Approximating Distance Fields in Image Space

Fabian Scheer[1], Mario Marschner[2] and Stefan Müller[3]
[1]Daimler AG, [2]Daimler Protics GmbH, [3]University of Koblenz-Landau

Abstract
Fast proximity query algorithms are needed for a vast majority of computer graphics applications, ranging from visualization, simulation, animation, modelling, virtual and augmented reality to computer games. We present a novel and fast approach to approximate distance fields in image space at interactive frame rates. The vicinity of a point is sampled and the minimal distance to its surrounding geometry is determined based on the scenes depth values obtained by multiple stencil routed A-Buffers. Two sampling methods are compared with ground truth data in image space precision. Additionally, we introduce a method to determine the distance of an object to the next occluded object and a method to sample a directed distance field. The algorithms perform entirely on the GPU at interactive frame rates without the need for any precomputations and can handle dynamic content of arbitrary and deformable objects in massive data sets, making them feasible for a variety of applications.

1. Introduction

Fast proximity queries are a major issue in research for decades and have a strong impact on games, industrial and medical applications. Distance fields (DFs) are a representation where at any point in the field the distance to the closest point on any object in the domain is known [JBS06]. Taking advantage of the evolved capabilities of graphics hardware recent approaches [MRS08, SGG*07] achieve interactive or real-time frame rates. Nevertheless, complex and dynamic scenes with massive data and deformable objects are still a challenging task. Nowadays, many expensive effects, like radiosity or ambient occlusion obtain huge speed ups by image space approximations, including the following advantages:

- Computation mostly independent from polygon count
- Feasible for fully dynamic content and deformable objects
- Entirely computed on the GPU
- No precomputations needed

Thus, screen space approaches get attractive for DF computations, especially for massive data sets with dynamic content at a large spatial extent. Therefore, we present a new and fast approach to approximate DFs in screen space, which we call *screen space distance fields* (SSDF), capable to handle complex and wide spread scenes, typical for industrial factory planning scenarios with more than 15 mio. polygons. Multiple stencil routed A-Buffers acquire depth values of fragments inside the view frustum and serve as a data structure. With this information the DF of an arbitrary object can be sampled and the minimal distance to objects in the domain determined. Furthermore, we present methods to compute a directed DF to the next occluded object and a directed DF of an arbitrary hemispherical direction of a point. To account for a precise solution on older graphics hardware a refinement step using inter-frame information is presented. Finally, an overview of the run time properties and an analysis of the precision and computational effort is given.

2. Related Work

Multi fragment depth acquisition

On current graphics hardware occluded fragments are discarded by a depth test. For the acquisition of multiple fragments per pixel many research has been done over the years. According to [Car84] an A-Buffer is a list of all fragments per pixel sorted in depth order. [Eve01] achieves such a sorted fragment list via depth peeling, applying a seperate render pass for every depth layer of the scene. Nevertheless, *N* passes are needed to acquire *N* layers, involving huge costs for complex scenes. [LBQ06] peels multiple layers simultaneously, but is limited to RGBA8 precision and merely achieve a doubled frame rate. [BCL*07] proposes the K-Buffer to access multiple fragments simultaneously through read-modify-write (RMW) operations. Anyhow, [LFEH09] reported serious artifacts of the K-Buffer due to RMW hazards and introduce efficient depth peeling via bucket sort. Although performing fast, they rely on a uniform distribution of the depth values. [MB07] exploits the multisampling capabilities of modern graphics hardware to acquire eight

