependent on the behavior of the other agents. Such couplings result in a highly nonlinear system dynamics.

The heading direction of the characters was changed by morphing between curved gaits, controlled by a nonlinear navigation dynamics. In the shown applications this dynamics steers the avatars towards goal points that were placed along parallel straight lines. The heading dynamics was given by a nonlinear first-order differential equation (see [GMP$^+$09] for details). Control of heading direction was only active during the the initial stage of the organization of the crowd, resulting in an alignment of the avatars along the parallel straight lines, independent of their initial positions and gait phases. (See Fig. 2 and Fig. 3).

3 CONTROL DYNAMICS

Beyond the control of heading direction, the analyzed scenarios of order formation in a group of characters require the control of the following variables: 1) phase within the step cycle, 2) step length, 3) gait frequency, and 4) heading direction.

The dynamics of each individual character was modelled by an Andronov-Hopf oscillator with constant equilibrium amplitude ($r_i^* = 1$). For appropriate choice of parameters, these nonlinear oscillators have a stable limit cycle that corresponds to a circular trajectory in phase space [AVK87].

In polar coordinates and with the instantaneous eigenfrequency ω this dynamics is given by: $\dot{r}(t) = r(t)\left(1 - r^2(t)\right)$, $\dot{\phi}(t) = \omega$. Control affects the instantaneous eigenfrequency ω of the Andronov-Hopf oscillators and their phases ϕ, while the first equation guarantees that the state stays on the limit cycle ($r(t) = 1, \forall t$).

The position z_i of each character along the parallel paths (see Fig. 2) fulfills the differential equation $\dot{z}_i(t) = \dot{\phi}_i g(\phi_i)$, where the positive function g determines the propagation speed of the character depending on the phase within the gait cycle. This nonlinear

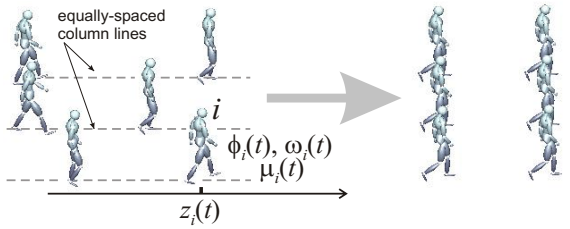


Figure 2: Crowd coordination scenario. Every character i is characterized by its position $z_i(t)$, the phase $\phi_i(t)$ and the instantaneous eigenfrequency $\omega_i(t) = \dot{\phi}_i(t)$ of the corresponding Andronov-Hopf oscillator, and a step-size scaling parameter $\mu_i(t)$.

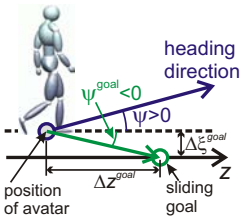


Figure 3: A sliding goal for each avatar was placed on a straight line at fixed distance ahead in z-direction. Heading direction angle: $\psi^{heading}$ and goal direction angle: ψ^{goal}.

function was determined empirically from a kinematic model of character. By integration of this propagation dynamics one obtains $z_i(t) = G(\phi_i(t) + \phi_i^0) + c_i$, with an initial phase shift ϕ_i^0 and some constant c_i depending on the initial position and phase of avatar i, and with the monotonously increasing function $G(\phi_i) = \int_0^{\phi_i} g(\phi) d\phi$, assuming $G(0) = 0$. Three control rules described:

I) Control of step frequency: A simple form of speed control is based on making the frequency of the oscillators $\dot{\phi}_i$ dependent on the behavior of the other characters. Let ω_0 be the equilibrium frequency of the oscillators without interaction. Then a simple controller is defined by the differential equation

$$\dot{\phi}_i(t) = \omega_0 - m_d \sum_{j=1}^{N} K_{ij}[z_i(t) - z_j(t) - d_{ij}] \qquad (1)$$

The constants d_{ij} specify the stable pairwise relative distances in the formed order for each pair (i, j) of characters. The elements of the link adjacency matrix $\mathbf{K}$ are $K_{ij} = 1$ if characters i and j are coupled and zero otherwise. In addition, we assume $K_{ii} = 0$. The constant $m_d > 0$ defines the coupling strength.

With the Laplacian $\mathbf{L}^d$ of the coupling graph, which is defined by $L_{ij}^d = -K_{ij}$ for $i \neq j$ and $L_{ii}^d = \sum_{j=1}^{N} K_{ij}$, and the constants $c_i = -\sum_{j=1}^{N} K_{ij} d_{ij}$ the last equation system can be re-written in vector form:

$$\dot{\phi} = \omega_0 \mathbf{1} - m_d(\mathbf{L}^d G(\phi + \phi^0) + \mathbf{c}) \qquad (2)$$

II) Control of step length: Step length was varied by morphing between gaits with short and long steps. A

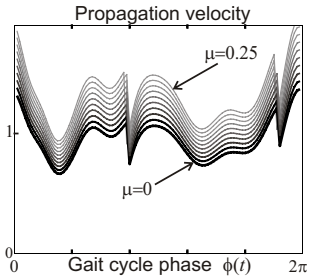


Figure 4: Propagation velocity for 10 different values the of step length morphing parameter $\mu = [0 \ldots 0.25]$ dependent on gait cycle phase $\phi(t)$ and $\omega(t) = 1$. The vertical axis is scaled in order to make all average velocities equal to one for $\mu = 0$ (lowest thick line). This empirical estimates are well approximated by $(1+\mu)g(\phi(t))$.

detailed analysis shows that the influence of step length on the propagation could be well captured by simple linear rescaling. If the propagation velocity of characters i is $v_i(t) = \dot{z}_i(t) = \dot{\phi}_i(t) g(\phi_i(t)) = \omega_i(t) g(\phi_i(t))$ for the normal step size, then the velocity for modified step size was well approximated by $v_i(t) = \dot{z}_i(t) = (1+\mu_i)\omega_i(t) g(\phi_i(t))$ with the morphing parameter μ_i. The range of morphing parameters was restricted to the interval $-0.5 < \mu_i < 0.5$, where this linear scaling law was fulfilled with high accuracy. The empirically estimated propagation velocity in heading direction, dependent on gait phase, is shown in Fig.4 for different values of the step length morphing parameter μ_i. Using the same notations as in equation (1), this motivates the definition of the following dynamics that models the influence of the step length control on the propagation speed:

$$\dot{\mathbf{z}} = \omega g(\phi + \phi^0)(1 - m_z(\mathbf{L}^z \mathbf{z} + \mathbf{c})) \qquad (3)$$

In this equation $\mathbf{L}^z$ signifies the Laplacian of the relevant coupling graph, and m_z the strength of the coupling. For uncoupled characters ($m_z = 0$) this equation is consistent with the the definition of propagation speed that was given before.

III) Control of step phase: By defining separate controls for step length and step frequency it becomes possible to dissociate the control of position and step phase of the characters. Specifically, it is interesting to introduce a controller that results in phase synchronization between different characters. This can be achieved by addition of a simple linear coupling term to equation (1), written in vector form:

$$\dot{\phi} = \omega_0 \mathbf{1} - m_d(\mathbf{L}^d G(\phi + \phi^0) + \mathbf{c}) - k\mathbf{L}^\phi \phi \qquad (4)$$

with $k > 0$ and the Laplacian $\mathbf{L}^\phi$. (All sums or differences of angular variables were computed by modulo 2π).

IV) Control of heading direction:

The heading directions of the characters were controlled by a navigation dynamics that steers the avatars

towards goal points, which were placed along parallel straight lines in front of the avatars (2). The heading dynamics was given by a nonlinear differential equation, independently for every character [GMP+09]:

$$\dot{\psi}_i = \sin(\psi_i^{goal} - \psi_i) \tag{5}$$

where $\psi_i^{goal} = \arctan(\Delta\xi_i^{goal}/\Delta z_i^{goal})$, $\Delta\xi_i^{goal}$ is the distance to the goal line in the direction orthogonal to the propagation direction, while Δz_i^{goal} is constant, (see Fig. 3). The morphing weight that controls the mixture of walking with left and right turning was proportional to $\dot{\psi}_i(t)$. For the mathematical stability analysis presented in the following, we neglected the influence of the dynamics of the control of heading direction, focusing on the order formation scenarios when the agents' heading directions are already aligned, when they walk along parallel straight lines towards sliding goal points. In this case, the positions of the agents can be described by a single position variable $z(t)$. An extension of the developed analysis framework including the control of the heading direction is in progress.

The mathematical results derived in the following sections apply to subsystems of the complete system dynamics that is given by equations (3) and (4). In addition, simulations are presented for the full system dynamics.

4 CONTRACTION THEORY

Dynamical systems describing the behavior of autonomous agents are essentially nonlinear. In contrast to the linear dynamical systems, a major difficulty of the analysis of stability properties of nonlinear is that the stability properties of parts usually do not transfer to composite systems. Contraction Theory [LS98] provides a general method for the analysis of essentially nonlinear systems, which permits such a transfer, making it suitable for the analysis of complex systems with many components. Contraction Theory characterizes the system stability by the behavior of the differences between solutions with different initial conditions. If these differences vanish exponentially over time, all solutions converge towards a single trajectory, independent from the initial states. In this case, the system is called *globally asymptotically stable*. For a general dynamical system of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) \tag{6}$$

assume that $\mathbf{x}(t)$ is one solution of the system, and $\tilde{\mathbf{x}}(t) = \mathbf{x}(t) + \delta\mathbf{x}(t)$ a neighboring one with a different initial condition. The function $\delta\mathbf{x}(t)$ is also called *virtual displacement*. With the Jacobian of the system $\mathbf{J}(\mathbf{x}, t) = \frac{\partial \mathbf{f}(\mathbf{x},t)}{\partial \mathbf{x}}$ it can be shown [LS98] that any nonzero virtual displacement decays exponentially to zero over time if the symmetric part of the Jacobian $\mathbf{J}_s = (\mathbf{J} + \mathbf{J}^T)/2$ is uniformly negative definite, denoted as $\mathbf{J}_s < 0$. This implies that it has only negative eigenvalues for all relevant state vectors $\mathbf{x}$. In this case, it can be shown that the norm of the virtual displacement decays at least exponentially to zero, for $t \to \infty$. If the virtual displacement is small enough, then

$$\dot{\delta\mathbf{x}}(t) = \mathbf{J}(\mathbf{x}, t)\delta\mathbf{x}(t)$$

implying through $\frac{d}{dt}||\delta\mathbf{x}(t)||^2 = 2\delta\mathbf{x}^T(t)\mathbf{J}_s(\mathbf{x}, t)\delta\mathbf{x}$ the inequality: $||\delta\mathbf{x}(t)|| \leq ||\delta\mathbf{x}(0)||\, e^{\int_0^t \lambda_{\max}(\mathbf{J}_s(\mathbf{x},s))\, ds}$. This implies that the virtual displacements decay with a *convergence rate* (inverse timescale) that is bounded from below by the quantity $\rho_c = -\sup_{\mathbf{x},t} \lambda_{\max}(\mathbf{J}_s(\mathbf{x}, t))$, where $\lambda_{\max}(.)$ signifies the largest eigenvalue. With $\rho_c > 0$ all trajectories converge to a single solution exponentially in time [LS98].

