I N F O R M A T I K

Faster Ray Tracing with SIMD Shaft Culling

Kirill Dmitriev Vlastimil Havran
Hans-Peter Seidel

MPI–I–2004–4–006 December 2004

FORSCHUNGSBERICHT RESEARCH REPORT

M A X - P L A N C K - I N S T I T U T
F Ü R
I N F O R M A T I K

Stuhlsatzenhausweg 85 66123 Saarbrücken Germany

Author's Address

Kirill Dmitriev Vlastimil Havran Hans-Peter Seidel
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany
{kirill, havran, hpseidel}@mpi-sb.mpg.de

Abstract

This paper presents a generic technique for acceleration of ray tracing of polygonal scenes. We propose scheduling rays in a way that forms pyramidal shafts. We show that under certain conditions fulfilled by the corner rays of a shaft, it is possible to immediately and conservatively answer visibility queries for the inner rays without expensive ray traversal through acceleration data structures (kd-tree in our case). We show that the presented technique is suitable for primary, secondary, and shadow rays.

Keywords

Ray Tracing, SIMD, Shaft Culling, Longest Common Traversal Sequence.

1 Introduction

Visibility computation via ray tracing is the most time consuming part of many rendering algorithms. Thus algorithms that improve ray tracing speed are very important for research and industry and have immediate use in a wide spectrum of applications.

In this paper we propose a new algorithm for speeding up the visibility computation. It is a combination of several existing techniques and it takes advantage of their best features. We aim at the following properties:

- algorithm should compute exact visibility information
- no additional preprocessing should be needed
- no additional user-definable parameters should be introduced
- technique should be easy to implement
- acceleration should be achieved for most of scenes, no significant deceleration should happen for others

The founding observation that is widely utilized in existing acceleration techniques is that rays frequently appear to have the same origin and very close directions. This holds for rays fired from the pinhole camera, shadow rays (the origin is located at the point light source position then), primary rays that are reflected or refracted at planar media boundary (the origin is located at the reflected camera observer in this case). This kind of coherence allows bucketing rays into coherent groups and applying different kinds of optimization techniques for intersecting the whole group rather than separate rays with the scene objects.

In the next section we review previous works in the area of ray tracing acceleration. In the section 3 we give details of our algorithm. Section 4 contains results and discussion that are concluded in section 5.

2 Previous Work

Since ray tracing was introduced in [16], tremendous number of techniques for improving its efficiency was proposed. This includes development of ray tracing data structures (e.g. uniform grids[4] and kd-trees[10, 12]), use of interpolation [2, 15], parallelization [11] and so on. Complete classification and review of such techniques can be found in [3, 9]. Here we focus only on techniques that are closely related to our algorithm.

2.1 Shadow Caching

In 1986 Haines and Greenberg suggested a shadow cache algorithm [5]. Each light source caches a reference to the opaque object that has most recently cast a shadow from this light source, or *null* reference if the last shadow ray fired to this light source did not encounter any opaque objects. Before shadow ray is traced towards the light source, a cached object is checked for an intersection with this ray. If an intersection occurs, no further processing is required.

The disadvantage of this technique is that if reference in light source is *null* or no intersection with cached object occurs, ray has to be traversed all the way. Our algorithm also caches objects, most recently intersected by shadow rays, but accelerates visibility computations even in 'not shadowed' case.

2.2 Shaft Culling

Haines and Wallace[6, 7] introduced shaft culling in the context of radiosity form-factors computation. The technique intersects a scene enclosed into hierarchy of axis-aligned bounding volumes against a convex shaft with planar boundaries. Testing a box against a shaft is a fast operation, as only one corner of the box needs to be compared to each particular plane in the shaft. All rays inside the shaft are then checked for intersection with objects that are not culled away. Zwaan et al. [17] uses the shaft culling method for ray tracing. Although they improve the speed of shaft culling construction, no ray tracing speedup is achieved. Thus authors suggest to use their technique to increase the coherence of computations for parallel ray tracing.

Similarly, our method encloses groups of rays into pyramidal convex shafts. However, instead of bounding volumes hierarchy we use far more efficient kd-tree structure ([9], chapter 3), and instead of testing each plane of the shaft our algorithm executes conventional traversal loop for its boundary rays.

2.3 Termination Object

LCTS (Longest Common Traversal Sequence) technique, proposed by Havran [8], groups rays in such a way, that they create convex shafts. Due to convexity, if common sequence of leaves visited by the boundary rays exists, exactly this sequence has to be visited by the inner rays as well. Thus a sequence of empty leaves can be skipped by the inner rays and they can immediately start from the first non-empty leaf encountered by the boundary rays.

An important special case of LCTS is given by the termination object criterion ([9], section 6.3.3.2). If all boundary rays traverse the same sequence of empty leaves and terminate at the same convex object that is the only object contained in the last visited leaf, then all inner shaft rays have to terminate at this object as well. In this case the visibility queries for the inner rays can be immediately answered without ray tracing.