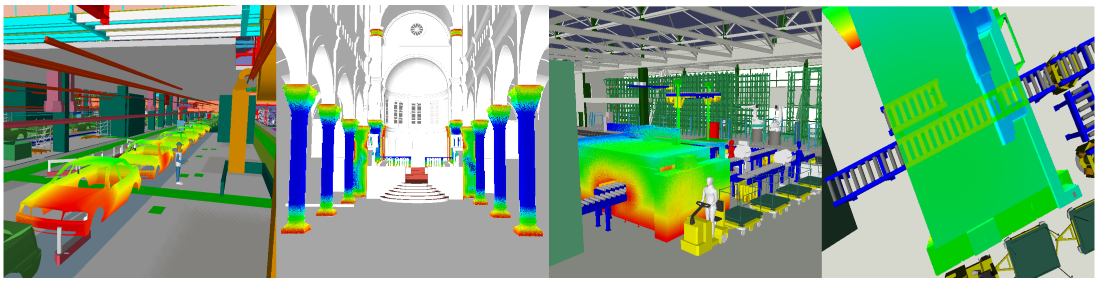

Figure 1: *DF visualization in false colors, ranging from red for near objects to blue for more distant objects. Refined SSDFs with 4096 samples (first). 128 halton samples for the Sibenik scene with a 400x400 A-B resolution (second). 256 random samples for the Factory scene with a 800x800 A-B resolution (third). X-Ray visualization of occluded objects (fourth).*

fragments per pixel via subpixel stencil routing in a single pass. Further fragments of the same pixel can be detected by an occlusion query and acquired by an additional pass. Up to 254 fragments can be captured representing a sufficient order of magnitude for handling massive data sets.

Distance fields

Distance fields are used for many purposes including object representation, proximity queries and collision detection. They are broadly categorised according to the model representation such as image, volume or polygonal representations [SGG*07]. [JBS06] describes several techniques in a survey. For a vast majority of approaches polygonal meshes must be closed, oriented 2-manifolds or organized in a hierarchical structure. In the last years interactive approaches emerged using the GPU. Many algorithms focus on grid-based techniques by rasterizing the distance function of slices using voronoi regions [SGGM06, HIZLM01]. [HIZLM01] describes interactive proximity queries using a coarse geometric localization for possibly intersecting or closest regions followed by image space queries to compute the low-level proximity information of 2D slices in a grid. [SGGM06] decomposes distance functions into linear factors and compute discretized DFs. Nevertheless, their approach implies an additional overhead due to read backs to the CPU, accuracy depends on grid resolution and highly tesselated objects lead to a decreased performance. Other approaches like [SGG*07] apply voronoi regions, but rely on precomputations implying costly data structure updates for large data sets with dynamic or deformable content. [MRS08] performs fast proximity queries on the GPU by sampling the DF in image space and presents parallel reduction techniques to minimise the readback bottleneck. However, at least one colliding object has to be closed and rigid.

3. Screen Space Approach

3.1. Setup of multiple A-Buffers

To acquire occluded fragments inside the view frustum stencil routed A-Buffers (A-Bs), described by [MB07], are used. Eight depth values with 16-bit floating point precision can be stored in one pass. Since a single A-B can miss depth values for complex and large data sets [JH08], multiple A-Bs are applied using a fragment overflow detection per pixel [MB07] and multiple render passes. Due to the limitation of an 8-bit stencil buffer on current graphics hardware, a maximum of 254 fragments can be stored per pixel. Anyhow, we experienced that 64 linear depth values acquired by eight passes are sufficient for most scenes. To account for a fast rendering of industrial factory planning scenarios with complex and massive data at a large extent (e.g. > 25 mio. polygons) a visibility guided renderer (VGR) [KBHP07] fills the A-Bs. Even though the buffers are consistently filled by the VGR, in some rare cases fragments can be missed due to a violation of the fragments rasterization order caused by the out of core reload mechanism of the VGR when massive amounts of data are inside the view frustum.

