Vision, Modeling, and Visualization (2017)
M. Hullin, R. Klein, T. Schultz, A. Yao (Eds.)

Pixel Cache Light Tracing

J. Jendersie K. Rohmer F. Brüll T. Grosch

Institut für Informatik, TU Clausthal, Germany

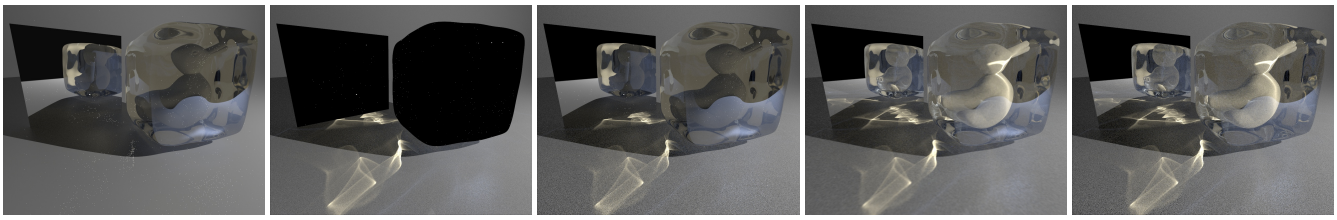


Figure 1: *Equal time comparison (1min) of different sampling strategies. F.l.t.r.: Path Tracing (PT), Light Tracing (LT), Bidirectional Path Tracing (BPT), Stochastic Progressive Photon Mapping (SPPM) and Pixel Cache Light Tracing (PCLT). PT fails to produce caustics, LT has the same problem with projection to the camera and BPT has still no option for Specular-Diffuse-Specular paths. Only SPPM shows all paths. PCLT produces sharper images than SPPM but misses some multi-specular reflections (blueish spot at the bottom of the cube).*

Abstract

In this paper, we introduce Pixel Cache Light Tracing, which is a new low-noise combination of eye-path and light-path tracing. In the first pass, eye-path vertices are distributed from the observer and stored in a hit point map analogous to progressive photon mapping. In the second pass, photons are traced from the light source and projected to the image as well as gathered by the hit point map.

We combine the paths from both sampling strategies in a deterministic way without multiple importance sampling, such that the final result is consistent and free from firefly artifacts. In many practical cases, this combination leads to sharper caustics and reduced noise when compared to alternative techniques at equal time.

Further, the simplicity of the path combination strategy is predestined for GPU-based implementations and requires less memory than a comparable photon mapping implementation. In addition, we provide a fast, parallel and lean hash map implementation for both photon and hit point queries.

CCS Concepts

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

1. Introduction

Light transport simulation is often done using Monte Carlo-based integration strategies. These methods are able to produce unbiased, physically plausible solutions of the rendering equation, but suffer from high-frequency noise. In contrast, photon mapping [Jen96] introduces a systematic error (bias), due to blurring light, which on the other hand decreases the noise. Photon mapping can be adjusted to produce consistent results in theory [HOJ08,KZ11] which means that the systematic error decreases over time. In practice, the initial bias often remains for an impractical long time.

PCLT combines SPPM and light tracing in a simple form, sacrificing some benefits from *Multiple Importance Sampling* (MIS). The emphasis is on unidirectional light tracing where possible, since it yields desirable results in many applications. The simplicity of the design allows fast GPU implementations compared to MIS-based techniques. It requires less complex, time consuming operations because of the few path crossover points and no rejection-based estimates like [QSH*15].

In the first pass we store eye-path vertices (hit points), having at least one glossy bounce, into a search data structure. While tracing a light path we search for close hit points in the map and estimate the radiance contribution for a pixel directly, without storing photons. The idea of storing hit points instead of photons is not new and was used before by [HHS05], PPM [HOJ08] and SPPM [HJ09]. It does not change the behavior opposed to a classic photon mapper

DOI: 10.2312/vmv.20171269

but requires less memory if the number of pixels (with glossy reflections) is smaller than the number of photons. For all other paths, including direct light, we prefer general unbiased light tracing with projection. This reduces the visible bias significantly and still produces results without fireflies.