Disadvantage of the technique is that it traces the rays bounding the shaft one by one, which requires storing the sequence of kd-tree leaves, traversed by each ray, and then comparing those sequences. We avoid any storage by tracing the 4 boundary rays simultaneously and checking their coherence as they propagate through the kd-tree. Only one additional bitwise operation in the kd-tree traversal loop is needed, introducing virtually no overhead when compared to conventional ray shooting. We also allow non-empty leaves on the ray paths and more than one triangle in the terminating leaf. This is very important for ray tracing of triangulated scenes, since kd-tree leaves rarely contain a single triangle.

2.4 SIMD Ray Tracing

SIMD (Single Instruction Multiple Data) technique suggested by Wald et al. [13] in 2001 traces rays in groups of 4 and uses SIMD instructions to perform kd-tree traversal and ray/triangle intersections for 4 rays simultaneously. The maximum efficiency is obtained if all of them actually go through the same kd-tree nodes, i.e. are coherent. In this case the cost of traversal loops and ray/triangle intersections is amortized over all 4 rays. The technique also reduces memory bandwidth by requesting data only once per ray packet, and increases the processor cache utilization at the same time. Algorithms that use SIMD ray tracing must be specifically designed to shoot the rays by coherent packs. It has been shown that both recursive ray tracing [13] and global illumination algorithms [14] can be designed in appropriate way.

Wald et al. [13] report speed improvement of an order of magnitude compared to other well-known ray tracers, which makes the method a proper starting point for further optimizations. We use SIMD instructions as well, but instead of tracing rays in packets of 4, we create pyramidal shafts consisting of at least 16 rays, 4 rays being at the corners and others - inside the shaft. When traversing the corner rays we collect all the triangles that potentially can be intersected by the inner rays in the memory buffer. Shooting of inner rays then usually consists only of intersecting them with such *relevant* triangles, which are very few in number (typically 1-2 per ray packet).

3 Relevant Triangles Tracing (RTT)

To create convex ray shafts, we compute images in 4×4 independent pixel tiles. Such image tiling requires that the image horizontal and vertical resolutions are multiples of 4. It is already so for most common resolutions (640x480, 1024x768, 1280x768, etc.). If not, we render slightly larger image and then crop it to match the initial resolution. Depending on the antialiasing setting, each 4×4 pixel tile can be subdivided to a number of subpixels (Figure 1).

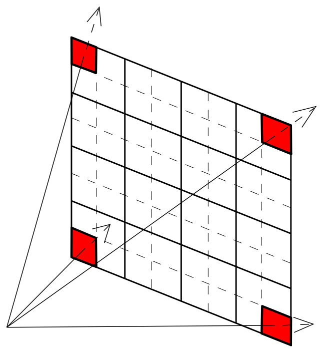


Figure 1: 4×4 pixels tile (pixel borders are shown with bold lines) with 2×2 antialiasing. Rays that go through the tile corners create a convex pyramidal shaft.

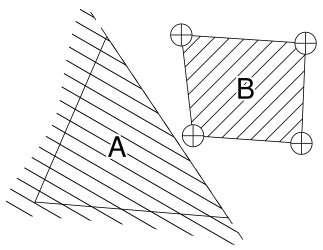


Figure 2: A is a zone where one of the barycentric coordinates is positive. If this barycentric coordinate is negative for all of the 4 corner rays, none of the inner shaft rays (zone B) can intersect the triangle.

4 rays that go through the corner subpixels of each tile are simultaneously traced using SIMD instructions. During traversal we check that rays go all the way through the same kd-tree nodes. This is an easy thing to track since traversal algorithm has this information anyway and all we have to do is to set some flag as soon as the rays split to different sub-trees. If this flag

happens to be set in the end, shaft is considered to be *not coherent* with respect to the kd-tree and RTT can not be applied.

Every time the corner rays encounter a nonempty kd-tree node, triangles contained there have to be tested for an intersection. Each triangle is simultaneously tested against 4 rays using SIMD instructions. 5 conventional tests are applied: 2 tests check that the triangle plane is located between the ray origin and the ray end, 3 tests check that each barycentric coordinate of plane/ray intersection is positive. Due to the shaft convexity, if any of those tests fails for all 4 rays simultaneously, the triangle can not be intersected by any of the inner shaft rays (see example in Figure 2).

If none of the intersection tests fails for all 4 corner rays simultaneously, the triangle has a chance to be intersected by the inner rays and thus its index is stored into a so called *relevant triangles list.* Under certain conditions visibility queries for the inner shaft rays can be answered without expensive kd-tree traversal, but instead by checking just triangles from the relevant list. As stated above, similarly to the termination criterion idea [9], the first of those conditions is coherence of the shaft corners. The second condition depends on the query type and is discussed below.