Contraction analysis can be applied also to hierarchically coupled systems [LS98]. Consider a composite dynamical system with two components, where the dynamics of the first subsystem is not influenced by the dynamics of the second one. Such system is called *hierarchically coupled*. The composite dynamical system:

$$\frac{d}{dt}\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{f}_1(\mathbf{x}_1) \\ \mathbf{f}_2(\mathbf{x}_1, \mathbf{x}_2) \end{pmatrix} \tag{7}$$

results in the Jacobian:

$$\mathbf{F} = \begin{pmatrix} \frac{\partial \mathbf{f}_1(\mathbf{x}_1)}{\partial \mathbf{x}_1} & 0 \\ \frac{\partial \mathbf{f}_2(\mathbf{x}_1,\mathbf{x}_2)}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{f}_2(\mathbf{x}_1,\mathbf{x}_2)}{\partial \mathbf{x}_2} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_{11} & 0 \\ \mathbf{F}_{21} & \mathbf{F}_{22} \end{pmatrix} \tag{8}$$

Consider then the smooth dynamics of virtual displacements: $\frac{d}{dt}\begin{pmatrix} \delta\mathbf{x}_1 \\ \delta\mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{F}_{11} & 0 \\ \mathbf{F}_{21} & \mathbf{F}_{22} \end{pmatrix}\begin{pmatrix} \delta\mathbf{x}_1 \\ \delta\mathbf{x}_2 \end{pmatrix}$, where $\mathbf{F}_{21}$ is bounded. The first subsystem does not depend on the second, so that $\delta\mathbf{x}_1$ exponentially converges to 0 if $(\mathbf{F}_{11})_s < 0$. Then, $\mathbf{F}_{21}\delta\mathbf{x}_1$ is an exponentially decaying disturbance for the second subsystem. In this case, (see [LS98] for details of proof), uniformly negative definite $\mathbf{F}_{22}$ implies exponential convergence of $\delta\mathbf{x}_2$ to an exponentially decaying ball. The whole system is then globally exponentially convergent to a single trajectory.

Many systems are not contracting with respect to all dimensions of the state space, but show convergence with respect to a subset of dimensions. Such behavior can be mathematically characterized by *partial contraction* [WS05], [PMSA09]. The underlying idea is to construct an auxiliary system that is contracting with respect to a subset of the arguments of the function $\mathbf{f}$. The major result is the following:

Theorem 1 *Consider a nonlinear system of the form*

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{x}, t) \tag{9}$$

and assume the existence of auxiliary system

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, \mathbf{x}, t) \tag{10}$$

that is contracting with respect to $\mathbf{y}$ *uniformly for all relevant* $\mathbf{x}$*. If a particular solution of this auxiliary system verifies a specific smooth property, then trajectories of the original system (9) verify this property exponentially. The original system is then said to be partially contracting. [WS05].*

A 'smooth property' is a property of the solution that depends smoothly on space and time, such as convergence against a particular solution or a properly defined distance to a subspace in phase space. The proof of the theorem is immediate noticing that the observer-like system (10) has $\mathbf{y}(t) = \mathbf{x}(t)$ for all $t \geq 0$ as a particular solution. Since all trajectories of the $\mathbf{y}$-system converge exponentially to a single trajectory, this implies that also the trajectory $\mathbf{x}(t)$ verifies this specific property with exponential convergence.

It is thus sufficient to show that the auxiliary system is contracting in order to prove the convergence to a subspace. Let us assume that system has a flow-invariant *linear subspace* $\mathscr{M}$, which is defined by the property that trajectories starting in this space always remain in it for arbitrary times ($\forall t : \mathbf{f}(\mathscr{M},t) \subset \mathscr{M}$). If matrix $\mathbf{V}$ is an orthonormal projection onto $\mathscr{M}^{\perp}$, then sufficient condition for global exponential convergence to $\mathscr{M}$ is:

$$\mathbf{V}\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)_{\mathrm{s}} \mathbf{V}^T < \mathbf{0}, \tag{11}$$

where smaller sign indicates that this matrix is negative definite (see [PS07, PMSA09]).

5 RESULTS

We derived contraction bounds for three scenarios that correspond to control dynamics with increasing levels of complexity.

1) Control of step phase without position control: The simplest case is a control of the phase within the step cycle of the walkers without simultaneous control of the position of the characters. Such simple control already permits to simulate interesting behaviors, such as soldiers synchronizing their step phases [PMSA09], [**Demo**[1]]. The underlying dynamics is given by (4) with $m_d = 0$. For N identical dynamical systems, with symmetric identical coupling gains k this dynamics can be written

$$\dot{\mathbf{x}}_i = \mathbf{f}(\mathbf{x}_i) + k \sum_{j \in \mathscr{N}_i} (\mathbf{x}_j - \mathbf{x}_i), \quad \forall i = 1,\ldots,N \tag{12}$$

where $\mathscr{N}_i$ defines the index set specifying the neighborhood in the coupling graph, i.e. the other subsystems or characters that are coupled with character i.

This type of symmetric coupling, where the interaction forces between subsystems depend only on the differences of the phase variables is called *diffusive coupling*. In this case, the *Laplacian matrix* of the coupling scheme is given by $\mathbf{L} = \mathbf{L}_G \bigotimes I_p$, where p is the dimensionality of the individual sub-systems, and where $\bigotimes$ signifies the *Kronecker product*. The Laplacian of the coupling graph is the matrix $\mathbf{L}_G$. The system then can be rewritten compactly as $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},t) - k\mathbf{L}\mathbf{x}$ with the concatenated phase variable $\mathbf{x} = [\mathbf{x}_1^T,...,\mathbf{x}_n^T]^T$. The Jacobian of this system is $\mathbf{J}(\mathbf{x},t) = \mathbf{D}(\mathbf{x},t) - k\mathbf{L}$, where the block-diagonal matrix $\mathbf{D}(\mathbf{x},t)$ has the Jacobians of the uncoupled components $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_i,t)$ as entries.

The dynamics has a flow-invariant linear subspace $\mathscr{M}$ that contains the particular solution $\mathbf{x}_1^* = \cdots = \mathbf{x}_n^*$. For this solution all state variables $\mathbf{x}_i$ are identical and thus in synchrony. In addition, for this solution the coupling term in equation (12) vanishes, so that the form of the solution is identical with the solution of the uncoupled systems $\dot{\mathbf{x}}_i = \mathbf{f}(\mathbf{x}_i)$. If $\mathbf{V}$ is a projection matrix onto the subspace $\mathscr{M}^{\perp}$, then, according to (11), the sufficient contraction condition for convergence toward $\mathscr{M}$ is given by $\mathbf{V}(\mathbf{D}(\mathbf{x},t) - k\mathbf{L})_s\mathbf{V}^T < \mathbf{0}$, [PMSA09]. This implies

$$\lambda_{\min}\left(\mathbf{V}(k\mathbf{L})_s\mathbf{V}^T\right) = k\lambda_{\mathbf{L}}^{+} > \sup_{\mathbf{x},t} \lambda_{\max}\left(\mathbf{D}_s\right)$$

with $\lambda_{\mathbf{L}}^{+}$ being the smallest non-zero eigenvalue of symmetrical part of the Laplacian $\mathbf{L}_s$. The maximal eigenvalue for the individual oscillator is $\sup_{\mathbf{x},t} \lambda_{\max}\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x},t)\right)$. The sufficient condition for global stability of the overall system is given by $k > \sup_{\mathbf{x},t} \lambda_{\max}\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x},t)\right)/\lambda_{\mathbf{L}}^{+}$. This implies the following minimum convergence rate: $\rho_{\mathrm{c}} = -\sup_{\mathbf{x},t} \lambda_{\max}(\mathbf{V}(\mathbf{D}(\mathbf{x},t) - \mathbf{L})_{\mathrm{s}}\mathbf{V}^T)$.

For the special case of (4) with $m_d = 0$ this implies the sufficient contraction conditions $k > 0$ and $(\mathbf{L}^{\phi})_s \geq 0$.

Different topologies of the coupling graphs result in different stability conditions, since for example $\lambda_{\mathbf{L}}^{+} = 2(1 - \cos(2\pi/N))$ for symmetric ring coupling, and $\lambda_{\mathbf{L}}^{+} = N$ for all-to-all coupling. (N is the number of avatars.) See [WS05] and [PMSA09] for details.

2) Speed control by variation of step frequency: The dynamics of this system is given by equations (2) and (3) for $m_z = 0$. Assuming arbitrary initial distances and phase offsets for different propagating characters, implying $G(\phi_i^0) = c_i$, $c_i \neq c_j$, for $i \neq j$, we redefine d_{ij} as $d_{ij} - (c_i - c_j)$ in (1), and accordingly redefine $\mathbf{c}$ in (2). Assuming this control dynamics, and two avatars i and j that follow a leading avatar, their phase trajectories converge to a single unique trajectory only if $c_i = c_j$. This is a consequence of the one-to-one correspondence between gait phase and position of the avatar that is given by equation (2). In all other cases the trajectories of the followers converge to one-dimensional, but distinct, attractors in phase-position space that are uniquely defined by c_i. These attractors correspond to a behavior where the follower's position oscillates around the position of the leader.

[1] www.uni-tuebingen.de/uni/knv/arl/avi/wscg/video0.avi

For the analysis of contraction properties we regard an auxiliary system obtained from (2) by keeping the terms which are only dependent on ϕ: $\dot{\phi} = -m_d \mathbf{L}^d G(\phi + \phi^0)$. The symmetrized Jacobian of this system projected to the orthogonal compliment of flow-invariant linear subspace $\phi_1^* + \phi_1^0 = \ldots = \phi_N^* + \phi_N^0$ determines whether this system is partially contracting. By virtue of a linear change of variables the study of the contraction properties of this system is equivalent to study the contraction properties of the dynamical system $\dot{\phi} = -m_d \mathbf{L}^d G(\phi)$ on trajectories converging towards its flow-invariant manifold, the linear subspace of $\phi_1^* = \ldots = \phi_N^*$.

