Tiled Depth of Field Splatting

Kai Selgrad Linus Franke Marc Stamminger

Computer Graphics Group, University of Erlangen-Nuremberg, Germany

Abstract

We present a method to compute post-processing depth of field (DOF) that produces more accurate results than previous approaches. Our method is based on existing approaches, namely DOF rendering by splatting and fast, tile-based particle accumulation. Using tile-based accumulation allows us to correctly sort out of focus pixels and apply proper alpha-blending to avoid artifacts commonly encountered with filter-based depth of field methods.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.3]: Image Generation—Display algorithms

1. Introduction

Depth of field is an important effect to achieve more plausible results in photography, movies and games, and a commonly used artistic element in itself. Physically based simulation of this effect is, however, computationally demanding as it requires an additional integration over the lens. Therefore, real-time applications such as games usually rely on lower fidelity algorithms that generate plausible, albeit not physically correct results. Similarly to other, fast DOF implementations, our method is a post-processing filter-chain, but we strive to provide higher quality results while not compromising much on render times. This is achieved by combining two existing ideas. Firstly, that high quality depth of field can be computed by splatting [PC81, KZB03], and secondly that those splats can be blended using fast, tile-based particle accumulation [HMY12, Tho14] instead of a costly global sorting step. This combination has, to the best of our knowledge, not been evaluated before.

2. Related Work

Depth of Field Rendering. The most common techniques for computing depth of field in interactive applications are *gathering* approaches, as they map well to hardware rendering. These approaches accumulate data from nearby fragments from a sharp input image to produce a blurred representation. To this end the circle of confusion is computed for each fragment and the area around that fragment is averaged by an image-space filter. To remove artifacts from averaging in-focus and out-of-focus fragments, a bilateral filter can be used to check for problematic cases [RTI03]. Splatting based approaches [PC81, KZB03], on the other hand, can be characterized as *scattering* as they distribute the contribution of a fragment over its circle of confusion in screen space. With properly sorted splats scattering algorithms avoid artifacts produced by gathering methods, however at the cost of having to globally sort the splats.

Efficient Particle Rendering. Harada et al. [HMY12] present a method that allows efficient rendering with many lights in a forward rendering context. This is usually a very expensive setup as for each fragment the contribution of all light sources has to be accumulated, even if many do not contribute to any given fragment. Harada et al. propose to first collect the set of relevant lights for each pixel by splatting their bounding geometry and accumulating their indices in tiles. During scene rendering it is then possible to traverse only the lights registered in a pixel's tile. Based on this work, Thomas [Tho14] shows how the same approach can be applied to render particles using a GPU-only particle system: Particles are binned into tiles, sorted to provide for correct blending and finally traversed and the result blended on top of the rendered image.

3. Tiled Depth of Field Rendering

Limitations of Gathering Approaches. When computing depth of field via gathering care has to be taken during filtering to prevent sharp features to be filtered into blurred background objects [RTI03], as otherwise the filter will produce halos around sharp objects in front of a blurred background. This can be avoided by using a bilateral filter that, for out-of-focus positions, only collects the contribution of fragments that are also out of focus, or behind the target fragment's depth. To avoid popping artifacts due to this condition the transition can further be smoothed. Filtering approaches also exhibit a more subtle artifact: The nature of an image-space filter is that it computes a weighted average of the contribution of the fragments in the filter radius. Therefore the relative z-order of the participating fragments is not taken into account. Note that correct ordering can be established and used for alpha-blending with gathering approaches, however for larger filter sizes per-pixel sorting will be prohibitively expensive.

Algorithm Overview. Based on the work of Thomas [Tho14] we propose to remove the bottleneck of global sorting from splatting

Merge	—	2×2	4×4
Avgerage list length	1160 entr.	384 entr.	199 entr.
Build lists	1.35 ms	1.07 ms	1.08 ms
Sort lists	7.28 ms	4.71 ms	4.04 ms
Apply blur	14.36 ms	5.85 ms	2.95 ms
Sum	22.99 ms	11.63 ms	8.07 ms

Table 1: *Detailed frame times of our method. Standard filter-based post processing [RTI03] for this setup can be applied in 8.15 ms on a Geforce GTX Titan.*

based DOF approaches by applying efficient particle accumulation. The algorithm, starting from an image ready for post-processing, is as follows: We first compute the circle of confusion for each pixel and determine which tiles it overlaps. The pixel's color, depth and circle of confusion are recorded for each of those tiles in a per-tile linked-list. We then sort the per-tile lists in parallel. This sort replaces the global operation usually associated with DOF splatting. With the sorted lists we use front-to-back alpha-blending for fast accumulation. In the following we describe implementation details that enable us to apply this scheme to the vast number of splats generated for DOF.

