Path Space Filtering

Alexander Keller, Ken Dahm, and Nikolaus Binder

Abstract The efficiency of quasi-Monte Carlo integro-approximation for light transport simulation is increased by locally smoothing the contribution of path space samples before averaging them. The proposed deterministic algorithm is constructed such that it converges to the solution of the underlying integro-approximation problem. The improvements and wide applicability of the consistent method are demonstrated by visual evidence.

1 Introduction

Light transport simulation comprises of summing up the contributions of light transport paths that connect sensors and light sources. As illustrated in Fig. 1, such light transport paths may be sampled by following photon trajectories from the lights, tracing paths from the camera, and connecting such path segments by proximity (photon mapping) and shadow rays (both dashed in black).

Modeling with physical entities like cameras, lights, and materials on top of the scene surface, light transport simulation may deliver photorealistic images. However, as analytic solutions are out of reach, simulation algorithms have to rely on sampling path space. Depending on the complexity of the model, the inherent noise of sampling may vanish only slowly during the course of computation.

Smoothing the contribution of light transport paths before reconstructing the image can efficiently reduce this noise. So far, intermediate approximations are computed for this purpose. However, removing frequent visual artifacts due to insufficient approximation usually forces simulation from scratch. In addition, adapting parameters of such methods requires quite extensive expertise.

Alexander Keller · Ken Dahm · Nikolaus Binder
NVIDIA, Fasanenstr. 81, 10623 Berlin, e-mail: keller.alexander@gmail.com, e-mail: ken.dahm@googlemail.com, e-mail: nikolaus.binder@googlemail.com

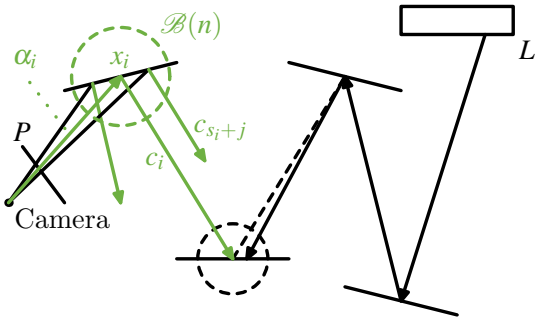

Fig. 1 Illustration of path space filtering: The green part of the schematic illustrates the principle of smoothing the contribution c_i to the vertex x_i of a light transport path by averaging contributions c_{s_i+j} to vertices inside the ball $\mathscr{B}(n)$. The averaged contribution $\bar{c}_i$ then is multiplied by the attenuation α_i along the path segment towards the camera and accumulated on the image plane P. This is complementary to photon mapping (see the dashed circle), where incident path segments are connected to a query ray. In analogy to progressive photon mapping, the radius $r(n)$ of the ball $\mathscr{B}(n)$ must vanish with the number n of samples in order to guarantee a consistent algorithm.

We therefore propose a simple and efficient deterministic algorithm that has a tiny set of parameters. Persistent approximation artifacts and tedious parameter tuning are avoided by the consistency of the scheme. While the algorithm unites the advantages of previous work, it also provides the desired noise reduction.

2 Consistent Weighted Averaging

Considering the i-th out of a total of n light transport paths, selecting a vertex x_i suitable for filtering the radiance contribution c_i of the light path segment towards x_i also determines the attenuation α_i along the path segment towards the camera (see Fig. 1). While any or even multiple vertices of a light transport path may be selected, a simple and practical choice is the first vertex along the path from the camera whose optical properties are considered sufficiently diffuse.

According to [5, 6], one low discrepancy sequence is transformed to sample path space in contiguous batches of $b^m \in \mathbb{N}$ light transport paths, where for each path one selected tuple (x_i, α_i, c_i) is stored for path space filtering. As the memory consumption is proportional to the batch size b^m, it is straightforward to determine the maximum natural number m given the size of the tuples and the maximum size of a memory block.