In order to derive an asymptotic stability condition, we consider the following auxiliary system, corresponding to a part of (2): $\dot{\phi} = -m_d \mathbf{L}^d G(\phi)$. The Jacobian of this system is given by $\mathbf{J} = -m_d \mathbf{L}^d \mathbf{D}_g$, where $(\mathbf{D}_g)_{ii} = g(\phi_i) = G'(\phi_i) > 0$ is a strictly positive diagonal matrix. Exploiting diagonal stability theory [Per69], it is straightforward to demonstrate that the auxiliary system is globally asymptotically stable and its state converges to an attractor with $\phi_1^* = \ldots = \phi_N^*$ for any initial condition assuming $(\mathbf{L}^d)_s \geq 0$ and $m_d > 0$. The sufficient conditions for asymptotic stability are satisfied for all types of symmetric diffusive couplings with positive coupling strength. For the case of asymmetric coupling graphs with more general structure including negative feedback links some results on asymptotic stability have been provided in [SA08].

The sufficient conditions for (exponential) partial contraction towards flow-invariant subspace are, (see (11)): $\mathbf{V}\mathbf{J}_s(\phi)\mathbf{V}^T = -m_d \mathbf{V}\mathbf{B}(\phi)\mathbf{V}^T < 0$, introducing $\mathbf{B}(\phi) = \mathbf{L}^d\mathbf{D}_g + \mathbf{D}_g(\mathbf{L}^d)^T$ and $\mathbf{V}$ signifying the projection matrix onto the orthogonal complement of the flow-invariant linear subspace. For diffusive coupling with symmetric Laplacian the linear flow-invariant manifold $\phi_1^* = \ldots = \phi_N^*$ is also the null-space of the Laplacian. In this case, the eigenvectors of the Laplacian that correspond to positive eigenvalues can be used to construct the projection matrix $\mathbf{V}$. For $m_d > 0$ the contraction conditions are thus satisfied if $\mathbf{V}\mathbf{B}(\phi)\mathbf{V}^T = \mathbf{V}(\mathbf{L}^d\mathbf{D}_g + \mathbf{D}_g(\mathbf{L}^d)^T)\mathbf{V}^T > 0$ for any diagonal matrix $\mathbf{D}_g > 0$.

Next we prove the exponential contraction conditions for the particular case of symmetrical all-to-all coupling. In this case $\mathbf{L}^d = N\mathbf{I} - \mathbf{1}\mathbf{1}^T \geq 0$, where $\mathbf{I}$ is identity matrix of size N. Since $\mathbf{V}\mathbf{1} = \mathbf{1}^T\mathbf{V}^T = 0$, we obtain $\frac{1}{2}\mathbf{V}(\mathbf{L}^d\mathbf{D}_g + \mathbf{D}_g(\mathbf{L}^d)^T)\mathbf{V}^T = N\mathbf{V}(\mathbf{D}_g)\mathbf{V}^T > 0$ for $\mathbf{D}_g > 0$. A lower bound for the contraction rate is computed from the projected symmetrized Jacobian $\mathbf{V}\mathbf{J}_s(\phi)\mathbf{V}^T = -\frac{m_d}{2}\mathbf{V}\mathbf{B}(\phi)\mathbf{V}^T$. Contraction theory also permits to compute the guaranteed contraction rate $\rho_{min} = m_d \min_\phi (g(\phi))\lambda^+_{\mathbf{L}^d}$, with $\lambda^+_{\mathbf{L}^d} = N$ for all-to-all symmetric coupling.

For a general symmetric couplings with positive links (with equal coupling strength $m_d > 0$) we obtain the sufficient contraction condition as: $\lambda^+_{min}(\mathbf{L}^d)/\lambda^+_{max}(\mathbf{L}^d) > \max_\phi(|g(\phi) - mean(g(\phi))|)/mean(g(\phi))$, where mean value of $g(\phi)$ over the gait cycle period T is: $mean(g(\phi)) = 1/T \int_0^T g(\phi)d\phi$. This condition is derived from the fact that for symmetric (positive) matrices M_1 and M_2 for $M_1 - M_2 > 0$ it is sufficient to satisfy $M_1 > M_2$ (the last means $\lambda_{min}(M_1) > \lambda_{max}(M_2)$). This sufficient condition put the constraints on admissible topologies of the coupling scheme dependent on the smoothness of gait velocity function $g(\phi)$. Alternatively, it is possible to introduce low-pass filtering of the forward kinematics of walking characters in order to increase the smoothness of $g(\phi)$.

An illustration of these stability bounds if given by the [**Demo**[2]]; that shows convergent behavior of the characters when the contraction condition $m_d > 0, (\mathbf{L}^d)_s \geq 0$ is satisfied for all-to-all coupling. [**Demo**[3]] shows the divergent behavior of a group when this condition is violated when $m_d < 0$.

The same proof can be extended for **nonlinear control rules**. In this case the eigenfrequency is given by a nonlinear modification of the control rule in eq. (1), for character i coupled to character j as: $\omega_i = \omega_0 + m_d h(z_j - z_i + d_{ij})$, where the saturating nonlinear function h could be given, for example by $h(z) = 1/[1 + \exp(-\gamma z)]$ with $\gamma > 0$. This nonlinear function limits the range of admissible speeds for the controller. Using the same notations as above, the dynamics of a single follower that follows a leader at position $P(t)$ is given by: $\dot{\phi}(t) = \omega_0 + m_d h(P(t) - G(\phi(t)) + c)$. The Jacobian of this dynamics $\mathbf{J}_s = -m_d h' g(\phi) < 0$ is negative, which follows from $m_d > 0$, $g(\phi) > 0$ and taking into account $h'(z) = dh(z)/dz > 0, \forall z$, what guarantees contraction.

Again this dynamics can be extended for N avatars, resulting in the nonlinear differential equation system: $\dot{\phi}_i(t) = \omega_0 - m_d \sum_{j=1}^N K_{ij} h(G(\phi_i) - G(\phi_j) + d_{ij}), \forall i$. The Jacobian of the system is: $\mathbf{J}(\phi) = -m_d \mathbf{L}^d(\phi)\mathbf{D}_g$, where $\mathbf{L}^d_{ij}(\phi) = -K_{ij}h'(G(\phi_i) - G(\phi_j) + d_{ij})$, $\mathbf{L}^d_{ii}(\phi) = \sum_{j\neq i}^N K_{ij}h'(G(\phi_i) - G(\phi_j) + d_{ij})$, $d_{ii} = 0$, ($\mathbf{D}_g$ is defined as before). Furthermore, the even function $h'(z) > 0$ implies that the Laplacian $\mathbf{L}^d(\phi)$ is symmetric diagonally dominant and it stays positive semidefinite for any positive $K_{ij} > 0$, by Gershgorin's Theorem [HJ85]. This implies that the system is asymptotically stable, its solutions converging to an attractor. The analysis of exponential convergence requires further steps that exceed the scope of this paper.

3) Stepsize control combined with a control of step phase: The dynamics is given by equations (3) and (4) with $m_d = 0$. This dynamics defines a hier-

[2] www.uni-tuebingen.de/uni/knv/arl/avi/wscg/video1.avi
[3] www.uni-tuebingen.de/uni/knv/arl/avi/wscg/video2.avi

Figure 5: Self-organized reordering of a crowd with 16 characters. Control dynamics affects direction, distance and gait phase. See [**Demo**[7]].

archically coupled nonlinear system (see (7), Section 4), which is difficult to analyze with classical methods [LS98]. The dynamics for $\mathbf{z}(t)$ given by equation (3) is partially contracting in case of all-to-all coupling for any bounded external input $\phi(t)$, if $m_z > 0$, $\mathbf{L}^z \geq 0$ and $\omega(t) > 0$. These sufficient contraction conditions can be derived from the requirement of the positive-definiteness of the symmetrized Jacobian applying a similar technique as above. The Jacobian of this subsystem is $\mathbf{J}(\phi,\omega) = -m_z \mathbf{D}_g^z(\phi,\omega)\mathbf{L}^z$, with the diagonal matrix $(\mathbf{D}_g^z(\phi,\omega))_{ii} = \omega_i g(\phi_i + \phi_i^0) > 0$ that is positive definite since $g(\phi) > 0$ and $\omega > 0$. This subsystem is (exponentially) contracting and its relaxation rate is determined by $\rho_z = m_z \min_\phi (g(\phi))\lambda^+_{\mathbf{L}^z}$ (in the case of all-to-all coupling) for any input from the dynamics of $\phi(t)$ eq. (4). The last dynamics is contracting when $(\mathbf{L}^\phi)_s \geq 0$ and its relaxation rate is $\rho_\phi = k\lambda^+_{\mathbf{L}^\phi}$, where $\lambda^+_{\mathbf{L}^\phi}$ is the smallest non-zero eigenvalue of $(\mathbf{L}^\phi)_s$. The effective relaxation time of the overall dynamics is thus determined by the minimum of the contraction rates ρ_ϕ and ρ_z.

Demonstrations of this control dynamics satisfying the contraction conditions are shown in [**Demo**[4]], without control of step phase, and in [**Demo**[5]], with control of step phase.

4) Advanced scenarios: A simulation of a system with stable dynamics with both types of speed control (via step size and step frequency) and step phase control is shown in [**Demo**[6]]. Using the same dynamics, a larger crowd of 16 avatars simulated with the open-source animation engine Horde3d [Sch09] is shown in [**Demo**[7]]. In this scenario, dynamic obstacle avoidance and control of heading direction were activated in an initial time interval for unsorting of a formation of avatars. In a second time interval navigation is deactivated, and speed and position control according to the discussed principles takes over. [**Demo**[8]] demonstrates a large synchronizing crowd with 36 avatars without initial reordering. [**Demo**[9]] shows the divergence dynamics of the crowd from previous example, when negative distance-to-eigenfrequency coupling strength used ($m_d < 0$). Two videos [**Demo**[10]] and [**Demo**[11]] show convergence dynamics of the crowd of 49 avatars for two different values of the strength of the distance-to-step size coupling (slow and fast). The coupling strength for the phase coupling dynamics is constant for this example. The development of stability bounds and estimates of relaxation times for even more advanced scenarios including multiple control levels of navigation is the goal of ongoing work.