3.1 Primary and Secondary Rays

For RTT to be applicable to primary and secondary ray shafts, the corner rays of the shaft have to terminate at the same triangle. Otherwise (Figure 3), testing accumulated relevant triangles for an intersection is not sufficient for conservative answering visibility queries. The case when corner rays hit two triangle that share an edge, could still be handled by supporting connectivity information, but this solution is more complicated to implement.

The approach is more difficult to apply for secondary rays because distortion caused by one or more reflections can destroy the convexity of the initial shaft of primary rays. Testing a shaft for convexity after such a transformation can be computationally expensive in general case. We therefore suggest to keep searching for terminating triangles only while a shaft is reflected by *planar* objects because in such a case the convexity is clearly preserved.

3.2 Shadow Rays

We create shafts for shadow rays by connecting the light source position with the hit points of corresponding primary (or secondary) rays as depicted in Figure 4. Clearly shaft B will be convex if the originating shaft A was convex and the rays of shaft A have terminated on coplanar triangles. Thus for shadow rays the condition for RTT to be applicable is more relaxed than

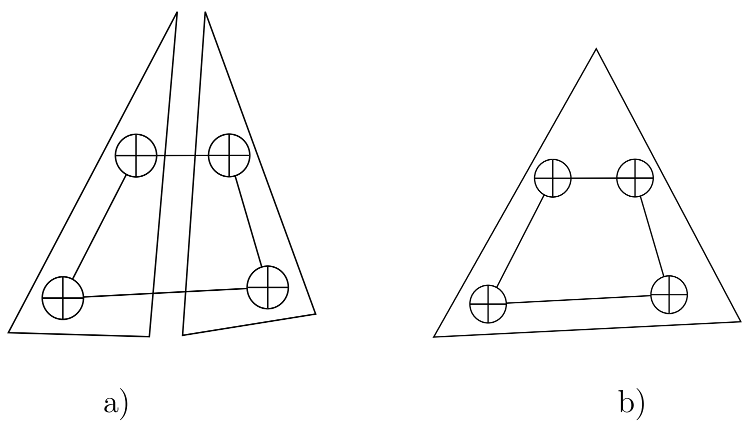


Figure 3: RTT for primary and secondary rays. a) Corner shaft rays hit different triangles of the same kd-tree leaf. It is not guaranteed, that the inner rays will pierce any of the depicted triangles, and even that they will terminate in the same leaf: they can slip through the gap between triangles. b) The inner shaft rays have to end at depicted triangle because all of the corner rays are ending there.

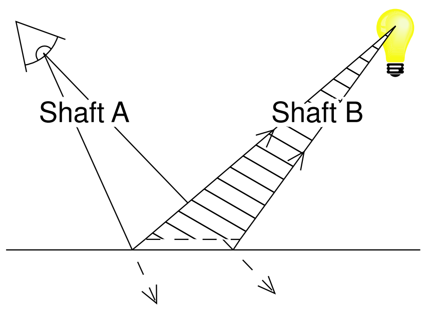


Figure 4: A shaft created by shadow rays originates at the light source position and ends some ϵ before reaching the hit points of primary (secondary) rays.

for primary rays. It is only required that the primary rays of originating shaft land at coplanar triangles, not necessarily at the same triangle.

4 Results and Discussion

The algorithm was implemented and tested within the 3DS MAX framework that offers vast modeling, animating, and rendering benefits. We only had to implement ray shooting. Other procedures needed for rendering, like shading of materials and evaluation of light sources emission, are provided by the 3DS MAX core. Timings were measured purely for ray shooting and do not include shading that strongly depends on the specifics of the underlying system. Timer based on RDTSC (Read Time-Stamp Counter) processor instruction was used. The resolution of such timer is equal to the processor frequency (in our case $2,385 \cdot 10^6$ ticks per second) and is enough for measuring relative performance of even single ray shooting queries.