Additionally, we introduce a lean and fast hash grid for hit points or photons which is useful for many mapping-based approaches.

2. Related Work

Path and Light Tracing are the two elementary path creation strategies. Path tracing was introduced in combination with the rendering equation in [Kaj86]. Paths are traced beginning at the observer. At each path vertex, a new random direction is sampled to provide a numerical estimate of the irradiance. Additionally, next event estimations can be performed at each path vertex in which the direct illumination is calculated for some chosen light source. Light tracing [Arv86, DLW93] starts paths at light sources and proceeds in the same Monte-Carlo manner like path tracing. A next event estimation in light tracing connects the light path vertex with the observer through projection [DLW93]. Another path tracing approach from Henrich et al. [HBGM11] uses back projection of eye-path vertices to generate more paths. This is not equivalent to projecting light-path vertices, as no bi-directionality is introduced.

Bi-directional Path Tracing combines the two elementary approaches [LW93, VG95]. It connects light-path vertices with eye-path vertices and weights their contribution based on heuristics (multiple importance sampling). If both elementary strategies create certain paths with a low probability (e.g. SDS[†]), the bidirectional path tracer will still produce a high variance output.

An entirely different approach which explicitly searches in path space is *Metropolis Light Transport* (MLT) [VG97, Vea97]. Here, the search is performed by mutations of initial paths. It is inherently difficult to implement and, depending on the quality of path mutation strategies, it may produce poor results often containing structured noise.

Photon Mapping connects light-paths with eye-paths in a biased way by merging close vertices [Jen96]. Therefore, photons (light-path vertices) are stored in a search data structure. Later, a density estimate is performed at the first diffuse eye-path vertex. There are numerous strategies considering different initial eye-paths before performing the density estimate [Chr99, Jen01, HHS05, QSH*15]. We reverse this idea by storing hit points and switching the two passes like [HHS05]. However, in our approach storing hit points decreases the memory requirements of the search data structure which is the opposite of the final gathering strategy in [HHS05].

The first consistent photon mapping is using a progressive decrease in the query radius such that the merging bias converges to zero over time [HOJ08] (PPM). While the original radius estimate of Hachisuka et al. guaranteed a constant amount of photons in the query area, this guarantee is not necessary. Knaus and Zwicker [KZ11] proved the different, simpler radius update

$$r_{i+1} = r_i \sqrt{\frac{i+\alpha}{i+1}} \tag{1}$$

to be consistent. The user parameter α steers the tradeoff between a late visible bias and early high-frequency noise. Values in the range [1/3, 2/3] produce the best results. We use this strategy in our consistent scenarios.

Like our approach, PPM stores hit points in a first pass and then distributes photons into the hit points until convergence. *Stochastic Progressive Photon Mapping* (SPPM) [HJ09] allows a wider range of effects like anti-aliasing and depth-of-field by redistributing hit points after each photon iteration. Our method differs in the point that we use light path projection instead of merging for many paths.

Other Sampling Strategies combine Photon Mapping with Bi-directional Path Tracing to reproduce even more complex light paths. Two equivalent methods, *Vertex Connection and Merging* (VCM) [GKDS12] and *Unified Path Sampling* (UPS) [HPJ12], were found independently at the same time. However, VCM/UPS and SPPM do not recover from early iterations' artifacts in practice. *Unbiased Photon Gathering* (UPG) [QSH*15] improves upon the gathering event by computing probabilities more carefully. To the best of our knowledge this is the most qualitative general GI method, but it requires a variable count of trial rays per connection event, which is not suitable for GPU implementations.