6 CONCLUSION

For the example of a learning-based system for the animation of locomoting groups, we have demonstrated first applications of Contraction Theory for the analysis and the design of stability and convergence properties of collective behaviors in animated crowds. The dynamics of the collective behavior of animated crowds is highly nonlinear and prevents the stability analysis using classical approaches. Opposed to these approaches, Contraction theory allows to transfer stability results from the components to composite systems. Future work has to extend this approach to more complex scenarios, especially including non-periodic movements.

ACKNOWLEDGEMENTS

Supported by DFG Forschergruppe 'Perceptual Graphics', the EC FP7/2007-2013 projects 'AMARSI' (grant agreement No.248311) and 'TANGO' and the Hermann und Lilly-Schilling-Stiftung. Authors thank Karsten Rohweder for help with animations in Horde3d.

REFERENCES

[AVK87] A. A. Andronov, A. A. Vitt, and S. E. Khaikin. *Theory of oscillators*. Dover Publ. Inc., New York, 1987.

[BC89] A. Bruderlin and T. W. Calvert. Goal-directed, dynamic animation of human walking. *Proc. of the 16th Conf. on Computer graphics and interactive techniques, ACM SIGGRAPH*, pages 233–242, 1989.

[BH97] D. C. Brogan and J. K. Hodgins. Group behaviors for systems with significant dynamics. *Autonomous Robots*, pages 137–153, 1997.

[4] www.uni-tuebingen.de/uni/knv/arl/avi/wscg/video3.avi
[5] www.uni-tuebingen.de/uni/knv/arl/avi/wscg/video4.avi
[6] www.uni-tuebingen.de/uni/knv/arl/avi/wscg/video5.avi
[7] www.uni-tuebingen.de/uni/knv/arl/avi/wscg/video6.avi
[8] www.uni-tuebingen.de/uni/knv/arl/avi/wscg/video7.avi
[9] www.uni-tuebingen.de/uni/knv/arl/avi/wscg/video8.avi

[10] www.uni-tuebingen.de/uni/knv/arl/avi/wscg/video9.avi
[11] www.uni-tuebingen.de/uni/knv/arl/avi/wscg/video10.avi

[BRI06] J. Buchli, L. Righetti, and A. J. Ijspeert. Engineering entrainment and adaptation in limit cycle systems - from biological inspiration to applications in robotics. *Biol. Cyb.*, 95, 6:645–664, 2006.

[CDF+01] S. Camazine, J. L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organization in Biological Systems*. Princeton University Press, New Jersey, 2001.

[Cou09] I. D. Couzin. Collective cognition in animal groups. *Trends in Cogn. Sci.*, 13, 1:1–44, 2009.

[CS93] J. J. Collins and I. N. Stewart. Coupled nonlinear oscillators and the symmetries of animal gaits. *J. Nonlinear Sci.*, 3:349–392, 1993.

[CS07] F. Cucker and S. Smale. Emergent behavior in flocks. *IEEE Trans. Automat. Control*, 52, 5:852–862, 2007.

[DH03] W. Daamen and S. P. Hoogendoorn. Controlled experiments to derive walking behaviour. *European Journal of Transport and Infrastructure Research*, 3, 1:39–59, 2003.

[GMP+09] M. A. Giese, A. Mukovskiy, A. Park, L. Omlor, and J. J. E. Slotine. Real-time synthesis of body movements based on learned primitives. In D. Cremers et al., editor, *Stat. and Geom. Appr. to Vis. Mot. Anal., LNCS5604*, pages 107–127. Springer, 2009.

[GRIL08] A. Gams, L. Righetti, A. J. Ijspeert, and J. Lenarcic. A dynamical system for online learning of periodic movements of unknown waveform and frequency. *Proc. of the IEEE RAS / EMBS Int. Conf. on Biomedical Robotics and Biomechatronics*, pages 85–90, 2008.

[GTH98] R. Grzeszczuk, D. Terzopoulos, and G. Hinton. Neuroanimator: Fast neural network emulation and control of physics based models. *Proc. ACM SIGGRAPH, Int. Conf. on Comp. Graph. and Interactive Techniques*, pages 9–20, 1998.

[HJ85] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, 1985.

[HMFB01] D. Helbing, P. Molnár, I. J. Farkas, and K. Bolay. Self-organizing pedestrian movement. *Environment and Planning B: Planning and Design*, 28:361–383, 2001.

[Ijs08] A. J. Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21, 4:642–653, 2008.

[KLLT08] T. Kwon, K. H. Lee, J. Lee, and S. Takahashi. Group motion editing. *ACM Transactions on Graphics, SIGGRAPH 2008*, 27, 3:80–87, 2008.

[LFCCO09] A. Lerner, E. Fitusi, Y. Chrysanthou, and D. Cohen-Or. Fitting behaviors to pedestrian simulations. *Proc. Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pages 199–208, 2009.

[LS98] W. Lohmiller and J. J. E. Slotine. On contraction analysis for nonlinear systems. *Automatica*, 34, 6:683–696, 1998.

[MT01] S. R Musse and D. Thalmann. A behavioral model for real time simulation of virtual human crowds. *IEEE Trans. on Vis. and Comp. Graph.*, 7, 2:152–164, 2001.

[NGCL09] R. Narain, A. Golas, S. Curtis, and M. Lin. Aggregate dynamics for dense crowd simulation. *ACM Transactions on Graphics, Art.122*, 28, 5:1–8, 2009.

[OG06] L. Omlor and M. A. Giese. Blind source separation for over-determined delayed mixtures. *Adv. in NIPS*, 19:1049–1056, 2006.

[PAB07] N. Pelechano, J. M. Allbeck, and N. I. Badler. Controlling individual agents in high-density crowd simulation. *Proc. Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, pages 99–108, 2007.

[Per69] S. K. Persidskii. Problem of absolute stability. *Automat. Remote Control*, 12:1889–1895, 1969.

[PLS+07] D. A. Paley, N. E. Leonard, R. Sepulchre, D. Grunbaum, and J. K. Parrish. Oscillator models and collective motion: Spatial patterns in the dynamics of engineered and biological networks. *IEEE Control Systems Magazine*, 27:89–105, 2007.

[PMSA09] A. Park, A. Mukovskiy, J. J. E. Slotine, and Giese M. A. Design of dynamical stability properties in character animation. *Proc. of VRIPHYS 09*, pages 85–94, 2009.

[PPS07] S. Paris, J. Pettré, and Donikian S. Pedestrian reactive navigation for crowd simulation: a predictive approach. *Proc. Eurographics 2007*, 26, 3:665–674, 2007.

[PRK03] A. Pikovsky, M. Rosenblum, and J. Kurths. *Synchronization, A Universal Concept in Nonlinear Sciences*. Cambridge University Press, Cambridge, 2003.

[PS07] Q. C. Pham and J. J. E. Slotine. Stable concurrent synchronization in dynamic system networks. *Neural Networks*, 20, 3:62–77, 2007.

[Rey87] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21, 4:25–34, 1987.

[RI06] L. Righetti and A. J. Ijspeert. Programmable central pattern generators: an application to biped locomotion control. *Proc. of the 2006 IEEE Int. Conf. on Rob. and Autom.*, pages 1585–1590, 2006.

[SA08] E. D. Sontag and M. Arcak. Passivity-based stability of interconnection structures. In V. et al. Blondel, editor, *Rec. Adv. in Learning and Control. Vol.371*, pages 195–204. Springer-Verlag, NY, 2008.

[Sch09] N. Schulz. http://www.horde3d.org/. 2006–2009.

[SDE95] G. Sch´oner, M. Dose, and C. Engels. Dynamics of behavior: Theory and applications for autonomous robot architectures. *Robotics and Autonomous Systems*, 16, 2-4:213–245, 1995.

[Slo03] J. J. E. Slotine. Modular stability tools for distributed computation and control. *Int. J. Adaptive Control and Signal Processing*, 17, 6:397–416, 2003.

[SS06] L. Scardovi and R. Sepulchre. Collective optimization over average quantities. *Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, California*, pages 3369–3374, 2006.

[TCP06] A. Treuille, S. Cooper, and Z. Popović. Continuum crowds. *Proc. ACM SIGGRAPH '06*, 25, 3:1160–1168, 2006.

[WS05] W. Wang and J. J. E. Slotine. On partial contraction analysis for coupled nonlinear oscillators. *Biological Cybernetics*, 92, 1:38–53, 2005.

Evaluation of the Linear Box-Spline Filter from Trilinear Texture Samples: A Feasibility Study

Balázs Domonkos
Mediso Medical Imaging Systems, Hungary
balazs.domonkos@mediso.hu

Balázs Csébfalvi
Budapest University of Technology and Economics
cseb@iit.bme.hu

ABSTRACT

The major preference for applying B-spline filtering rather than non-separable box spline filtering on the BCC lattice is the fact that separable filtering can be performed more efficiently on current GPUs due to the utilization of the hardware-accelerated trilinear texture fetching. In order to make a fair comparison, a similar, efficient evaluation scheme is required that uses trilinear texture fetches instead of nearest-neighbor ones also for the box splines. Thus, in this paper, we propose an evaluation scheme for the linear BCC box spline built upon a trilinear B-spline basis. We compare our trilinearly evaluated linear box spline scheme to the latest method, that uses twice as many nearest neighbor fetches. Then we give a comparison to the major competitive methods: the BCC B-spline filtering and the BCC DC-spline filtering in terms of their performance.

Keywords: Volume Rendering, Filtering, Reconstruction.

1 INTRODUCTION

In many applications in engineering and computing science, a continuous phenomenon is represented by its discrete samples. In order to operate on the underlying continuous function, first it has to be accurately reconstructed from its discrete representation. Reconstruction filters have received attention also in image processing and volume visualization since appropriate reconstruction of multivariate functions is a key step of the processing pipeline [2, 3, 17, 18].

According to the most commonly-used sampling scheme in practice, volumetric data is often acquired on a uniform lattice by regular sampling, while reconstruction is performed by convolution filtering. An appropriate choice of both the sampling lattice and the reconstruction filter kernel is of crucial importance as they together directly determine the quality of the reproduced continuous function and the efficiency of the reconstruction.