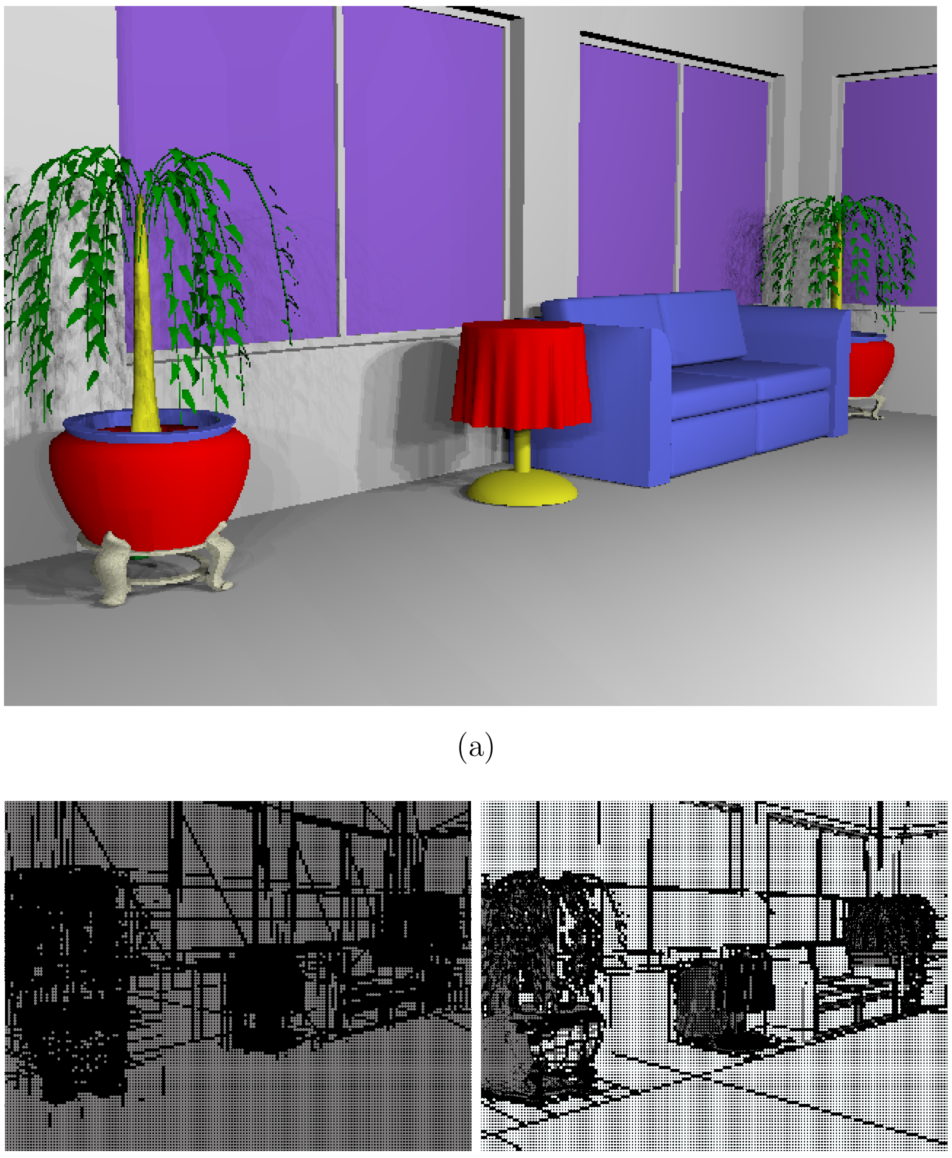
Figure 5 a) demonstrates one of the testing scenes, which is an office environment illuminated by 16 omni-directional light sources. Figure 5 b) illustrates RTT for primary rays. Pixels are black where kd-tree traversal was used, they are white - where visibility queries that were answered by RTT without ray traversal. Lines of black pixels appear at the borders between two triangles or between two kd-tree nodes. Concentration of black color corresponds to the image regions with detailed geometry, where the projected triangle size is comparable to the size of the 4x4 pixels tile used for shafts construction. Increasing the image resolution changes the notion of 'detailed' and makes the algorithm perform better, which will be demonstrated later on. Figure 5 c) shows similar statistics for shadow rays from one of the light sources, except that we introduce two gradations for queries answered by RTT. White corresponds to pixels where relevant triangles list contained 0 triangles, gray - to pixels where it contained 1 or more triangles.

The other two test scenes are depicted in Figure 6. Figure 6 a) is a room environment illuminated by 7 spot light sources. Figure 6 b) is an office environment filled with plants. Many small leaves destroy image space coherence and make this scene a hard case for the suggested algorithm.

We compare RTT with the conventional SIMD ray tracing technique [13], which is currently one of the fastest ray tracing algorithms and was used as a starting point for our technique.

Table 1 lists miscellaneous statistics for the test scenes computed at the resolution 640x480. It is seen that even at this small resolution, technique allows to save significant amount of primary and shadow rays. For the ROOM scene RTT needs about 1.8 times less primary rays and about 1.9 times less shadow rays compared to SIMD ray tracing. The time reduction, however, is not that spectacular: 1.22 and 1.17 for primary and shadow rays tracing respectively. Reason for that is that the rays saved by RTT usually traverse empty or poorly populated scene regions. kd-tree in such a regions is quite shallow and it does not cost much to traverse it. Even though almost halve of all rays are skipped, it is actually the other halve that takes most of the time.