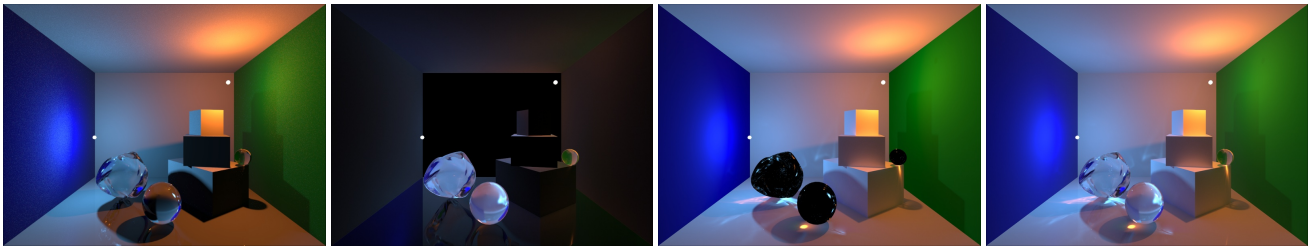
Guidance can be used to enhance any of the previous sampling methods. The idea is to improve the importance sampling while tracing paths by using the adjoint information. I.e. eye-paths are distributed under a light field while light-paths are modified by an importance field. The first approach in that direction is the predecessor of photon mapping by Jensen [Jen95]. Guidance methods were improved over the years, ranging from [HP02] up to [VKŠ*14] and [HEV*16] where Gaussian mixture models are used to model the light/importance fields. Another variance reduction technique, which can be combined with all sampling strategies, is path splitting and termination based on expected path contribution [VK16].

3. Pixel Cache Light Tracing

The idea behind pixel cache light tracing is to combine several low-variance sampling techniques without multiple importance sampling. Whitted style ray tracing [Whi79] (Figure 2a) is one of those techniques. It stops the tracing at the first diffuse bounce and computes the direct illumination at that point. All missing global illumination effects, like indirect light and caustics, are included in light tracing with high quality (Figure 2c). However, light tracing cannot cover specular reflections towards the eye on highly glossy surfaces (e.g. mirrors and glass) without an extremely high variance.

The PCLT algorithm is a two pass algorithm which uses progressive photon gathering (Equation (1)) for Whitted style paths and light tracing otherwise. In the first pass, the diffuse endpoints of traced eye-paths are stored in a hash grid (Figure 3 paths 4 and 5), excluding those which are directly visible (path 6). In the second pass, a light tracer is executed with two different operations at each vertex. The first operation is a radiance estimate for close eye-path

[†] Heckbert's notation [Hec90]: Paths can be described by regular expressions with one symbol per vertex, where L=Light, E=Eye, D=Diffuse reflection and S=Specular reflection.



(a) *Whitted ray tracing* **(b)** *Pixel cache* **(c)** *Light tracing* **(d)** *Full solution (b)+(c)*

Figure 2: *Contributions of different tracing strategies. Note that the materials have a Fresnel specular component with varying roughness, which produce the reflections visible in image (b) and (d).*

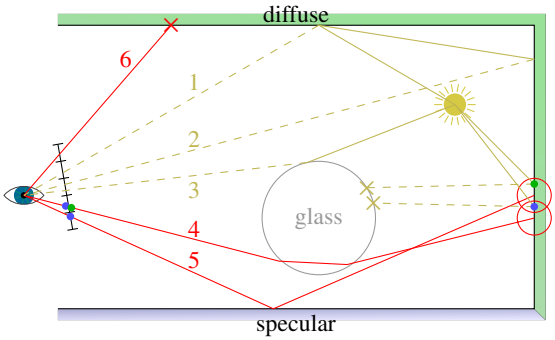


Figure 3: *Paths used by PCLT: most paths are created by light tracing with a connection to the eye (dashed: 1,2,3). Only eye-paths with at least one specular bounce (4,5) are stored in the pixel cache. They contribute through gathering if hit by a photon directly (colored dots).*

vertices using the hash grid. The second operation is a projection to the camera (next event estimation; Figure 3 paths 1,2 and 3). The radiance estimate is skipped if there is no close pixel cache entry and the projected contribution is skipped if the point is not visible from the eye. The contributions of the two operations are shown in Figure 2b and 2c.

