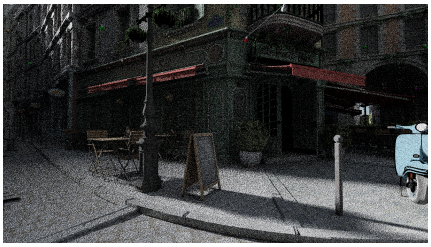
Fast Path Space Filtering by Jittered Spatial Hashing

Nikolaus Binder
NVIDIA

Sascha Fricke
University of Braunschweig

Alexander Keller
NVIDIA



(a) 1 sample/pixel

(b) 1 sample/pixel with path space filtering

(c) reference at 1024 samples/pixel

ABSTRACT

Restricting path tracing to a small number of paths per pixel for performance reasons rarely achieves a satisfactory image quality for scenes of interest. However, path space filtering may dramatically improve the visual quality by sharing information across vertices of paths classified as "nearby". While thus contributions can be filtered in path space and beyond the first intersection, searching "nearby" paths is more expensive than filtering in screen space. We greatly improve over this performance penalty by storing and looking up the required information in a hash map using hash keys constructed from jittered and quantized information, such that only a single query may replace costly neighborhood searches.

CCS CONCEPTS

• **Computing methodologies** → **Ray tracing**;

KEYWORDS

Ray tracing, path space filtering, hashing, realtime.

ACM Reference Format:
Nikolaus Binder, Sascha Fricke, and Alexander Keller. 2018. Fast Path Space Filtering by Jittered Spatial Hashing. In *Proceedings of SIGGRAPH '18 Talks.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3214834.3214841

1 INTRODUCTION

Path space filtering [Keller et al. 2014] averages contributions of light transport paths in path space, which allows for filtering at non-primary intersections and for a more efficient handling of disocclusions during temporal filtering. Querying the contributions in path space so far has been significantly more expensive than filtering the contributions of neighboring pixels in screen space [Sen et al. 2015].

SIGGRAPH '18 Talks, August 12-16, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5809-5/18/08.
https://doi.org/10.1145/3214834.3214841

Neglecting the fact that locations that are close in path space are not necessarily adjacent in screen space enables interactive filtering in screen space [Gautron et al. 2014]. As a consequence, filtering is almost only a good approximation for primary rays. In fact, such filtering algorithms are a variant of a bilateral filtering using path space proximity to determine weights.

Fast filtering is also possible in texture space [Munkberg et al. 2016], which at least requires a bijection between the scene surface and texture space. While this may be tricky already, issues may arise along discontinuities of a parametrization in addition.

Querying hash maps of sufficiently large size often requires only a single memory access, resulting in a constant complexity most of the time. Hence costly neighborhood search can be replaced by accumulating contributions directly at quantized path space descriptors, which is key to the proposed algorithm.

2 ALGORITHM

After a set of light transport paths has been traced, a hash key is constructed for each one selected vertex of each path. This hash key consists of all information required to classify "nearby" vertices. Besides quantized world space position, considering the normal at the vertex helps to avoid smearing across edges. Other possible information includes the incoming direction for non-diffuse materials and the layer identifier for layered materials, for example.

Now contributions from vertices with the same key are atomically added using a hash map. After all contributions have been summed up for every key, averages are computed by dividing by the number of contributions accumulated per key, which again is maintained by an atomic counter.

For an efficient key construction we propose to use a simple and fast hash function on the quantized information, which determines the initial location in the hash map. For verification, we use a second, different, hash function with the same input to avoid storing and comparing keys of a much larger size. Upon a hash collision, we employ linear probing with a very small number of steps to increase the occupancy in the hash map. We observed that for a hash map size equal to the number of rays and practical quantization, linear probing with only a few steps is already sufficient. If a hash collision can still not be resolved, the unfiltered contribution of the path is used.

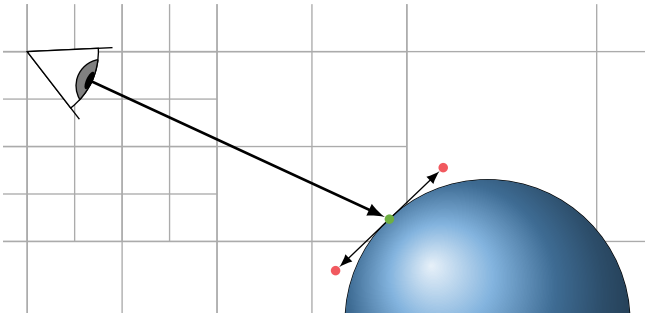


Figure 1: Spatial jittering and subsequent quantization according to level of detail.