To exploit the power of the technique better, two possibilities exist. The first is increasing the ratio of inner shaft rays, thus increasing acceleration


(a)

(b) (c)

Figure 5: Visualization of RTT for OFFICE test scene. (a) OFFICE test scene. (b) Visualization of RTT for primary rays. (c) Visualization of RTT for shadow rays.

potential. Without antialiasing each 4x4 tile contains just 16 rays resulting in ratio of corner rays equal to 25%. When using antialiasing 2x2, each 4x4 tile requires 64 rays and the ratio of corner rays is reduced to 6.25%. Graph in Figure 7 shows the dependency of relative acceleration from antialiasing settings. Note that even for the hard case of PLANTS scene technique shows

	ROOM	OFFICE	PLANTS
SIMD pr. rays time	119	157	477
#primary rays	307200	307200	307200
RTT pr. rays time	96.8	142	502
#primary rays	172524	170928	273588
#avr. relevant trgs	1.134	1.139	1.127
SIMD sh. rays time	81.8	2127	5991
#shadow rays	247251	4812053	3708088
RTT sh. rays time	70.1	1716	5485
#shadow rays	129461	2356197	2400053
#avr. relevant trgs	1.095	3.246	4.274

Table 1: Performance statistics for the test scenes. Time for primary and shadow rays shooting separately is given in milliseconds. For RTT we additionally provide an average number of entries in the relevant triangles list.

some (though not large) acceleration.

Second possibility to exploit the suggested technique in a better way is computing images of larger resolution. So far we have only experimented with image resolution of 640x480 that does not supply much image space coherency. Acceleration dependency on the image resolution is shown on the Figure 8.

Increasing both resolution and antialiasing simultaneously can yield even more benefit. For instance computing image of ROOM scene with antialiasing 6x6 at the resolution 2048x1560 is 2.43 times faster using RTT than SIMD algorithm.

5 Conclusion

We have suggested a robust and simple ray tracing acceleration algorithm. The algorithm is based on SIMD ray tracing technique, but further utilizes image space ray tracing coherence to achieve even greater performance. The neighboring rays are grouped into coherent packs, bounded by convex shafts. When executing ray traversal loop for the shaft corners, the algorithm collects information that allows to immediately and conservatively answer visibility queries for the inner shaft rays without expensive kd-tree traversal. The information collection is performed on-the-fly and does not require any additional scene preprocessing.

The proposed technique achieves acceleration on most of scenes. When computing small resolution images without antialiasing, speedup of 1.1 to 1.3 times can be expected. Especially large performance benefits up to 2.5 times are received when computing high quality images of large resolution.

References

[1] A. Appel. Some Techniques for Shading Machine Renderings of Solids. *Spring Joint Computer Conference Proceedings*, pages 37–45, 1968.

[2] K. Bala, J. Dorsey, S. Teller. Radiance Interpolants for Accelerated Bounded-Error Ray Tracing. *ACM Transactions on Graphics*, Vol.18 No.3, pages 213–256, 1999.

[3] A. Chang. A survey of Geometric Data Structures for Ray Tracing. *Technical Report TR-CIS-2001-06*, Polytechnic University, Brooklyn, 2001.

[4] A. Fujimoto and K. Iwata. Accelerated ray tracing. *Computer Graphics: Visual Technology and Art (Computer Graphics Tokyo Proceedings)*, pages 41–65, Tokyo, 1985.

[5] E. Haines and D. Greenberg. The Light Buffer: a Shadow-Testing Accelerator. *IEEE Computer Graphics and Applications*, Vol.6 No.9, pages 6–16, 1986.

[6] E. Haines and J. Wallace. Shaft Culling for Efficient Ray-Cast Radiosity. *2nd Eurographics Workshop on Rendering Proceedings*, pages 122–138, 1991.

[7] E. Haines. Shaft Culling Tool. *Journal of Graphics Tools*, pages 23-26, 2000.

[8] V. Havran and J. Bittner. LCTS: Ray Shooting using Longest Common Traversal Sequences. *Eurographics Conference Proceedings*, pages 59–70, Interlaken, 2000.

[9] V. Havran. Heuristic Ray Shooting Algorithms. *Ph.D. Thesis*, Czech Technical University, 2000.

[10] M. Kaplan. Space-Tracing: A Constant Time Ray-Tracer. *Siggraph State of the Art in Image Synthesis Seminar Notes*, 1985.

[11] S. Parker, W. Martin, P.-P. Sloan, P. Shirley, B. Smits and C. Hansen. Interactive Ray Tracing. *ACM Symposium on Interactive 3D Graphics*, pages 119–126, 1999.

[12] K. Sung and P. Shirley. Ray Tracing with the BSP Tree. *Graphics Gems III*, pages 271–274, San Diego, 1992.