Recent results advocate the benefits of non-Cartesian lattices for regular sampling. The application of Body-Centered Cubic (BCC) sampling received increased attention from the perspective of continuous signal reconstruction in the last decade [5, 6, 11, 12]. This lattice is optimal for sampling 3D signals of isotropic bandwidth [19, 21], unlike the commonly used Cartesian Cubic (CC) lattice along with tensor-product reconstruction. To perfectly reconstruct a signal of a spherically bounded spectrum from its discrete representation, roughly 30% fewer samples per unit volume have to be taken on a BCC lattice than on an equivalent CC lattice. In addition to the improved spectral isotropy, this directly translates into an explicit reduction of the storage cost.

A crucial question of BCC sampling is the way in which the original continuous signal is reconstructed from its discrete samples. Although due to the shift-invariant property of the sampling lattice, the reconstruction can be implemented by a simple convolution, the choice of the filter kernel has a direct impact on both numerical accuracy and visual quality. Generally, an appropriate filter is chosen by making a compromise between quality and efficiency.

Currently, three promising resampling techniques exist for the BCC lattice that provide high visual quality, numerical accuracy, and efficiency at the same time: the box splines [11], the BCC B-splines [8, 6], and the BCC DC-splines [10]. As only the latter two methods can exploit the hardware-accelerated trilinear filtering, it has not been possible to make a fair comparison so far. To remedy this problem, we propose an algorithm that uses trilinear fetches for the box spline filtering as well. Since these filters have already been compared in terms of visual quality and numerical accuracy [6, 10, 13], in this paper, we focus on a fair comparison of their performance.

2 RELATED WORK

One of the most important aspects of rendering sampled data is how to perform proper and efficient resampling depending on the applied lattice. For the CC lattice, reconstruction filters are usually designed in 1D, and then

Plzen, Czech Republic.

extended to the trivariate setting by a separable tensor-product extension. However, the BCC lattice is not separable itself, therefore the advantageous properties of a 1D filter are not necessarily inherited in 3D by a separable extension [21, 22, 15].

The first reconstruction filters tailored to the geometry of the non-Cartesian lattices were proposed by Entezari et al. [11]. They applied *box splines*, that offer a mathematically elegant toolbox for constructing a class of multidimensional elements with flexible shape and support. Box splines are often considered as a generalization of B-splines to multivariate setting. Theoretically, the computational complexity of a box spline is lower than that of an equivalent B-spline, since its support is more compact and its total polynomial degree is lower. To investigate this potential also in practice, several attempts were made. Although de Boor's recurrence relation [9] is the most commonly used technique for evaluating box splines at an arbitrary position, it is computationally inefficient and has numerical instabilities [14]. Addressing this issue, Entezari et al. [12] derived a piecewise-polynomial representation of the linear and quintic box splines for the BCC lattice. In a CPU-based implementation, due to the smaller support of the box spline kernels, the data access cost of discrete BCC samples turned out to be twice as low as for the equivalent B-spline filters on the CC lattice [12]. Following their work, Finkbeiner et al. proposed an algorithm to convolve the BCC samples with these box spline kernels [13]. Though they applied early selection of polynomial segments of the piecewise polynomial form that enabled them to avoid a full kernel evaluation for each affected sample point, the theoretical advantages of box splines could not be exploited on the GPUs, which are rather optimized for separable filtering.

Another family of non-separable filters is represented by the *Voronoi splines* [16] that inherit the geometry of a sampling lattice through its Voronoi cell. For Cartesian lattices, Voronoi splines coincide with tensor-product B-splines. For the 2D hexagonal lattice, Voronoi splines were originally proposed by Van de Ville et al. [23] as Hex-splines. For the BCC lattice Voronoi splines were derived as BCC-splines by Csébfalvi [5]. Recently, Mirzargar et al. [16] formulated the BCC-splines in terms of multi-box splines. In spite of their theoretical elegance, Voronoi splines are currently impractical, since their piecewise evaluation is not known yet.

Csébfalvi recommended a *prefiltered Gaussian reconstruction scheme* [4] adapting the principle of generalized interpolation [1] to the BCC lattice. According to this approach, first a non-separable discrete prefiltering is performed as a preprocessing step, and afterwards a fast separable Gaussian filtering is used for continuous resampling on the fly. This method was extended also to the *B-spline family of filters* [8]. An efficient GPU implementation was proposed exploiting the fact that the BCC lattice consists of two interleaved CC lattices, where the second CC lattice is translated by half of the grid spacing. The reconstruction can be performed separately for these two CC lattices in the given sample position by using a standard trilinear or tricubic B-spline resampling, and then the contributions are averaged [8]. BCC B-splines reconstruction was reported to be four to five times faster on an NVIDIA GeForce 6800 graphics card than a non-separable box spline reconstruction of the same approximation power [6], since the B-splines can utilize the hardware-accelerated trilinear texture fetching [20].

Recently, Domonkos et al. [10] proposed a *discrete/-continuous filter family* generated by the impulse response of the BCC trilinear kernel. This technique is theoretically equivalent to the discrete upsampling of the BCC-sampled volume on a higher resolution CC lattice, where the standard trilinear interpolation is used for resampling. In practice, however, the missing CC samples are calculated on the fly and not in a preprocessing. Using an optimized GPU implementation, the linear DC-spline was reported to be slightly faster than the linear box spline.

3 SPLINE RECONSTRUCTION FOR THE BCC LATTICE

In the following, we briefly review the main properties of the BCC lattice, as well as the box spline, B-spline, and DC-spline family of filters, as they are applied for reconstruction on the BCC lattice.

3.1 BCC Lattice

The BCC lattice Λ_{BCC} is a discrete subgroup of $\mathbb{R}^3$ generated by integer linear combinations of the following basis vectors:

$$\begin{aligned} \Xi_{BCC} &= [\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_3] = \frac{1}{2}\begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix} \\ \Lambda_{BCC} &= \left\{ \Xi_{BCC}\mathbf{i} \,:\, \mathbf{i} \in \mathbb{Z}^3 \right\} \subset \mathbb{R}^3. \end{aligned}$$

Besides, the BCC lattice points are located on a CC lattice with an additional sample placed in the center of each cube. Thus, the BCC lattice can also be considered as two interleaved CC lattices Λ_{CC_A} and Λ_{CC_B}. By shifting the secondary CC lattice Λ_{CC_B} by half of the grid spacing, the vertices of the secondary CC lattice are moved to the centers of the primary CC cells:

$$\begin{aligned} \Lambda_{BCC} &= \Lambda_{CC_A} \cup \Lambda_{CC_B} \\ \Lambda_{CC_A} &= \left\{ \mathbf{i} \,:\, \mathbf{i} \in \mathbb{Z}^3 \right\} \\ \Lambda_{CC_B} &= \left\{ \mathbf{i} + \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \end{bmatrix} \,:\, \mathbf{i} \in \mathbb{Z}^3 \right\}. \end{aligned} \tag{1}$$

On the other hand, the BCC lattice can be obtained also from a dense CC lattice by keeping only the lattice points whose coordinates have identical parity:

$$\Lambda_{BCC} = \left\{ \frac{1}{2} \begin{bmatrix} i \\ j \\ k \end{bmatrix} \quad : \; i \equiv j \equiv k \pmod 2 \atop i,j,k \in \mathbb{Z} \right\}. \quad (2)$$

3.2 Box Splines for the BCC Lattice

A box spline M_{Ξ} is the shadow of a unit-hypercube in $\mathbb{R}^n$ projected to $\mathbb{R}^s, s \leq n$ where the projection is characterized by $\Xi = [\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_n] \in \mathbb{R}^{s \times n}$, $\boldsymbol{\xi}_i \in \mathbb{R}^s \backslash \mathbf{0}$ [9]. The shape, the continuity order, and the approximation power of a given box spline M_{Ξ} is determined by Ξ. The simplest box spline is constructed when $s = n$ as a normalized characteristic function of its support:

$$M_{\Xi}(\mathbf{x}) = \begin{cases} \frac{1}{\det \Xi} & \text{if } \Xi^{-1}\mathbf{x} \in [0,1)^n \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

When adding a further direction vector $\boldsymbol{\xi} \in \mathbb{R}^s$ to Ξ, $s < n$, the box spline $M_{[\Xi,\boldsymbol{\xi}]}$ is given by the convolution:

$$M_{[\Xi,\boldsymbol{\xi}]}(\mathbf{x}) = \int_0^1 M_{\Xi}(\mathbf{x} - t\boldsymbol{\xi})\, dt. \quad (4)$$

The linear box spline $M_{\Xi^1_{BCC}} \in C^0$ for the BCC lattice is constructed as a 3D shadow of a tesseract along its antipodal axis, resulting a function with a rhombic dodecahedron support, which is the first neighbors cell of the BCC lattice [12]:

$$\Xi^1_{BCC} = \begin{bmatrix} & & 1/2 \\ \Xi_{BCC} & & 1/2 \\ & & 1/2 \end{bmatrix}. \quad (5)$$

$M_{\Xi^1_{BCC}}$ has its maximum value at the center, and has a linear falloff towards the 14 first-neighbor vertices:

$$M_{\Xi^1_{BCC}}(\mathbf{x}) = \max(1 - x - y,\, 0), \quad (6)$$

where x is the largest and y is the second largest component of the absolute coordinates of $\mathbf{x}$ [12].

3.3 B-Splines for the BCC Lattice

The B-spline of order zero is defined as a box filter:

$$\beta^0(t) = \begin{cases} 1 & \text{if } |t| < \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

Generally, the B-spline filter of order n is derived by successively convolving $\beta^0(t)$ n times with itself. The first-order B-spline is the linear interpolation filter or tent filter:

$$\beta^1(t) = \beta^0(t) * \beta^0(t) = \begin{cases} 1 - |t| & \text{if } |t| \leq 1 \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

The 1D B-splines can be extended to the 3D CC lattice by a tensor product extension. BCC B-spline resampling exploits the decomposition property of the BCC lattice (Eq. 1). The reconstruction is performed separately for the two CC sub-lattices in the given sample position by using a standard separable CC B-spline resampling, and then the contributions are simply averaged [8, 6]. This evaluation is equivalent to the convolution of the BCC samples with a B-spline kernel.