Processing the batch of b^m paths starting at index $s_i := \left\lfloor \frac{i}{b^m} \right\rfloor b^m$, the image is formed by accumulating $\alpha_i \cdot \bar{c}_i$, where

$$\bar{c}_i := \frac{\sum_{j=0}^{b^m-1} \chi_{\mathscr{B}(n)}(x_{s_i+j} - x_i) \cdot w_{i,j} \cdot c_{s_i+j}}{\sum_{j=0}^{b^m-1} \chi_{\mathscr{B}(n)}(x_{s_i+j} - x_i) \cdot w_{i,j}} \tag{1}$$

input at 1 sample per pixel path space filtered filter iterated 4 times

Fig. 2 Iterated weighted averaging very efficiently smooths the solution by relaxation at the cost of losing some detail. Obviously, path space filtering brightens up the image by replacing black pixels of the input with the weighted averages. Model courtesy G. M. Leal LLaaguno.

is the weighted average of the contributions c_{s_i+j} of all vertices x_{s_i+j} in a ball $\mathscr{B}(n)$ of radius $r(n)$ centered in x_i normalized by the sum of weights $w_{i,j}$ as illustrated in Fig. 1. Iterating the averaging process given by Eqn. 1, i.e. computing $\bar{\bar{c}}_i$ from $\bar{c}_i$ and so on, results in a dramatic speed up at the cost of some blurred illumination detail as can be seen in Fig. 2. While [14, 15] already showed that a very sparse set of samples may provide sufficient information for high quality image synthesis, path space filtering provides a simpler, faster, and consistent algorithm.

Centered in x_i, the characteristic function $\chi_{\mathscr{B}(n)}$ always includes the i-th path (as opposed to for example [21]). Therefore, given non-zero weights $w_{i,j}$ (see Sec. 2.1), an initial radius r_0 (see Sec. 2.2), and a radius

$$r(n) = \frac{r_0}{n^\alpha} \text{ for } \alpha \in (0,1) \tag{2}$$

vanishing with the total number n of paths guarantees $\lim_{n\to\infty} \bar{c}_i = c_i$ and thus consistency. As a consequence, all artifacts visible during progressive computation must be transient, even if they may vanish slowly. Note that this holds for the iterated weighted average, too. In fact having a small radius to hide the transient artifacts is goal competing with a large radius to include as many as possible contributions in the weighted average.

Fig. 3 illustrates the noise reduction, transient artifacts, and consistency by the straightforward application of the algorithm to path space samples of a path tracer with next event estimation and implicit multiple importance sampling [13]. The lighting consists of an environment map and a directional light. The first hit points as seen from the camera are stored as the vertices x_i.

In spite of the apparent similarity of Eqn. 1, methods used for scattered data interpolation [18, 8], and weighted uniform sampling [16, 20], there are principal differences: First, an interpolation property $\bar{c}_i = c_i$ would inhibit any averaging right from the beginning and second, $b^m \ll \infty$, as b^m is proportional to the required amount of memory to store light transport path. Nevertheless, the batch size b^m should be chosen as large as memory permits, because the efficiency results from simultaneously filtering as many vertices as possible.

Caching samples of irradiance and interpolating them to increase the efficiency of light transport simulation [24] has been intensively investigated [12] and has been