3.2. Sampling distance fields in screen space

A distance field DF^D is defined as a representation where at any point in the domain D the closest distance to any object in D is known [JBS06]. In this paper we focus on the sampling of unsigned distance fields and define DF^D for an arbitrary point p in the domain D as

$$DF^D = \min_{\forall s \in M, p \in D} \|p - s\|, \tag{1}$$

where M represents the set of all surface points in the domain and $\|x\|$ the euclidean norm. Since DF^D is approximated by sampling, we define a screen space distance field (SSDF) DF^D_S as

$$DF^D_S = \min_{\forall s \in S, p \in D} \|p - s\|, \tag{2}$$

where S represents a sampled subset of M. Once the A-Bs are filled a fullscreen quad is drawn according to the deferred shading approach with an equivalent resolution of the A-Bs. Thereby, the desired distance field DF^D_S is sampled in a postprocessing step. Even though the sampling of a DF of an arbitrary point in space is possible, we focus on SSDFs of visible surface points to visualize them for safeguarding methods in the automotive industry (see fig. 1). Such a SSDF is evaluated by determining the minimal euclidean distance of the visible surface point $p \in M$ to other sampled surface points $s \in S$ evaluated by the stored fragments in the A-Bs. To determine the minimal depth value of p closest to the camera $8 \cdot N_{A-Buf}$ comparision are needed, where N_{A-Buf} depicts the number of A-Bs. The SSDF at each fragment p of a considered object is then computed by comparing the euclidean distance of all other fragments for every screen space sample position (x_i, y_i). Thus a total of

$$pixel_{obj} \cdot \left(8 \cdot N_{A-Buf} + \left(s_{num} \cdot 8 \cdot N_{A-Buf}\right)\right) \qquad (3)$$

comparisons are needed. $pixel_{obj}$ denotes the sum of visible pixels of the object and s_{num} the number of samples.

Image space sampling

Sampling the set M by the subset S forms the basis for a well approximated DF. Therefore, a gaussian random sampling and a uniform distributed halton sampling within a certain area per pixel in image space is implemented. The covered area of both methods can be modified by a scale parameter [SK10] dependent on the respective depth value influencing the size of the domain D. Hence, reconstructing the view space position by backprojection, the length of the vector from p to s can be determined to find the surface point with minimal distance to p. Fragments acquired by the A-Bs are processed in rasterization order and not sorted according to their depth values, since we experienced that such brute force comparison is faster than sorting the fragments and excluding some by determining if they lie in the domain D.

Directed view space sampling

Additionally, we present a method to sample a DF in a specific direction called *directed screen space distance field* (DSSDF). Halton samples are created in a restricted region of a precomputed lattitude-longitude texture (LUT) which encodes points of the unit sphere as direction vectors. Looking up the samples texture coordinates yields direction vectors, which are used to create view space samples in a specific hemispherical direction of a considered view space point. Projecting these samples to image space, the hit pixels can be determined. Dividing the view space length of a sample vector by the number of covered pixels, provides us with a stepwidth in view space to incrementally step along and assuring to hit any pixel in image space by backprojection. Hence, the A-Bs can be evaluated and the depth difference with the fragments of the hit pixel can be checked. Thus, the closest fragment of the surrounding objects in the desired direction yields the minimal distance in the domain. Setting the number of samples to one allows the exact consideration of the direction along a polygons surface normal. Due to its nature the method is computationally more expensive than the sampling methods described above. On the other side the minimal distance along a given direction can be determined.

Accounting for occluded distances

Similar to DSSDFs, the distance of an occluded object to the occluder can be computed (see fig. 1). First, the minimal depth value closest to the camera in the A-Bs is determined, obtaining the depth value and the id, which we stored in the material color, of the visible object. Searching for the closest fragment with a different object-id in the A-Bs yields the minimal distance to the next occluded object per pixel and is suitable for the computation of the penetration depth of an object. The method is henceforth called X-Ray visualization. To account for the distance of the backface of a visible object to the next occluded one an additional search over the values in the A-Bs has to be done to capture the depth of the back side fragment. Even though the method is heavily view dependent it may be useful for visualization purposes.

Refinement

To account for the visualization of SSDFs for visual safeguarding operations and for older graphics hardware, SSDFs can be refined from frame to frame whenever the camera is fix. In subsequent frames different sampling positions are used and compared to the minimal distances of the previous frames. Thus, more samples can be provided consequently refining the correctness (see fig. 1). This can be useful for applications relying on visual judgements of distances and for older graphics hardware to run at interactive rates.