3.4 DC-Splines for the BCC Lattice

The BCC lattice can be obtained from a CC lattice by removing the lattice points whose coordinates have different parity (Eq. 2). The BCC trilinear interpolation reproduces these "missing CC samples" by interpolating between the available BCC samples on the fly using a discrete filter. The resultant impulse response χ^1_{BCC} of the linear BCC DC-spline is obtained by convolving this discrete filter with a scaled trilinear kernel $\beta^1(2\mathbf{x})$:

$$\chi^1_{BCC}(\mathbf{x}) = \beta^1(2\mathbf{x}) + \frac{1}{2}\sum_{k=1}^{6} \beta^1(2(\mathbf{x} - \nu_{\mathbf{k}})) \quad (9)$$

$$[\nu_{\mathbf{1}\ldots\mathbf{6}}] = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

4 EVALUATION OF THE LINEAR BOX SPLINE FROM TRILINEAR TEXTURE SAMPLES

The major preference for applying the BCC B-spline filtering over the non-separable box spline filtering is the fact that separable filtering can be performed significantly faster on current GPUs due to the utilization of the hardware-accelerated trilinear texture fetching [20].

In order to make a fair comparison, an efficient evaluation scheme is required that uses trilinear texture fetches instead of nearest neighbor ones also for the box splines. In the following, we propose an algorithm for evaluation of the linear BCC box spline built upon a trilinear B-spline basis.

According to Eq. 6, the support of $M_{\Xi^1_{BCC}}$ covers four BCC samples that form a tetrahedron, thus the B-form of resampling is [11]:

$$f(\mathbf{r}) = \sum_{i=1}^{4} s(\mathbf{r}_i) M_{\Xi^1_{BCC}}(\mathbf{r_i} - \mathbf{r}), \quad (10)$$

where $\mathbf{r}$ is an arbitrary resampling point and s is a 3D array of the discrete BCC samples. Direct implementation of this B-form is rather inefficient, since a full kernel evaluation is performed for each $\mathbf{r_i}$ sample point [13].

A more efficient piecewise-polynomial evaluation scheme can be set up, since it is possible to evaluate the ordering of the absolute coordinates of $\mathbf{r_i} - \mathbf{r}$ in advance for each $\mathbf{r_i}$ lattice points [13].

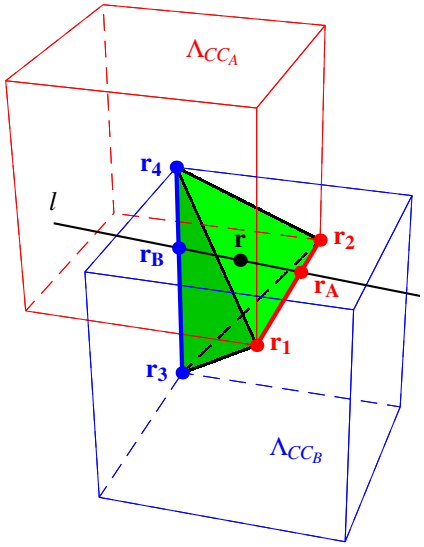


Figure 1: Trilinear evaluation scheme. For an arbitrary point $\mathbf{r}$, interpolation is performed within the green tetrahedron formed by the nearest points $\mathbf{r_1}, \mathbf{r_2} \in \Lambda_{CC_A}$ of the red CC lattice and the nearest points $\mathbf{r_3}, \mathbf{r_4} \in \Lambda_{CC_B}$ of the blue CC lattice. When $\mathbf{r}$ is an internal point, that is, $\mathbf{r} \notin \mathbf{r_{1,2}}$ and $\mathbf{r} \notin \mathbf{r_{3,4}}$, there is exactly one line l that intersects $\mathbf{r}$, and edges $\mathbf{r_{1,2}}$ and $\mathbf{r_{3,4}}$.

4.1 Trilinear Evaluation Scheme

The key point of the derivation lies in the fact that the linear box spline constitutes a linear interpolator on the BCC lattice [11]. This enables us to evaluate the linear interpolation within the tetrahedron more efficiently than a direct evaluation of Eq. 10.

The first observation we make is that the tetrahedron is composed of four congruent isosceles triangles (see Fig. 1):

1. $\mathbf{r_{1,2,3}}$ 2. $\mathbf{r_{1,2,4}}$ 3. $\mathbf{r_{3,4,1}}$ 4. $\mathbf{r_{3,4,2}}$

Four edges of the tetrahedron are formed by the equal sides of these triangles with the length of $\frac{\sqrt{3}}{2}$ while the remaining two edges of the tetrahedron are formed by the sides $\mathbf{r_{1,2}}$ and $\mathbf{r_{3,4}}$ of the triangles with the length of 1:

$$\begin{aligned} \frac{\sqrt{3}}{2} &= |\mathbf{r_{1,3}}| = |\mathbf{r_{2,3}}| = |\mathbf{r_{1,4}}| = |\mathbf{r_{2,4}}| \\ 1 &= |\mathbf{r_{1,2}}| = |\mathbf{r_{3,4}}| \end{aligned}$$

The edges $\mathbf{r_{1,2}}$ and $\mathbf{r_{3,4}}$ overlap the edges of the BCC lattice. Moreover, when the BCC lattice is considered as two interleaved CC lattices (Eq. 1), edge $\mathbf{r_{1,2}}$ is contained by the first CC lattice Λ_{CC_A}, while edge $\mathbf{r_{3,4}}$ is contained by the second CC lattice Λ_{CC_B}.

This enables us to rewrite the tetrahedral interpolation as the compound of three linear interpolations using the following scheme:

1. First, we define line l that contains $\mathbf{r}$ and intersects both $\mathbf{r_{1,2}} \in \Lambda_{CC_A}$ and $\mathbf{r_{3,4}} \in \Lambda_{CC_B}$ (see Fig. 1). The intersection points with edges $\mathbf{r_{1,2}}$ and $\mathbf{r_{3,4}}$ are $\mathbf{r_A}$ and $\mathbf{r_B}$, respectively. This decouples the BCC resampling problem into resamplings of two separate CC lattices, to Λ_{CC_A} and Λ_{CC_B}.

2. Next, the discrete data is resampled in $\mathbf{r_A}$ for Λ_{CC_A} and $\mathbf{r_B}$ for Λ_{CC_B} using a simple linear kernel:

$$\begin{aligned} f_A &= s_A(\mathbf{r_1} + |\mathbf{r_{1,A}}|\mathbf{r_{1,2}}) \\ f_B &= s_B(\mathbf{r_3} + |\mathbf{r_{3,B}}|\mathbf{r_{3,4}}), \end{aligned} \tag{11}$$

   where s_A and s_B are linearly addressable 3D arrays of the discrete CC samples corresponding to Λ_{CC_A} and Λ_{CC_B}, respectively.

3. Finally, the linear combination of the two CC samples is calculated:

$$f(\mathbf{r}) = f_A + \frac{|\mathbf{r} - \mathbf{r_A}|}{|\mathbf{r_{A,B}}|}(f_B - f_A). \tag{12}$$

The clear advantage of this evaluation scheme is that Step 2 can be performed by only two trilinear fetches on the GPU instead of four nearest neighbor fetches. Actually, these trilinear fetches involve in fact only 1D linear interpolations since $\mathbf{r_A}$ and $\mathbf{r_B}$ lie on a lattice edge. Regarding the storage scheme, the consequence is that the BCC samples need to be stored in two separate CC lattices, i.e. conventional 3D textures, to be able to exploit the trilinear fetching capability of the GPU just like in case of the BCC B-spline and the BCC DC-spline.

4.2 Orientation Cases

Addressing $\mathbf{r_1}$, $\mathbf{r_2}$, $\mathbf{r_3}$, and $\mathbf{r_4}$ for an arbitrary $\mathbf{r}$ is required in Step 1 which needs some further explanation. Let $\mathbf{r_{base}} = \text{round}(\mathbf{r})$ be the nearest lattice point in Λ_{CC_A} and let $\mathbf{d} = \mathbf{r} - \mathbf{r_{base}}$ be the relative resampling coordinates with their absolute values $\mathbf{a} = [|d_x|, |d_y|, |d_z|]^T \in [0, \frac{1}{2})^3$ and their signs $\mathbf{s} = [\text{sgn}(d_x), \text{sgn}(d_y), \text{sgn}(d_z)]^T$. Considering the symmetries of the rhombic dodecahedral support of $M_{\Xi^1_{BCC}}$, six different orientations of the resampling tetrahedron can be distinguished (see Fig. 2). These six cases are the 3! possible orderings of the absolute coordinates $\mathbf{a}$ in Eq. 6 as it was reported in [13].

Since using any control flow statement in the resampling implementation dramatically cuts the performance of the GPUs which have a SIMD architecture, it is advisable to avoid this six-fold branching. Descending order of three scalars can be calculated in a SIMD-aware manner as:

$$\begin{aligned} x &= max(a_x, a_y, a_z) \qquad z = min(a_x, a_y, a_z) \\ y &= a_x + a_y + a_z - x - z. \end{aligned} \tag{13}$$

On the other hand, based on the sort order of a_x, a_y, and a_z all the six orientations of the resampling tetrahedron can be transformed back to the first one (the green

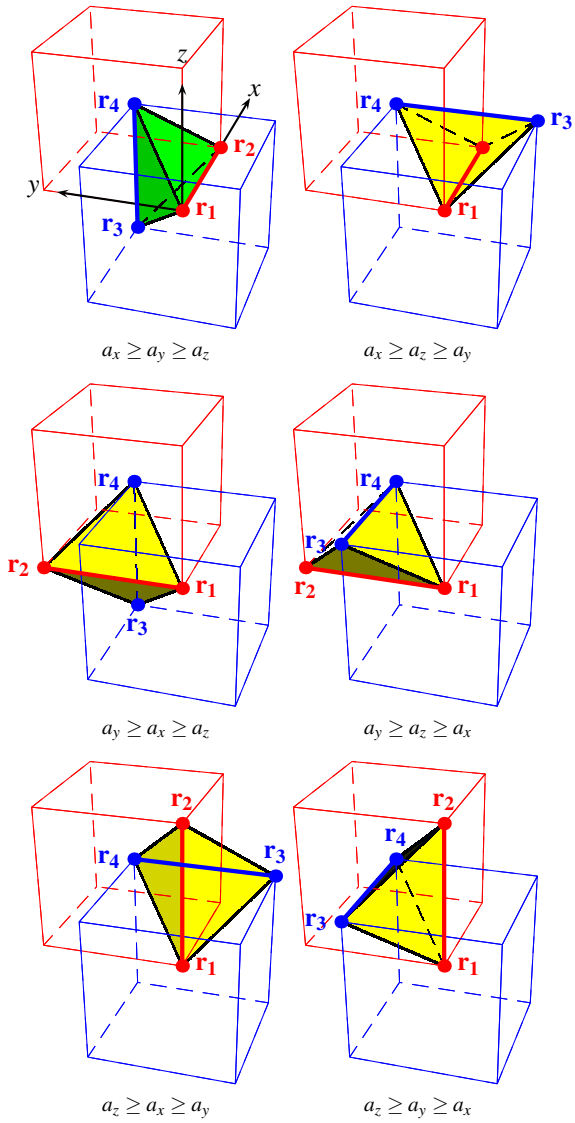


Figure 2: There are six orientation cases for ordering the coordinates of $\mathbf{a} \in [0, \frac{1}{2})^3$. The required resampling points $\mathbf{r_1}, \mathbf{r_2} \in \Lambda_{CC_A}$ and $\mathbf{r_3}, \mathbf{r_4} \in \Lambda_{CC_B}$ are determined by these six cases. These resampling points are indicated as red and blue dots for each orientation case.