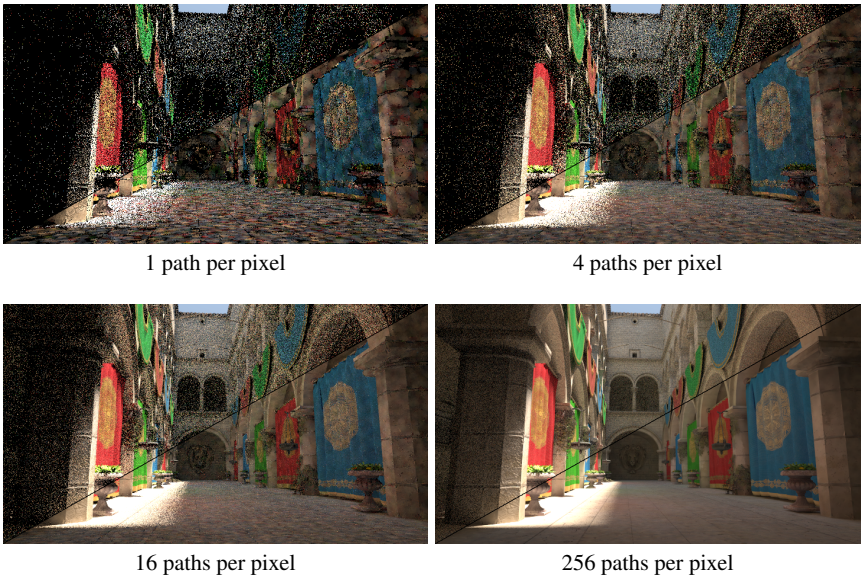

Fig. 3 The series of images illustrates progressive path space filtering. Each image shows the unfiltered input above and the accumulation of weighted averages below the diagonal. As more and more batches of paths are processed, the splotchy artifacts vanish due to the consistency of the algorithm as guaranteed by the decreasing range search radius $r(n)$. Model courtesy M. Dabrovic and Crytek.

implemented in many renderers (see Fig. 5). Scintillation in animations is the key artifact of this method, which appears due to interpolating cached irradiance samples that are noisy [11, Sec.6.3.2] and cannot be placed in a coherent way over time. Such artifacts require to adjust a set of multiple parameters followed by simulation from scratch, because the method is not consistent,

Other than irradiance interpolation, path space filtering efficiently can filter across discontinuities such as detailed geometry (for examples, see Fig. 6). It overcomes the necessity of excessive trajectory splitting to reduce noise in the cached samples, too, which enables path tracing using the fire-and-forget paradigm as required for efficient parallel light transport simulation. This in turn fits the observation that with an increasing number of simulated light transport paths trajectory splitting becomes less efficient. In addition, reducing artifacts in a frame due to consistency only requires to continue computation instead of starting over from scratch.

2.1 Weighting by Similarity

Although Eqn. 1 is consistent even without weighting, i.e. $w_{i,j} \equiv 1$, for larger radii $r(n)$ the resulting images may look overly blurred as contributions c_{s_i+j} become

Fig. 4 The effect of weighting: The top left image was rendered by a forward path tracer at 16 path space samples per pixel. The bottom left image shows the same algorithm with path space filtering. The improvement is easy to see in the enlarged insets. The right column illustrates the effect of the single components of the weights. From top to bottom: Using uniform weights, the image looks blurred and light is transported around corners. Including only samples with similar normals (middle), removes a lot of blur resulting in crisp geometry. The image at the bottom right in addition reduces texture blur by not filtering contributions with too different local attenuation by the surface reflectance properties. Finally, the bottom left result adds improvements on the shadow boundaries by excluding contributions that have too different visibility. Model courtesy M. Dabrovic and Crytek.

included in the average that actually never could have been gathered in x_i (see Fig. 4). In order to reduce this transient artifact of light leaking and to benefit from larger radii to include more contributions in the average, the weights $w_{i,j}$ should value how likely the contribution c_{s_i+j} could have been created in x_i by trajectory splitting.

Various heuristics for such weights are known from irradiance interpolation [12], the discontinuity buffer [4, 23], photon mapping [2], light field reconstruction [14, 15], and Fourier histogram descriptors [1]. The effect of the following weights of similarity is shown in Fig. 4:

Blur across geometry: The similarity of the normal n_i in x_i and other normals n_{s_i+j} in x_{s_i+j} can be determined by their scalar product $\langle n_{s_i+j}, n_i \rangle \in [-1,1]$. While