4. Results

Table 1 shows the results for two scenes. A desktop PC with an Intel Core i7 920 with 3GB RAM and a NVIDIA GTX 285 graphics card were used. Interactive frame rates are achieved for a scene with a moderate polygon count and for complex scenes with massive amounts of data. Dynamic and deformable content of arbitrary objects can be handled without a loss of performance. Object distances in virtual worlds can be interactively explored, e.g. to check if cars can pass through an assembly line within given safety zones (see fig. 1). For complex scenes the A-B acquisition is the most expensive step. Thus, smaller A-B resolutions speed up the algorithms. For the final image composition the results are upsampled. Halton sampling performs slightly better and more precise than random sampling due to the uniform sample distribution. To compare the sampling results a ground truth in image space was implemented. Therefore, the minimal distance of a point was determined by comparing all fragments in the A-Bs for every screen pixel. Even though the ground truth is not correct in a mathematical sense of exact intersection points due to the limited three places after the decimal point precision of the 16-bit floating point depth values, an error rate of the sampling methods in image space precision can be computed. All scenes are compared with a 800x800 ground truth. Precision and performance depends on the A-B

	Sibenik (3 passes, 700.000 polygons)			**Factory (8 passes, 15.4 mio. polygons)**		
A-B resolution	**200x200**	**400x400**	**800x800**	**200x200**	**400x400**	**800x800**
Random, 64 samples	20 ms **&** 8.5%	59 ms **&** 6%	210 ms **&** 5.5%	22 ms **&** 15%	73 ms **&** 13.3%	235 ms **&** 13%
Random, 128 samples	39 ms **&** 6.5%	118 ms **&** 4.5%	411 ms **&**4%	41 ms **&** 11%	140 ms **&** 9.6%	441 ms **&** 9%
Random, 256 samples	78 ms **&** 5.5%	242 ms **&** 3.5%	808 ms **&** 2.5%	81.9 ms **&** 8%	272 ms **&** 6.6%	852 ms **&** 6%
Halton, 64 samples	18 ms **&** 7.5%	53 ms **&** 6.5%	170 ms **&** 5%	17 ms **&** 14%	49 ms **&** 13%	156 ms **&** 12.5%
Halton, 128 samples	35 ms **&** 6%	104 ms **&** 4%	331 ms **&** 3%	32 ms **&** 10%	91 ms **&** 9.4%	290 ms **&** 9%
Halton, 256 samples	69 ms **&** 5.5%	206 ms **&** 3.5%	646 ms **&** 2%	63 ms **&** 7%	177 ms **&** 6.5%	553 ms **&** 6%
X-Ray	1.4 ms	4.3 ms	14.8 ms	2.2 ms	5.7 ms	27.3 ms
1 Vector	8 ms	36 ms	82 ms	19 ms	53 ms	169 ms
A-B creation	3 passes · 1.6 ms	3 passes · 1.8 ms	3 passes · 3.3 ms	8 passes · 11.7 ms	8 passes · 12.2 ms	8 passes · 15.5 ms

Table 1: *Performance and error results. In three passes all fragments in the frustum are acquired for scene Sibenik and in eight passes for scene Factory. The A-B creation time is for one A-B and has to be multiplied with the number of passes for the total time. Timings for the two sampling methods are listed, followed by the error rate that was determined against the image space ground truth. Times for a X-Ray DSSDF and a DSSDF along the surface normal with one sample (1 Vector) are also listed.*