tetrahedron for $a_x \geq a_y \geq a_z$ in Fig. 2). Thus, the resampling formula needs to be written only for the first orientation case, and the other cases can be retrieved by using this transformation. The transformation can be defined by a rotation matrix $\mathbf{\Pi}$ as

$$\Pi_{i,j} = s_i \cdot e_{\pi(j),i}, \tag{14}$$

where $\mathbf{e}_{\pi(1)}$, $\mathbf{e}_{\pi(2)}$, and $\mathbf{e}_{\pi(3)}$ are the unit vectors corresponding to x, y, and z, respectively (Eq. 13). As a compact notation, π represents the descending order of a_x, a_y, and a_z as a permutation. By using $\mathbf{\Pi}$, the lattice points can be addressed as

$$\mathbf{r_1} = \mathbf{r_{base}} + \mathbf{\Pi}\,[0\;0\;0]^T \qquad \mathbf{r_2} = \mathbf{r_{base}} + \mathbf{\Pi}\,[1\;0\;0]^T$$
$$\mathbf{r_3} = \mathbf{r_{base}} + \mathbf{\Pi}\left[\frac{1}{2}\;\frac{1}{2}\;-\frac{1}{2}\right]^T \qquad \mathbf{r_4} = \mathbf{r_{base}} + \mathbf{\Pi}\left[\frac{1}{2}\;\frac{1}{2}\;\frac{1}{2}\right]^T.$$

4.3 Formal Derivation

In the following, we also give a formal derivation of the proposed algorithm. The derivation is based on the rewriting of the tetrahedral interpolation in barycentric coordinates. Barycentric coordinates provide a convenient way for interpolation on a tetrahedral mesh:

$$f(\mathbf{r}) = \sum_{i=1}^{4} \lambda_i s(\mathbf{r_i}), \tag{15}$$

where scalars $\lambda_{1\ldots4}$ are barycentric coordinates of $\mathbf{r}$ with respect to the vertices of the tetrahedron $\mathbf{r_{1\ldots4}}$ under the constraint $\sum_{i=1}^{4} \lambda_i = 1$. The barycentric expansion of $\mathbf{r}$ is set up in terms of the vertices of the tetrahedron as:

$$\begin{aligned} \mathbf{T\lambda} &= \mathbf{r} - \mathbf{r_4} \\ \mathbf{T} &= [\mathbf{r_1} - \mathbf{r_4} \mid \mathbf{r_2} - \mathbf{r_4} \mid \mathbf{r_3} - \mathbf{r_4}] \\ \boldsymbol{\lambda} &= \left[\;\lambda_1 \quad \lambda_2 \quad \lambda_3\;\right]^T. \end{aligned} \tag{16}$$

The solution of this linear equation system is

$$\mathbf{T} = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & -\frac{1}{2} & -1 \end{bmatrix}, \quad \mathbf{T}^{-1} = \begin{bmatrix} -1 & -1 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix},$$

$$\lambda_1 = 1 - x - y \qquad \lambda_2 = x - y$$
$$\lambda_3 = y - z \qquad \lambda_4 = y + z.$$

This enables us to write Eq. 10 as

$$f(\mathbf{r}) = \sum_{i=1}^{2} \lambda_i s_A(\mathbf{r_i}) + \sum_{i=3}^{4} \lambda_i s_B(\mathbf{r_i}). \tag{17}$$

Using the separable trilinear technique of Sigg and Hadwiger [20], evaluation of Eq. 17 can be derived by two linear fetches instead of four nearest neighbor ones. In general, two nearest neighbor fetches can be replaced by a linear fetch as:

$$\begin{aligned} (1-t)\,f_i + t\,f_{i+1} &\;\Rightarrow\; f(i+t) \\ a\,f_i + b\,f_{i+1} &\;\Rightarrow\; (a+b)\,f\left(i + \frac{b}{a+b}\right), \end{aligned} \tag{18}$$

as long as $t \in [0,1]$ and $\frac{b}{a+b} \in [0,1]$. By combining both λ_1 with λ_2 and λ_3 with λ_4, the linear box spline can be evaluated by two linear texture fetches:

$$\sum_{i=1}^{2} \lambda_i s_A(\mathbf{r_i}) \;\Rightarrow\; (1-2y)\,s_A\left(\mathbf{r_1} + \frac{x-y}{1-2y}\mathbf{\Pi}\begin{bmatrix}1\\0\\0\end{bmatrix}\right)$$
$$\sum_{i=3}^{4} \lambda_i s_B(\mathbf{r_i}) \;\Rightarrow\; 2y\,s_B\left(\mathbf{r_3} + \frac{y+z}{2y}\mathbf{\Pi}\begin{bmatrix}0\\0\\1\end{bmatrix}\right)$$

Summing these terms up, we get

$$
\begin{aligned}
f_A &= s_A\left(\mathbf{r_{base}} + \frac{x-y}{1-2y}\,\mathbf{s}\circ\mathbf{e}_{\pi(1)}\right) \qquad (19)\\
f_B &= s_B\left(\mathbf{r_{base}} + \mathbf{s}\circ\left(\begin{bmatrix}1/2\\1/2\\1/2\end{bmatrix} + \frac{z-y}{2y}\mathbf{e}_{\pi(3)}\right)\right),\\
f(\mathbf{r}) &= (1-2y)\,f_A + 2y\,f_B
\end{aligned}
$$

where $\circ$ represents the element-wise product. This is exactly what was claimed in Step 2 and Step 3 of the proposed evaluation scheme.

5 GPU IMPLEMENTATION

We employed the proposed trilinear evaluation scheme formulated in Eq. 19 in a GPU-based first-hit ray-casting application by using ray marching with equidistant steps. At each sample position, a filter kernel was used to reconstruct the volume from the discrete BCC samples. To get a numerically stable formulation when the resampling point lies within a triangular face, on an edge, or coincides a vertex, the divisions in Eq. 19 are evaluated as $\lim_{\varepsilon\to 0}\varepsilon\cdot s_i(\frac{\text{constant}}{\varepsilon}) = 0$. This numerical safeguard was incorporated in the GPU implementation as well.

In our GPU implementation, the lattice samples are stored as textures. Function $s_A(\mathbf{r})$ fetches the sample set s_A at $\mathbf{r}+[\frac{1}{2},\frac{1}{2},\frac{1}{2}]^T$, while function $s_B(\mathbf{r})$ fetches the shifted sample set s_B at $\mathbf{r}$. Sample sets s_A and s_B can be implemented as two separate textures or as one texture with two channels. We have not found an appreciable difference between these two methods. We present the complete Cg source of the proposed linear box spline resampling algorithm in the appendix.

We compare the rendering speed of our trilinearly evaluated linear box spline scheme to the latest method, that uses twice as many nearest neighbor fetches [13]. We also give a comparison to the major competitive methods: to the BCC B-spline [8] and to the BCC DC-spline [10]. Comprehensive analysis of the numerical accuracy, and visual quality of these splines are out of the scope of this paper. We refer the interested reader to [6, 10, 13] for a more thorough overview.

The number of texture lookups and the arithmetic costs differ for each filter (see Table 1). The arithmetic cost of the trilinear B-spline filtering is practically negligible [6, 8], the DC-spline filtering has moderate addressing overhead [10], while the trilinear and nearest neighbor linear box spline schemes have the highest number of floating point operations [13]. Concerning the number of texture fetches, the trilinear B-spline and the trilinearly evaluated linear box spline are in the best position: they need only two lookups, while the linear box spline filtering needs four fetches, and the DC-spline filtering needs six lookups.

Filter	lookups	complexity
Lin. box spline (nearest)	4	high
Lin. box spline (linear)	2	high
Trilinear B-spline	2	low
Linear DC-spline	6	medium

Table 1: Number of texture lookups and the arithmetic cost of different reconstruction filters. These properties determine the rendering performance.

The skeleton of the ray caster application was the same for each filtering technique, only the filter kernels and the storage scheme of the BCC samples were altered. For the nearest neighbor box spline evaluation, the BCC samples are stored in a one-channel texture by shifting the samples of the second lattice by half a grid spacing in every dimension [13]. For the trilinear box spline scheme, for the BCC B-splines, and for the BCC DC-splines, the BCC samples were stored as two separate set of CC samples as a two-channel texture.

5.1 Rendering Speed

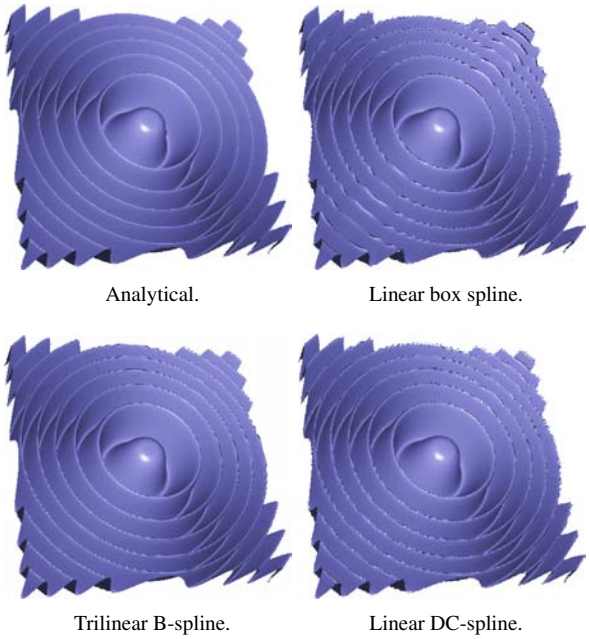
We rendered four data sets of different voxel counts at an image resolution of 512×512. The analytically defined Marschner-Lobb test signal was sampled at $64^3\times 2$ BCC resolution. The other three data sets are well-known CT scans reconstructed originally on a CC lattice. To get a BCC representation of them, we employed a frequency-domain upsampling [7].