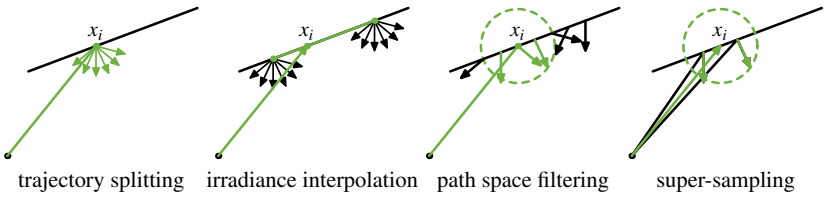

Fig. 5 In order to determine the radiance in x_i as seen by the long ray, many rays are shot into the hemisphere to sample the contributions. As this becomes too expensive due to the large number of rays, irradiance interpolation interpolates between cached irradiance samples that were smoothed by trajectory splitting. Path space filtering mimics trajectory splitting by averaging the contributions of paths in the proximity. Supersampling the information provided by the paths used for path space filtering is possible by tracing additional path segments from the camera. Note that then x_i does not have an intrinsic contribution.

obviously contributions with negative scalar product will be excluded in order to prevent light leaking through the backside of an opaque surface, including only contributions with $\langle n_{s_i+j}, n_i \rangle \geq 0.95$ (in our implementation) avoids light being transported across geometry that is far from planar.

Blur across textures: The images would be most crisp if for all contributions included in the average the optical surface properties would be evaluated in x_i. For surfaces other than diffuse surfaces, like for example glossy surfaces, these properties also depend on the direction of observation, which then must be explicitly stored with the x_i. Some of this additional memory can be saved when directions are implicitly known to be similar, as for example for query locations x_i as directly seen from the camera.

In situations where this evaluation is too costly or not feasible, the algorithm has to rely on data stored during path segment generation. Such data usually includes a color term, which is the bidirectional scattering distribution function (BSDF) multiplied by the ratio of the cosine between surface normal and direction of incidence and the probability density function (pdf) evaluated for the directions of transport. For the example of cosine distributed samples on diffuse surfaces only the diffuse albedo remains, because all other terms cancel. If a norm of the difference of these terms in x_{s_i+j} and x_i is below a threshold (0.05 in our implementation), the contribution of x_{s_i+j} is included in the average. Unless the surface is diffuse, the similarity of the directions of observation must be checked as well, which results in reduced blur across texture details and incorrect in-scattering on glossy materials.

Finally, [1] computes the weighted average for each component resulting from a decomposition of path space induced by the basis functions used to represent the optical surface properties.

Blurred shadows: Given a point light source, its visibility as seen from x_i and x_{s_i+j} may be either identical or different. In order to avoid sharp shadow boundaries to be blurred, contributions may be only included upon identical visibility. For ambient occlusion and illumination by an environment map, blurred shadows can

be reduced by comparing the lengths of each one ray shot into the hemisphere at x_i and x_{s_i+j} by thresholding their difference.

Using only binary weights that are either zero or one, the denominator of the ratio in Eqn. 1 amounts to the number of included contributions. Excluding contributions from the weighted average just does not reduce the noise level, which acts convenient to the eye. However, using the norms to directly to weight the contributions results in higher variance. This can be explained by the fact that under the assumption of perfect importance sampling, each path segment created by trajectory splitting would have been included with the same weight. In a similar way, using kernels (for examples see [19] or kernels used in the domain of smoothed particles hydrodynamics (SPH)) other than the characteristic function $\chi_{\mathscr{B}(n)}$ to weight contributions by their distance to the query location x_i increases the variance. This is even simpler to see, as obviously a smaller search radius inhibits more smoothing.

2.2 Range Search

The vertices x_{s_i+j} selected by the characteristic function $\chi_{\mathscr{B}(n)}$ centered at x_i efficiently may be queried by a range search using a hash grid [22], a bounding volume hierarchy or a kd-tree organizing the entirety of stored vertices in space, or a divide-and-conquer method [7] simultaneously considering all queries. As the sets of query and data points are identical, data locality is high and implementation is simplified.

Note that storing vertex information only in screen space even enables realtime path space filtering [1], however, can only query a subset of the neighborhood relations as compared to the full 3d range search. In fact, [1] improves on [17, Ch.4, p.83] with respect to similarity criteria and path space decomposition, while the basic ideas are similar and both are based on earlier attempts of efficient filtering approaches [4, 23, 10] in order to improve efficiency.