resolution, size of the domain and sample count. For scene Sibenik the radius of the domain was two meters and for scene Factory three meters. Thus, the errors for scene Factory are higher (see table 1), since more samples are needed for larger domains to achieve a high precision. Using the refinement, the error can be reduced under two percent. Nevertheless, a few drawbacks of the proposed methods have to be considered. First, we rely on a discretized scene representation due to the rasterization in a specific resolution and bit-depth. Increasing the A-Bs bit depth to 32 is possible, but reduces the MSAA samples to four on current graphics hardware. Furthermore, only objects inside the view frustum are processed leading to possible artifacts at the viewport edge. This can be resolved by enlarging the view frustum during A-B acquisition and SSDFs computation, followed by a pass in the original resolution. A more severe problem implies a miss of fragments for polygons that are orthogonally aligned to the viewer, since they are not rasterized. Using the VGR enables the processing of massive data, but can lead to a miss of some fragments due to the violation of fragments rasterization order by the out of core reload mechanism triggered by massive data inside the view frustum. Finally, the algorithms are strongly fill-rate limited.

5. Conclusion and future work

Fast approximations of DFs at interactive rates are presented handling dynamic or deformable content of arbitrary shaped objects. Complex scenes with massive amounts of data are handled due to multiple A-Bs supported by a VGR. DFs are sampled in screen space avoiding additional read-backs. Additionally, DFs are sampled in a specific direction. Modifying the SSDF algorithm to discard samples of a specific direction can also be used to generate DSSDFs, allowing better performances and will be part of our future work. Furthermore, a method to account for DSSDFs of occluded objects is presented. Overcoming the view dependency to determine the penetration depth will also be part of our future research. The algorithms operate entirely on the GPU and are well suited to benefit of future generation graphics hardware.

References

[BCL*07] Bavoil L., Callahan S. P., Lefohn A., Comba Jo a. L. D., Silva C. T.: Multi-fragment effects on the gpu using the k-buffer. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games* (2007). 1

[Car84] Carpenter L.: The a-buffer an antialiased hidden surface method. In *SIGGRAPH proceedings of the Conference on Computer graphics and interactive techniques* (1984). 1

[Eve01] Everitt C.: Interactive order-independent transparency. *White paper, nVIDIA 2*, 6 (2001). 1

[HIZLM01] Hoff III K., Zaferakis A., Lin M., Manocha D.: Fast and simple 2D geometric proximity queries using graphics hardware. In *Proceedings of the 2001 symposium on Interactive 3D graphics* (2001), ACM, p. 148. 2

[JBS06] Jones M., Baerentzen J., Sramek M.: 3D distance fields: A survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics 12* (2006). 1, 2

[JH08] Jang H., Han J.: Fast collision detection using the A-buffer. *The Visual Computer* (2008). 2

[KBHP07] Kasik D., Brüderlin B., Heyer M., Pfützner S.: Visibility-guided rendering to accelerate 3d graphics hardware performance. In *ACM SIGGRAPH Courses* (2007). 2

[LBQ06] Liu B.-Q. Wei L.-Y. X. Y.-Q.: *Multi-layer depth peeling via fragment sort*. Tech. rep., Microsoft Research Asia, 2006. 1

[LFEH09] Liu F. Huang M.-C. L. X.-H., E.-H. W.: Efficient depth peeling via bucket sort. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009* (2009). 1

[MB07] Myers K., Bavoil L.: Stencil routed A-buffer. In *International Conference on Computer Graphics and Interactive Techniques* (2007). 1, 2

[MRS08] Morvan T., Reimers M., Samset E.: High performance GPU-based proximity queries using distance fields. In *Computer Graphics Forum* (2008), vol. 27. 1, 2

[SGG*07] Sud A., Govindaraju N., Gayle R., Andersen E., Manocha D.: Surface distance maps. In *Proceedings of Graphics Interface 2007* (2007). 1, 2

[SGGM06] Sud A., Govindaraju N., Gayle R., Manocha D.: Interactive 3D distance field computation using linear factorization. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games* (2006). 2

[SK10] Scheer F., Keutel M.: Screen space ambient occlusion for virtual and mixed reality factory planning. *Journal of WSCG 18*, 1-3 (2010). 3