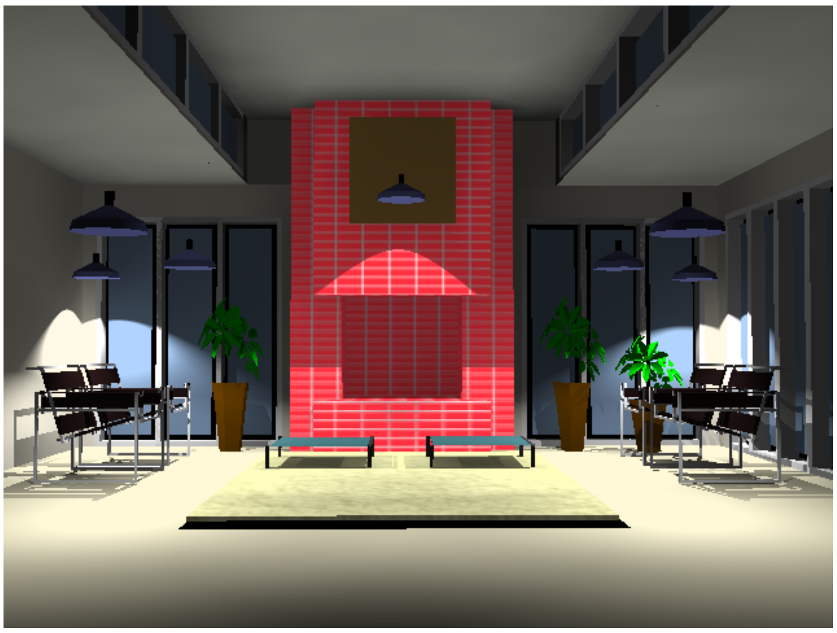
[13] I. Wald, C. Benthin, M. Wagner and P. Slusallek. Interactive Rendering with Coherent Ray Tracing. *Eurographics Conference Proceedings*, 2001.

[14] I. Wald, T. Kollig, C. Benthin, A. Keller and P. Slusallek. Interactive Global Illumination. *13th Eurographics Workshop on Rendering Proceedings*, 2002.

[15] G. Ward and P. Heckbert. Irradiance gradients. *3rd Eurographics Workshop on Rendering Proceedings*, 1992.

[16] T. Whitted. An Improved Illumination Model for Shaded Display. *Siggraph Conference Proceedings*, pages 1–14, 1979.

[17] M. van der Zwaan, E. Reinhard, F. Jansen Pyramid Clipping for Efficient Ray Traversal. *6th Eurographics Workshop on Rendering Proceedings*, pages 1–10, 1992.


(a)

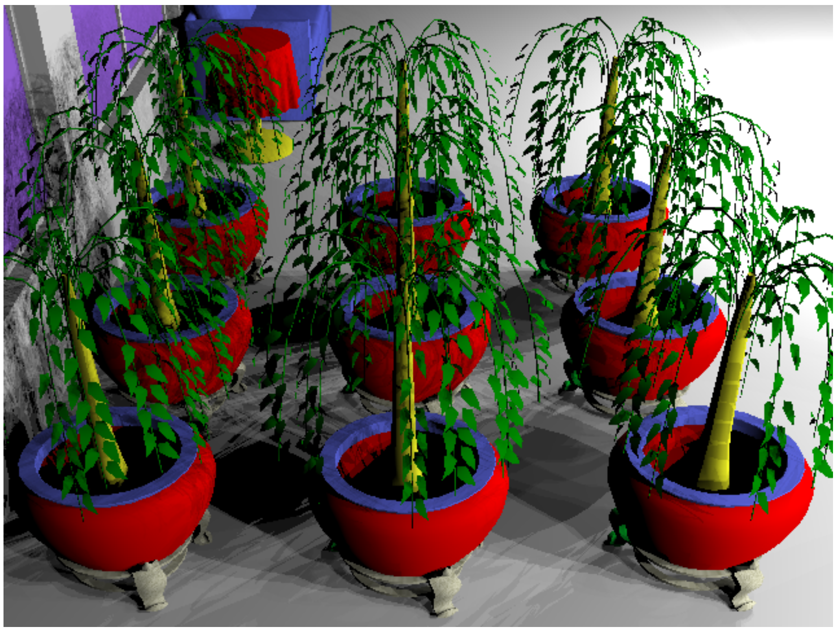

(b)

Figure 6: More test scenes. (a) ROOM test scene. (b) PLANTS test scene.

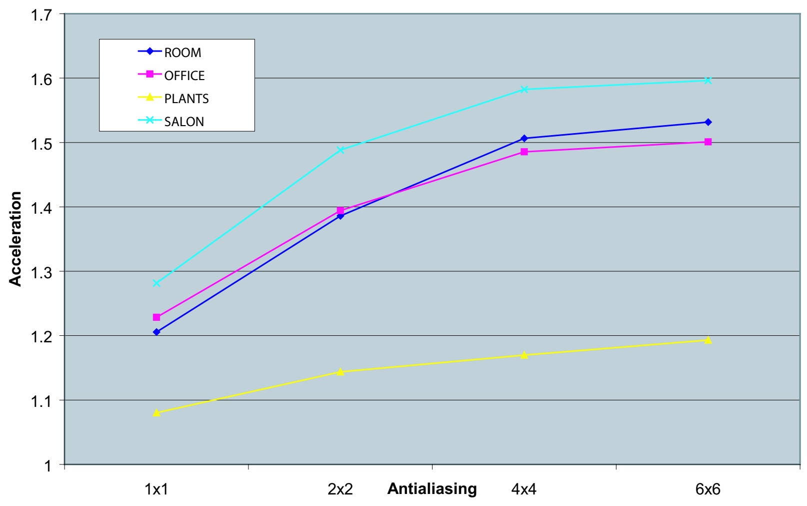


Figure 7: Dependency of acceleration on antialiasing.