As already observed in [6], the parameter α in Eqn. 2 does not have much influence and $\alpha = \frac{1}{4}$ is a robustly working choice. In fact, the radius is decreasing arbitrarily slowly, which leaves the initial radius r_0 as the most important parameter.

As fewer and fewer contributions are averaged with decreasing radius, there is a point in time, where actually no averaging takes place any longer as only the central vertex x_i is included in the queries. On the one hand, this leads to the intuition that comparing the maximum of the number of averaged contributions to a threshold can be utilized to automatically switch off the algorithm. On the other hand, it indicates that the initial radius needs to be selected sufficiently large in order to include a meaningful number of contributions in the weighted averages from Eqn. 1.

The initial radius r_0 also may depend on the query location x_i. For example, it may be derived from the definition of the solid angle $\Delta\omega := \frac{\pi r_0^2}{d^2}$ of a disk of radius r_0 in x_i perpendicular to a ray at a distance d from the ray origin. For a fixed solid angle $\Delta\omega$, the initial radius

$$r_0 = \sqrt{\frac{d^2 \Delta\omega}{\pi}} \sim d$$

then is proportional to the distance d. The factor of proportionality may be either chosen by the user or can be determined using a given solid angle. For example, $\Delta\omega$ can be chosen as the solid angle determined by the area of 3×3 pixels on the screen with respect to the focal point. Finally, the distance d may be chosen as the length of the camera path towards x_i.

Note that considering an anisotropic footprint is not practical for several reasons: The requirement of dividing by the cosine between surface normal in x_i and the ray direction may cause numerical issues for vectors that are close to perpendicular. In addition the efficiency of the range search may be decreased, as now the query volume may have an arbitrarily large extent. Finally, this would result in possibly averaging contributions from vertices that are spatially far apart, although the local environment of the vertex x_i may be small such as for example in foliage or hair.

2.3 Complement to Progressive Photon Mapping

Photon mapping [2], on the one hand, aims to connect photon trajectories to camera path segments by proximity in order to capture the contribution of light transport paths that are difficult to sample due to the problem of insufficient techniques [9, Fig. 2]. These contributions are determined by querying the flux of photons inside a ball around a query point divided by the corresponding disk area in order to obtain radiance. [6] introduced a consistent and numerically robust quasi-Monte Carlo method for progressive photon mapping, while the references in the article reflect the latest developments in photon mapping.

Path space filtering, on the other hand, computes the contribution to a vertex by the weighted average from Eqn. 1 and includes light transport paths connected by progressive photon mapping. Despite all obvious algorithmic similarities and as illustrated in Fig. 1, path space filtering is a variance reduction technique complementary to progressive photon mapping: Without progressive photon mapping the contribution of difficult to sample light transport paths would be just missing or add high variance sporadically. In fact, the contribution to vertices obtained by photon mapping should not be included in path space filtering, as this would mean averaging twice very similar contributions, which does not improve the rendering at all. Contribution to that same vertex other than from photon mapping are included in the weighted average.

Although both methods are based on the same kind of range search in a point cloud, photon map query locations are not part of the photon cloud, while in path space filtering the ensemble vertices represents both data and query locations.

Temporally visible light leaks and splotches are due to a large range search radius $r(n)$, which allows for collecting light beyond opaque surfaces and due to the shape of the ball $\mathscr{B}(n)$ blurs light into disk-like shapes. If the local environment around a