The viewing rays were evaluated in front-to-back order, which enabled us to use early ray termination. The first-hit isosurfaces were shaded by the Blinn-Phong model using gradients calculated from central differences. The ray marching step and the central differencing step were adjusted to the voxel size of the data sets.

The renderings of the Marschner-Lobb test signal are illustrated in Figure 3. Note that the linear box spline introduces postaliasing artifacts along the diagonal directions, while the artifacts produced by the linear DC-spline or the trilinear B-spline are less apparent.

To get relevant rendering speeds, we chose the middle-aged NVIDIA Geforce 8700 GPU for our experiments. The observed frame rates are illustrated in Table 2. According to our prior expectations, the frame rates depended on the number of samples fetched, the algorithmic complexity of the filter kernel, the resolution of the volume, and the distance of the iso-surface from the image plane.

We can confirm the observation, that the frame rates get similar as the number of voxels increases with appropriately decreasing the sampling distance. Possibly, the texture fetches become the bottleneck of the rendering pipeline. This can be the reason why the DC-spline results in the lowest frame rates for the highest voxel counts.



Analytical. Linear box spline.

Trilinear B-spline. Linear DC-spline.

Figure 3: Renderings of the analytical Marschner-Lobb test signal and its sampled representations at $64^3 \times 2$ reconstructed by different resampling filters.

Data set	$M_{\Xi^1_{BCC},nearest}$	$M_{\Xi^1_{BCC},linear}$	β^1	χ^1_{BCC}
ML	21.03	22.90	53.64	21.75
Engine	16.28	17.18	41.82	16.42
Carp	9.22	10.00	25.07	9.19
Xmas Tree	5.77	6.19	6.57	4.61

Table 2: Frame rates in frames per second for different reconstruction filters and popular data sets: the Marschner-Lobb test signal sampled at $64^3 \times 2$, the "Engine Block" at $256^2 \times 110 \times 2$, the "Carp" at $256^2 \times 512 \times 2$, and the "Christmas Tree" at $512 \times 499 \times 512 \times 2$.

On the other hand, for low and moderate volume resolutions, the arithmetic complexity seems to be more important than the number of texture fetches. It is interesting to note that the concept of applying linear fetches instead of nearest neighbor ones [20] does not always pay off. We think that the texture cache operates very well for filters with a narrow support. This might explain that the nearest neighbor version and the linear version of the linear box spline filtering as well as the linear DC-spline filtering with even six samples attain similar frame rates, while the trilinear B-spline holds a towering lead in performance.

6 CONCLUSION AND FUTURE WORK

In this paper, we have proposed a GPU evaluation scheme for the linear BCC box spline filtering exploiting the hardwired trilinear texture fetching. This result enabled us to make a fair comparison of the linear box spline, the BCC B-spline, and the BCC DC-spline in terms of their performance. We found that, in general, the proposed linear evaluation scheme operates slightly faster than the evaluation scheme with nearest neighbor fetches [13]. However, using an optimized GPU implementation, the trilinear B-spline can still achieve the best performance, as it takes the minimum number of samples with the lowest arithmetic cost. Since the texture fetches become more expensive when the support of the filter gets wider or the resolution of the volume increases, we plan to develop a similar scheme for the quintic box spline for the BCC lattice.

ACKNOWLEDGMENTS

This project was supported by Mediso Medical Imaging Systems, the Hungarian National Office for Research and Technology (Project ID: TECH 08/A2), and the New Hungary Development Plan (Project ID: TÁMOP-4.2.1/B-09/1/KMR-2010-0002). This work is connected to the scientific program of the "Development of quality-oriented and harmonized R+D+I strategy and functional model at BME" project.

REFERENCES

[1] T. Blu, P. Thévenaz, and M. Unser. Generalized interpolation: Higher quality at no additional cost. In *Proceedings of IEEE International Conference on Image Processing*, pages 667–671, 1999.

[2] Dutta Roy S. C. and Kumar B. *Handbook of Statistics*, volume 10, chapter Digital Differentiators, pages 159–205. Elsevier Science Publishers B. V., N. Holland, 1993.

[3] I. Carlbom. Optimal filter design for volume reconstruction and visualization. In *Proceedings of the 4th conference on Visualization*, pages 54–61. IEEE Computer Society, 1993.

[4] B. Csébfalvi. Prefiltered gaussian reconstruction for high-quality rendering of volumetric data sampled on a body-centered cubic grid. In *Proceedings of IEEE Visualization*, pages 311–318, 2005.

[5] B. Csébfalvi. BCC-splines: Generalization of B-splines for the body-centered cubic lattice. *Journal of WSCG 16, 1–3 (2008)*, pages 81–88, 2008.

[6] B. Csébfalvi. An evaluation of prefiltered B-spline reconstruction for quasi-interpolation on the body-centered cubic lattice. *IEEE Transactions on Visualization and Computer Graphics 16, 3 (2010)*, pages 499–512, 2010.

[7] B. Csébfalvi and B. Domonkos. Frequency-domain upsampling on a body-centered cubic lattice for efficient and high-quality volume rendering. In *Proceedings of Vision, Modeling, and Visualization*, pages 225–232, 2009.

[8] B. Csébfalvi and M. Hadwiger. Prefiltered B-spline reconstruction for hardware-accelerated rendering of optimally sampled volumetric data. In *Proceedings of VMV*, pages 325–332, 2006.

[9] C. de Boor, K. Höllig, and S. Riemenschneider. *Box Splines*, volume 98. Springer-Verlag, 1993.

[10] B. Domonkos and B. Csébfalvi. DC-splines: Revisiting the trilinear interpolation on the body-centered cubic lattice. In *Proceedings of VMV*, pages 275–282, 2010.

[11] A. Entezari. *Optimal Sampling Lattices and Trivariate Box Splines*. PhD thesis, Simon Fraser University, Vancouver, Canada, July 2007.

[12] A. Entezari, D. Van De Ville, and T. Möller. Practical box splines for reconstruction on the body centered cubic lattice. *IEEE Trans. on Vis. and Computer Graphics*, 14(2):313–328, 2008.

[13] B. Finkbeiner, A. Entezari, D. Van De Ville, and T. Möller. Efficient volume rendering on the body centered cubic lattice using box splines. *Computers and Graphics*, 34(4):409–423, 2010.

[14] L. Kobbelt. Stable evaluation of box splines. *Numerical Algorithms*, 14, 1996.

[15] O. Mattausch. Practical reconstruction schemes and hardware-accelerated direct volume rendering on bodycentered cubic grids. Master's thesis, Vienna University of Technology, 2003.

[16] M. Mirzargar and A. Entezari. Voronoi splines. *IEEE Transactions on Signal Processing*, 58:4572–4582, Sept 2010.

[17] T. Möller, R. Machiraju, K. Mueller, and R. Yagel. A comparison of normal estimation schemes. In *Proceedings of the 8th conference on Visualization*, pages 19–ff., Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[18] T. Möller, K. Mueller, Y. Kurzion, R. Machiraju, and R. Yagel. Design of accurate and smooth filters for function and derivative reconstruction. In *Proceedings of the IEEE symposium on Volume visualization*, pages 143–151, New York, NY, USA, 1998. ACM.

[19] D. P. Petersen and D. Middleton. Sampling and reconstruction of wave-number-limited functions in n-dimensional euclidean spaces. *Information and Control*, 5(4):279–323, 1962.

[20] C. Sigg and M. Hadwiger. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, pages 313–329. Addison- Wesley, 2005.

[21] T. Theußl, T. Möller, and M. E. Gröller. Optimal regular volume sampling. In *Proceedings of the conference on Visualization*, pages 91–98. IEEE Computer Society, 2001.

[22] T. Theußl, T. Möller, and M. E. Gröller. Reconstruction schemes for high quality raycasting of the body-centered cubic grid. Technical Report TR-186-2-02-11, Institute of Computer Graphics and Algorithms, TU Vienna, Dec 2002.

[23] D. Van De Ville, T. Blu, M. Unser, W. Philips, I. Lemahieu, and R. Van de Walle. Hex-Splines: A novel spline family for hexagonal lattices. *IEEE Transactions on Image Processing*, 13(6):758–772, 2004.

CG SHADER CODE

```
uniform float3    Size;
uniform sampler3D Volume;

// Handle removable singularity
#define DIV( A, B ) \
  ( abs(B) ? ( A ) / ( B ) : 0.0 )

// I^th coordinate of unit vector e_π(1)
#define E_PI_1( I ) ( a.I == x )

// I^th coordinate of unit vector e_π(3)
#define E_PI_3( I ) \
  ( a.I == z && a.I != x && a.I != y )

// Unit vector e_π(J)
#define E_PI( J ) float3( E_PI_ ## J( x ), \
  E_PI_ ## J( y ), E_PI_ ## J( z ) )

// Fetching a trilinear sample from Λ_CC_A at R
#define S_A( R ) \
  tex3D(Volume, (r_base + (R) + 0.5) / Size).r

// Fetching a trilinear sample from Λ_CC_B at R
#define S_B( R ) \
  tex3D( Volume, ( r_base + (R) ) / Size ).a

float linearBoxSpline( float3 texCoords ) {
  // Resampling point r
  float3 r = texCoords * Size - 0.5;

  // Nearest lattice point of Λ_CC_A
  float3 r_base = round( r );

  // Relative coordinates d,
  // their absolute values a, and signs s
  float3 d = r - r_base;
  float3 a = abs( d );
  float3 s = sign( d );

  // Sorting a by its components
  float x = max( a.x, max( a.y, a.z ) );
  float z = min( a.x, min( a.y, a.z ) );
  float y = a.x + a.y + a.z - x - z;

  // Fetching from sample sets Λ_CC_A and Λ_CC_B
  float two_y = 2.0 * y;
  float tA = DIV( x - y, 1.0 - two_y );
  float tB = DIV( z - y, two_y );
  float fA = S_A( tA * s * E_PI(1) );
  float fB = S_B(s * ( 0.5 + tB * E_PI(3) ));

  // Linear interpolation of the two samples
  return lerp( fA, fB, two_y );
}
```