Implementation Details. This scheme maps particularly well to hardware: Firstly, the atomic operations required to construct per-tile lists have become very cheap. Secondly, sorting the elements in a tile can be mapped to tile-parallel radix sort with CUDA where each thread block sorts a single tile in shared memory. Sorting is further sped up by using packed data to allow fast 16-byte load/store operations (2 bytes for each HDR color channel and the screen space positions, 4 bytes for blur size). Finally, accumulation from within a single tile can be parallelized such that all threads in a warp read the same tile and thus list data can be loaded efficiently to shared memory, leading to very fast traversal.

In our implementation we employ a tile size of 16×16 pixels to prevent the lists from becoming too long. We also limit the blur size to overlap at most four tiles (i.e. 32×32). This provides locality for store operations while also keeping lists short. Since sorting and traversal time both directly depend on the length of the lists we provide a simple means to trade minor reduction in quality for significantly increased rendering performance: Similarly to Selgrad et al. [SRP*15] it is possible to merge neighboring fragments with very close depth values. We show results for merging 2×2 and 4×4 pixels of out-of-focus regions with similar depth.

Results. In the following we show metrics obtained for the image shown in Figure 1 when rendered on a Geforce GTX Titan. Figure 1 shows standard filtered depth of field (left) [RTI03] and results from splatting using our method. Most noteworthy is the difference at the railing where the bilateral filter does not distinguish between out of focus objects of different (but nearby) depth values. Our result using alpha blending on sorted tile lists is much more accurate.

Table 1 shows the influence of list length on the render time of the different stages of our algorithm. It is visible that even minimal merging (that does not introduce noticeable difference) already provides a considerable performance gain. With more aggressive merging our method arrives at the performance of filter-based approaches while still maintaining higher quality.

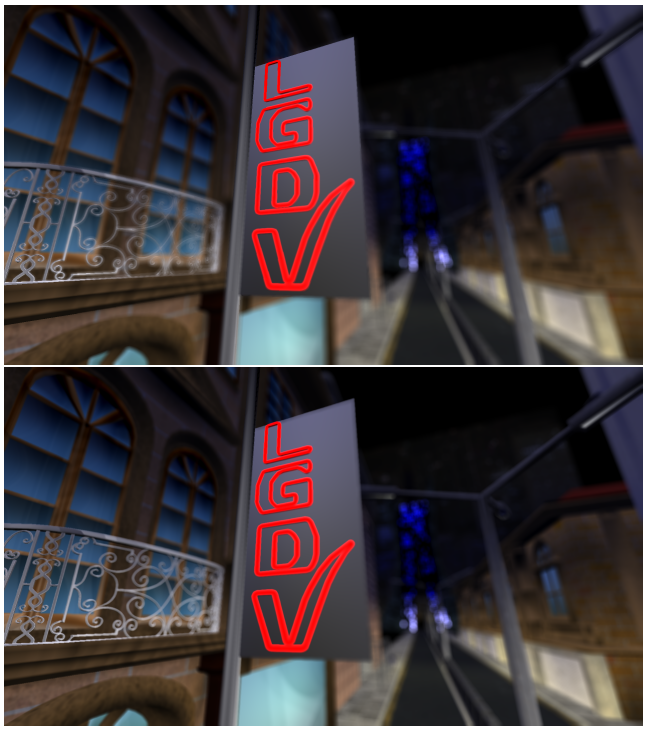

Figure 1: *Standard filtered DOF (top) and our method (bottom). Note how with our method the railing is more clearly in front of the windows.*

Conclusion. We have shown how two established approaches combine to yield a higher quality depth of field algorithm while not compromising on render times. The two approaches combine in a straightforward fashion, but an evaluation of this use case and the required extensions to gain competing performance has not been presented earlier.

References

[HMY12] HARADA T., MCKEE J., YANG J. C.: Forward+: Bringing deferred lighting to the next level. In *Eurographics 2012 - Short Papers Proceedings, Cagliari, Italy, May 13-18, 2012* (2012), pp. 5–8. 1

[KZB03] KRIVANEK J., ZARA J., BOUATOUCH K.: Fast depth of field rendering with surface splatting. In *Computer Graphics International, 2003. Proceedings* (2003), IEEE, pp. 196–201. 1

[PC81] POTMESIL M., CHAKRAVARTY I.: A lens and aperture camera model for synthetic image generation. In *Proceedings SIGGRAPH 1981* (1981), ACM, pp. 297–305. 1

[RTI03] RIGUER G., TATARCHUK N., ISIDORO J. R.: Real-time depth of field simulation. In *ShaderX2: Shader Programming Tips and Tricks with DirectX 9.0*, Engel W., (Ed.). Wordware, Plano, Texas, 2003. 1, 2

[SRP*15] SELGRAD K., REINTGES C., PENK D., WAGNER P., STAMMINGER M.: Real-time Depth of Field Using Multi-layer Filtering. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2015), i3D '15, ACM, pp. 121–127. 2

[Tho14] THOMAS G.: Compute-Base GPU Particle Systems, 2014. GDC'14. 1