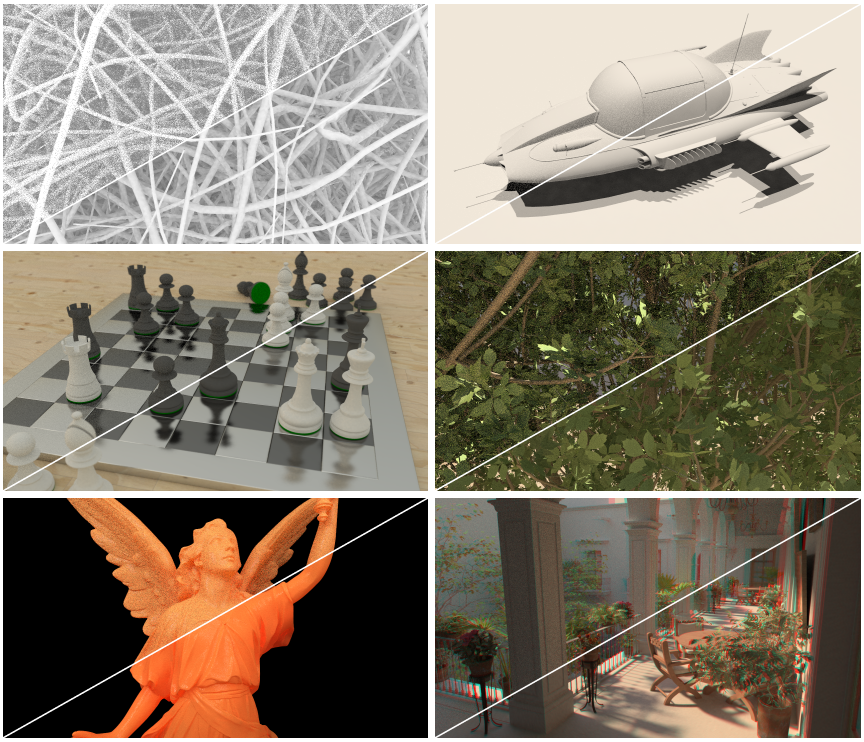

Fig. 6 The split image comparisons show how path space filtering reduces noise in ambient occlusion, shadows, light transport simulation, complex geometry, rendering translucent material, and across multiple cameras. Please zoom into the images to see more detail. Path space filtering can remove substantial amounts of noise, even across complex geometry (as shown in the bottom image). Models courtesy S. Laine, cgtrader, Laubwerk, Stanford Computer Graphics Laboratory, and G. M. Leal LLaaguno.

query point is not a disk, as for example close to a geometric edge, the division by the disk area in photon mapping causes an underestimation resulting in a darkening along such edges. While this does not happen for the weighted average, path space filtering may reduce contrast (see the foliage rendering in Fig. 6). In addition, so-called fire flies that actually are rarely sampled peaks of the integrand, are attenuated by the weighted average and therefore may look more like splotches instead of single bright pixels. Since both algorithms are consistent, all of these artifacts must be transient.

3 More Applications in Light Transport Simulation

Path space filtering is simple to implement and due to linearity (see Eqn. 1) works for any decomposition of path space including any variant of (multiple) importance sam-

pling. It can overcome the need for excessive trajectory splitting (see the schematic in Fig. 5) for local averaging in x_i in virtually all common use cases in rendering: Ambient occlusion, shadows from extended and/or many light sources (like for example instant radiosity), final gathering, ray marching, baking light probes and textures for games, rendering with participating media, or effects like depth of field simulation can be determined directly from path space samples. Some exemplary results are shown in Fig. 6 and some more applications are briefly sketched in the following:

Animations: A common artifact in animations rendered with interpolation methods is scintillation due to for example temporally incoherent cached samples, noisy cached samples, or temporal changes in visibility. Then parameters have to be tweaked and computation has to be started from scratch. Progressive path space filtering removes this critical source of inefficiency: Storing the next batch starting index s_i with each frame (see Sec.2), any selected frame can be refined by just continuing the computation as all artifacts are guaranteed to be transient.

Multiple views: In addition, path space filtering can be applied across vertices generated from multiple views. As such, rendering depth of field, stereo pairs of images, multiple views of a scene, rendering for light field displays, or an animation of a static scene can greatly benefit as vertices can be shared among all frames to be rendered.