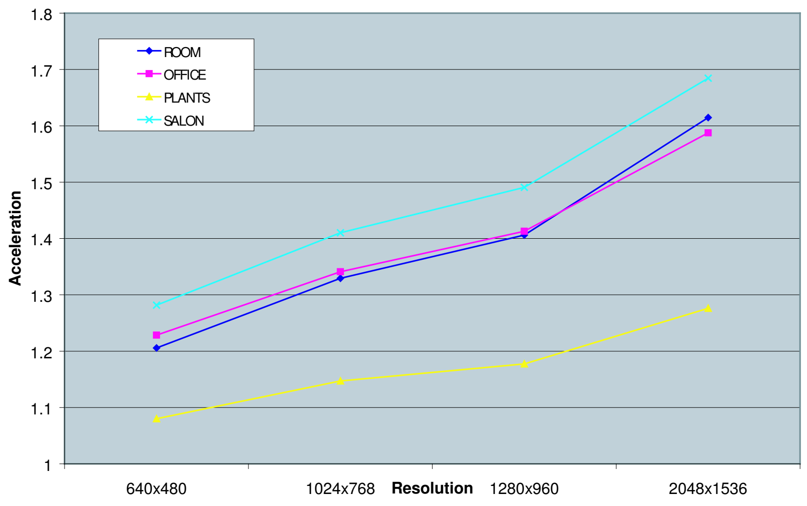


Figure 8: Dependency of acceleration on resolution.

MPI-I-2005-4-001	M. Fuchs, V. Blanz, H. Lensch, H. Seidel	Reflectance from Images: A Model-Based Approach for Human Faces
MPI-I-2004-NWG3-001	M. Magnor	Axisymmetric Reconstruction and 3D Visualization of Bipolar Planetary Nebulae
MPI-I-2004-NWG1-001	B. Blanchet	Automatic Proof of Strong Secrecy for Security Protocols
MPI-I-2004-5-001	S. Siersdorfer, S. Sizov, G. Weikum	Goal-oriented Methods and Meta Methods for Document Classification and their Parameter Tuning
MPI-I-2004-4-006	K. Dmitriev, V. Havran, H. Seidel	Faster Ray Tracing with SIMD Shaft Culling
MPI-I-2004-4-005	I.P. Ivrissimtzis, W.-. Jeong, S. Lee, Y.a. Lee, H.-. Seidel	Neural Meshes: Surface Reconstruction with a Learning Algorithm
MPI-I-2004-4-004	R. Zayer, C. Rssl, H. Seidel	r-Adaptive Parameterization of Surfaces
MPI-I-2004-4-003	Y. Ohtake, A. Belyaev, H. Seidel	3D Scattered Data Interpolation and Approximation with Multilevel Compactly Supported RBFs
MPI-I-2004-4-002	Y. Ohtake, A. Belyaev, H. Seidel	Quadric-Based Mesh Reconstruction from Scattered Data
MPI-I-2004-4-001	J. Haber, C. Schmitt, M. Koster, H. Seidel	Modeling Hair using a Wisp Hair Model
MPI-I-2004-2-002	P. Maier	Intuitionistic LTL and a New Characterization of Safety and Liveness
MPI-I-2004-2-001	H.d. Nivelle, Y. Kazakov	Resolution Decision Procedures for the Guarded Fragment with Transitive Guards
MPI-I-2004-1-007	S. Wagner	?
MPI-I-2004-1-006	L.S. Chandran, N. Sivadasan	On the Hadwiger's Conjecture for Graph Products
MPI-I-2004-1-005	S. Schmitt, L. Fousse	A comparison of polynomial evaluation schemes
MPI-I-2004-1-004	N. Sivadasan, P. Sanders, M. Skutella	Online Scheduling with Bounded Migration
MPI-I-2004-1-003	I. Katriel	On Algorithms for Online Topological Ordering and Sorting
MPI-I-2004-1-002	P. Sanders, S. Pettie	A Simpler Linear Time 2/3 - epsilon Approximation for Maximum Weight Matching
MPI-I-2004-1-001	N. Beldiceanu, I. Katriel, S. Thiel	Filtering algorithms for the Same and UsedBy constraints
MPI-I-2003-NWG2-002	F. Eisenbrand	Fast integer programming in fixed dimension
MPI-I-2003-NWG2-001	L.S. Chandran, C.R. Subramanian	Girth and Treewidth