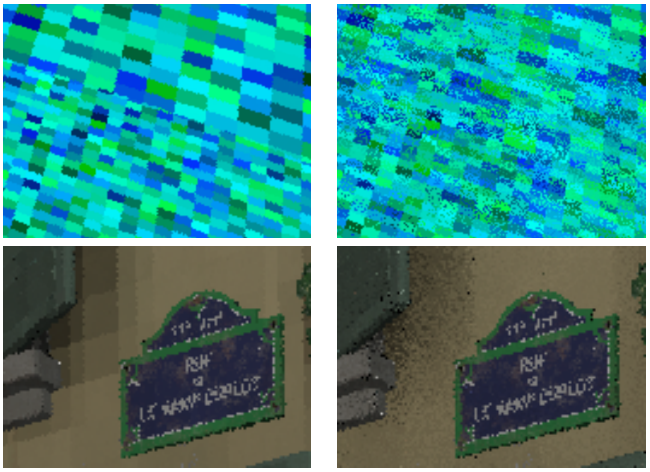


Figure 2: Top row: Spatial jittering hides otherwise visible discontinuities across different levels of detail. Bottom row: At low sampling rates this results in noise instead of quantization artifacts.

2.1 Jittered Spatial Hashing

Quantization artifacts can be hidden in noise by jittering key parameters. This noise is clearly preferable over visible boundaries resulting from quantization, as illustrated and shown in Fig. 2. Jittering depends on the kind of parameter, for example, positions are jittered in the tangent plane of an intersection, see Fig. 1. In fact, jittering realizes the stochastic evaluation of filter kernels.

Filtering can be further improved by introducing level of detail to the quantization. A heuristic as simple as the distance to the eye is sufficient to select a level of detail for filtering primary intersections. More advanced heuristics allow for a better control of the quantization in order to account for ray density and variance, but may require multiple lookups.

2.2 Temporal Path Space Filtering

The straightforward approach to temporal filtering is to accumulate contributions across frames. As long as the scene is static, the averages will converge. For dynamic environments, radical changes even for keys with a high number of old contributions can be controlled by a parameter α: Maintaining two sets of averaged contributions and combining them with an exponential moving average $v = \alpha \cdot v_{\text{old}} + (1-\alpha) \cdot v_{\text{new}}$ is a commonly accepted tradeoff between convergence and temporal adaptivity.

Evicting contributions of keys which have not been queried for a certain period of time is necessary for larger scenes and changing camera. Besides the least recently used (LRU) eviction strategy, more elaborate "health" heuristics based on longer term observations are efficient.

A very simple implementation relies on replacing the most significant bits of the verification hash by a priority composed of for example density and last access time during temporal filtering. Thus the pseudo-randomly hashed least significant bits guarantee missing contributions to be uniformly distributed across the scene, while the most significant bits ensure that contributions are evicted according to priority. As a consequence, collision handling can be realized by a single `atomicMin` operation.

3 RESULTS AND DISCUSSION

Using the proposed method, primary rays for HD resolution (1920× 1080) can be filtered in ∼1 ms on an NVIDIA Titan V GPU and in ∼2 ms on an NVIDIA Titan X GPU. Filtering incoherent rays is penalized by incoherent memory access, which leads to a slowdown of ∼2×.

Filtering, and especially accumulating contributions, is always prone to light and shadow leaking, which is the price we pay for performance. Some artifacts may be ameliorated by employing suitable heuristics.

While filtering contributions in primary intersections with the proposed algorithm is quite fast, it offers only few advantages over filtering in screen space. However, filtering in non-primary intersections opens up new possibilities: One key advantage for shifting filtering "back" along the path is that averaging over filtered contributions often hides visible artifacts of the underlying approximation. Note that our path space filtering algorithm is the only efficient fallback when screen space filtering fails or is not available, for example, for specular or transparent objects.

Filtering on non-diffuse surfaces requires additional parameters for the key and heuristics such as increasing the quantization in areas with non-diffuse materials to minimize the visible artifacts.

While path space filtering dramatically reduces the noise at low sampling rates (see the teaser image series), some noise is added back by spatial jittering. Temporal anti-aliasing and complimentary noise filters in screen space are appropriate to further reduce noise.

REFERENCES

P. Gautron, M. Droske, C. Wächter, L. Kettner, A. Keller, N. Binder, and K. Dahm. 2014. Path Space Similarity Determined by Fourier Histogram Descriptors. In *ACM SIGGRAPH 2014 Talks (SIGGRAPH '14)*. ACM, New York, NY, USA, Article 39, 1 pages. https://doi.org/10.1145/2614106.2614117

A. Keller, K. Dahm, and N. Binder. 2014. Path Space Filtering. In *ACM SIGGRAPH 2014 Talks (SIGGRAPH '14)*. ACM, New York, NY, USA, Article 68, 1 pages. https://doi.org/10.1145/2614106.2614149

J. Munkberg, J. Hasselgren, P. Clarberg, M. Andersson, and T. Akenine-Möller. 2016. Texture Space Caching and Reconstruction for Ray Tracing. *ACM Trans. Graph.* 35, 6, Article 249 (Nov. 2016), 13 pages. https://doi.org/10.1145/2980179.2982407

P. Sen, M. Zwicker, F. Rousselle, S.-E.Yoon, and N. Kalantari. 2015. Denoising Your Monte Carlo Renders: Recent Advances in Image-space Adaptive Sampling and Reconstruction. In *ACM SIGGRAPH 2015 Courses (SIGGRAPH '15)*. ACM, New York, NY, USA, Article 11, 255 pages. https://doi.org/10.1145/2776880.2792740