Motion blur: Identical to [5], the consistent simulation of motion blur may be realized by averaging images at distinct points in time. As an alternative, extending the range search to include proximity in time allows for averaging across vertices with different points in time. In cases, where linear motion is a sufficient approximation and storing linear motion vectors is affordable, reconstructing the visibility as introduced in [14, 15] may improve the speed of convergence.

Spectral rendering: The consistent simulation of spectral light transport may be realized by averaging monochromatic contributions c_i associated to a wavelength λ_i. The projection onto a suitable color system may happen during the averaging process, where the suitable basis functions are multiplied as factors to the weights. One example of such a set of basis functions are the CIE XYZ response curves. One very compact continuous approximation of these response curves is introduced in [25].

Participating media and translucency: As the path space filtering works for any kind of path space samples, it is straightforward to apply it to the simulation of subsurface scattering and participating media in order to improve rendering performance. Fig. 6 features a statuette with light transported through translucent matter, where path space filtering has been performed across the surface of the statuette. At this level of efficiency, the consistent direct simulation may become affordable over approximations like for example bidirectional subsurface scattering distribution functions (BSSRDF) based on the dipole approximation [3].

Decoupling anti-aliasing from shading is straightforward by just sampling more camera path segments with query locations x_i and attenuations α_i as illustrated in Fig. 5. Note that similar to [21] queries with these locations may be empty due to the lack of the central contribution c_i and in that case cannot be considered in the

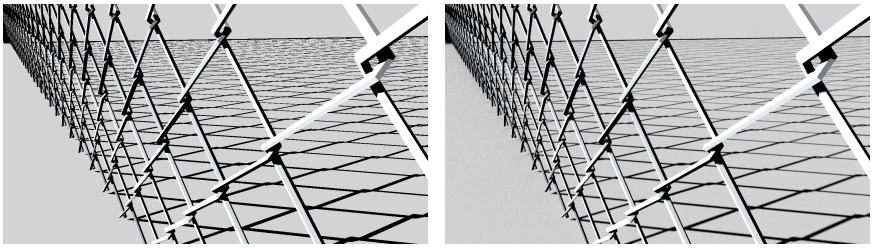

Fig. 7 The left image shows a fence rendered with one light transport path per pixel. The image on the right shows the result of anti-aliasing by path space filtering using the paths from the left image and an additional three camera paths per pixel. Model courtesy Chris Wyman.

accumulation process. Fig. 7 illustrates how anti-aliasing by super-sampling with path space filtering works nicely across discontinuities.

4 Conclusion

Path space filtering is simple to implement on top of any sampling-based rendering algorithm and has low overhead. The progressive algorithm efficiently reduces variance and is guaranteed to converge without persistent artifacts due to consistency. It will be interesting to explore the principle applied to integro-approximation problems other that computer graphics and to investigate how the method fits into context of multilevel Monte Carlo methods.

References

1. Gautron, P., Droske, M., Wächter, C., Kettner, L., Keller, A., Binder, N., Dahm, K.: Path space similarity determined by Fourier histogram descriptors. In: ACM SIGGRAPH 2014 Talks, SIGGRAPH '14, pp. 39:1–39:1. ACM, New York, NY, USA (2014). DOI 10.1145/2614106.2614117. URL http://doi.acm.org/10.1145/2614106.2614117
2. Jensen, H.: Realistic Image Synthesis Using Photon Mapping. AK Peters (2001)
3. Jensen, H., Buhler, J.: A rapid hierarchical rendering technique for translucent materials. ACM Trans. Graph. **21**(3), 576–581 (2002). DOI 10.1145/566654.566619. URL http://doi.acm.org/10.1145/566654.566619
4. Keller, A.: Quasi-Monte Carlo Methods for Photorealistic Image Synthesis. Ph.D. thesis, University of Kaiserslautern, Germany (1998)
5. Keller, A.: Quasi-Monte Carlo image synthesis in a nutshell. In: J. Dick, F. Kuo, G. Peters, I. Sloan (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2012, pp. 203–238. Springer (2013)
6. Keller, A., Binder, N.: Deterministic consistent density estimation for light transport simulation. In: J. Dick, F. Kuo, G. Peters, I. Sloan (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2012, pp. 467–480. Springer (2013)
7. Keller, A., Droske, M., Grünschloß, L., Seibert, D.: A divide-and-conquer algorithm for simultaneous photon map queries. Poster at High-Performance Graphics in Vancouver (2011).