MPI-I-2003-4-009	N. Zakaria	FaceSketch: An Interface for Sketching and Coloring Cartoon Faces
MPI-I-2003-4-008	C. Roessl, I. Ivrissimtzis, H. Seidel	Tree-based triangle mesh connectivity encoding
MPI-I-2003-4-007	I. Ivrissimtzis, W. Jeong, H. Seidel	Neural Meshes: Statistical Learning Methods in Surface Reconstruction
MPI-I-2003-4-006	C. Roessl, F. Zeilfelder, G. Nrnberger, H. Seidel	Visualization of Volume Data with Quadratic Super Splines
MPI-I-2003-4-005	T. Hangelbroek, G. Nrnberger, C. Roessl, H.S. Seidel, F. Zeilfelder	The Dimension of C^1 Splines of Arbitrary Degree on a Tetrahedral Partition
MPI-I-2003-4-004	P. Bekaert, P. Slusallek, R. Cools, V. Havran, H. Seidel	A custom designed density estimation method for light transport
MPI-I-2003-4-003	R. Zayer, C. Roessl, H. Seidel	Convex Boundary Angle Based Flattening
MPI-I-2003-4-002	C. Theobalt, M. Li, M. Magnor, H. Seidel	A Flexible and Versatile Studio for Synchronized Multi-view Video Recording
MPI-I-2003-4-001	M. Tarini, H.P.A. Lensch, M. Goesele, H. Seidel	3D Acquisition of Mirroring Objects
MPI-I-2003-2-004	A. Podelski, A. Rybalchenko	Software Model Checking of Liveness Properties via Transition Invariants
MPI-I-2003-2-003	Y. Kazakov, H. Nivelle	Subsumption of concepts in $DL\ \mathcal{FL}_0$ for (cyclic) terminologies with respect to descriptive semantics is PSPACE-complete
MPI-I-2003-2-002	M. Jaeger	A Representation Theorem and Applications to Measure Selection and Noninformative Priors
MPI-I-2003-2-001	P. Maier	Compositional Circular Assume-Guarantee Rules Cannot Be Sound And Complete
MPI-I-2003-1-018	G. Schaefer	A Note on the Smoothed Complexity of the Single-Source Shortest Path Problem
MPI-I-2003-1-017	G. Schfer, S. Leonardi	Cross-Monotonic Cost Sharing Methods for Connected Facility Location Games
MPI-I-2003-1-016	G. Schfer, N. Sivadasan	Topology Matters: Smoothed Competitive Analysis of Metrical Task Systems
MPI-I-2003-1-015	A. Kovcs	Sum-Multicoloring on Paths
MPI-I-2003-1-014	G. Schfer, L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, T. Vredeveld	Average Case and Smoothed Competitive Analysis of the Multi-Level Feedback Algorithm
MPI-I-2003-1-013	I. Katriel, S. Thiel	Fast Bound Consistency for the Global Cardinality Constraint
MPI-I-2003-1-012		- not published -
MPI-I-2003-1-011	P. Krysta, A. Czumaj, B. Voecking	Selfish Traffic Allocation for Server Farms
MPI-I-2003-1-010	H. Tamaki	A linear time heuristic for the branch-decomposition of planar graphs
MPI-I-2003-1-009	B. Csaba	On the Bollobás – Eldridge conjecture for bipartite graphs
MPI-I-2003-1-008	P. Sanders	Polynomial Time Algorithms for Network Information Flow
MPI-I-2003-1-007	H. Tamaki	Alternating cycles contribution: a strategy of tour-merging for the traveling salesman problem
MPI-I-2003-1-006	M. Dietzfelbinger, H. Tamaki	On the probability of rendezvous in graphs
MPI-I-2003-1-005	M. Dietzfelbinger, P. Woelfel	Almost Random Graphs with Simple Hash Functions
MPI-I-2003-1-004	E. Althaus, T. Polzin, S.V. Daneshmand	Improving Linear Programming Approaches for the Steiner Tree Problem
MPI-I-2003-1-003	R. Beier, B. Vcking	Random Knapsack in Expected Polynomial Time
MPI-I-2003-1-002	P. Krysta, P. Sanders, B. Vcking	Scheduling and Traffic Allocation for Tasks with Bounded Splittability
MPI-I-2003-1-001	P. Sanders, R. Dementiev	Asynchronous Parallel Disk Sorting

MPI-I-2002-4-002	F. Drago, W. Martens, K. Myszkowski, H. Seidel	Perceptual Evaluation of Tone Mapping Operators with Regard to Similarity and Preference
MPI-I-2002-4-001	M. Goesele, J. Kautz, J. Lang, H.P.A. Lensch, H. Seidel	Tutorial Notes ACM SM 02 A Framework for the Acquisition, Processing and Interactive Display of High Quality 3D Models
MPI-I-2002-2-008	W. Charatonik, J. Talbot	Atomic Set Constraints with Projection