URL http://www.highperformancegraphics.org/previous/www_2011/media/Posters/HPG2011_Posters_Keller1_abstract.pdf
8. Knauer, E., Bärz, J., Müller, S.: A hybrid approach to interactive global illumination and soft shadows. The Visual Computer: International Journal of Computer Graphics **26**(6-8), 565–574 (2010)
9. Kollig, T., Keller, A.: Efficient bidirectional path tracing by randomized quasi-Monte Carlo integration. In: H. Niederreiter, K. Fang, F. Hickernell (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2000, pp. 290–305. Springer (2002)
10. Kontkanen, J., Räsänen, J., , Keller, A.: Irradiance filtering for Monte Carlo ray tracing. In: D. Talay, H. Niederreiter (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2004, pp. 259–272. Springer (2004)
11. Křivánek, J.: Radiance caching for global illumination computation on glossy surfaces. Ph.d. thesis, Université de Rennes 1 and Czech Technical University in Prague (2005)
12. Křivánek, J., Gautron, P.: Practical Global Illumination with Irradiance Caching. Synthesis lectures in computer graphics and animation. Morgan & Claypool (2009). DOI 10.2200/S00180ED1V01Y200903CGR010. URL http://dx.doi.org/10.2200/S00180ED1V01Y200903CGR010
13. Lafortune, E.: Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering. Ph.D. thesis, Katholieke Universitiet Leuven, Belgium (1996)
14. Lehtinen, J., Aila, T., Chen, J., Laine, S., Durand, F.: Temporal light field reconstruction for rendering distribution effects. ACM Trans. Graph. **30**(4) (2011)
15. Lehtinen, J., Aila, T., Laine, S., Durand, F.: Reconstructing the indirect light field for global illumination. ACM Trans. Graph. **31**(4) (2012)
16. Powell, M., Swann, J.: Weighted uniform sampling – a Monte Carlo technique for reducing variance. IMA J. Appl. Math. **2**(3), 228–236 (1966)
17. Schwenk, K.: Filtering techniques for low-noise previews of interactive stochastic ray tracing. Ph.d. thesis, Technische Universität Darmstadt (2013)
18. Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. In: Proceedings of the 1968 23rd ACM National Conference, pp. 517–524. ACM (1968)
19. Silverman, B.: Density Estimation for Statistics and Data Analysis. Chapman & Hall/CRC (1986)
20. Spanier, J., Maize, E.: Quasi-random methods for estimating integrals using relatively small samples. SIAM Review **36**(1), 18–44 (1994)
21. Suykens, F., Willems, Y.: Adaptive filtering for progressive Monte Carlo image rendering. In: WSCG 2000 Conference Proceedings (2000)
22. Teschner, M., Heidelberger, B., Müller, M., Pomeranets, D., Gross, M.: Optimized spatial hashing for collision detection of deformable objects. In: Proceedings of VMV'03, Munich, Germany, pp. 47–54 (2003)
23. Wald, I., Kollig, T., Benthin, C., Keller, A., Slusallek, P.: Interactive global illumination using fast ray tracing. In: P. Debevec, S. Gibson (eds.) Rendering Techniques 2002 (Proc. 13th Eurographics Workshop on Rendering), pp. 15–24 (2002)
24. Ward, G., Rubinstein, F., Clear, R.: A ray tracing solution for diffuse interreflection. In: Computer Graphics, pp. 85 – 90 (1988)
25. Wyman, C., Sloan, P., Shirley, P.: Simple analytic approximations to the CIE XYZ color matching functions. Journal of Computer Graphics Techniques (JCGT) **2**, 1–11 (2013). URL http://jcgt.org/published/0002/02/01/