The pixel cache is a data structure that contains at most one hit point for each pixel. In each iteration, we trace eye-paths until the importance sampling yields a diffuse (Lambertian) scattering event and store this last path vertex. This allows materials to have combined specular and diffuse layers and produces paths of the form ES*D. Since light tracing often yields superior results compared to gathering we do not store paths of the form ED (direct visible diffuse) and use projection in that case. A radiance estimate with the stored hit points only needs to compute diffuse lighting. If the material has a specular layer the contribution is handled by other eye-paths through importance sampling. Opposed to the adjoint photon maps, this cache depends on the number of eye-paths. We observed that increasing the number of light-paths (photon count) is more beneficial for the image quality than increasing the number of eye-paths (for an example see Figure 7). Thus, using a pixel cache reduces the memory requirements and build times for the search data structures.

4. GPU Hash Grid

To query hit points in the surrounding of light path vertices we need a fast search structure. On CPU, a common choice is a kd-tree. On GPU, hash grids are usually faster and simpler to implement [HJ10]. In a hash grid, positions are discretized to grid coordinates. The content of the cells is then stored into a hash map. To find all particles in the vicinity of a query point, only a few cells must be iterated. If the grid spacing is at least twice as large as the query radius it is sufficient to look at the $2^3 = 8$ closest cells.

The most difficult part of the hash grid is the underlying parallel hash map. In [ASA*09] a complex hash map hybrid using cuckoo hashing is presented. However, according to Hachisuka and Jensen [HJ10] stochastic hash maps are more efficient for photon mapping. In a stochastic hash map collisions are resolved by discarding data with a probability inverse to the number of collisions at this location. We implemented a different hash map using chaining based on [Alc11]. A comparison between stochastic hash mapping and our version is shown in Figure 8.

Our hash map consists of two data structures. One buffer with key-link pairs and one buffer containing the data plus a linked list index serving as next pointer for other data entries. The algorithm for adding a data value to the hash grid is shown in Listing 1.

Listing 1: *Chained hash map implementation*

```
insertToHashGrid(vec3 position, Data data)
  # store data (will be referenced from HM or
  # linked list later).
  dataIdx = atomicCounterIncrement(g_counter)
  g_dataBuffer[dataIdx] = data;

  hash = worldPosToGridCellHash(position)
  idx = hash % HASH_MAP_SIZE
  while true:
    # Try to place the current cell in the map.
    key = atomicCompSwap(g_hashMap[idx].x,~0,hash)
    # If empty, or the same cell add to list
    if key == hash || key == ~0:
      g_dataBuffer[dataIdx].next =
        atomicExchange(g_hashMap[idx].y, dataIdx)
      break
    end
    # Collision with different cell -> probing
    idx = (idx + <probeDistance>) % HASH_MAP_SIZE
  end
end
```

The first atomic operation implements an append buffer. This can be removed if each thread gets a memory range for its output. In our experiments this did not change the performance.

The other two atomic operations handle the cell-insertion to the map and the data insertion to the linked list of that cell. If two cells have an identical hash, their particles will fall into the same list. This increases the runtime of the search in theory - in practice we did not observe any penalty on runtime.

For collision handing between cell with a different hash, we tried linear and quadratic probing and found quadratic probing to be much faster which is in accordance to [Alc11].

4.1. Pixel Cache Data

We use the above hash map to store the following entry for the last vertex on an eye-path:

```
uint32 nextEntry              # hash map chain
vec3 eyePathThroughputWeight
vec3 hitPosition              # vertex position
uint16[2] pixelCoordinate     # pixel origin
```

Here, `eyePathThroughputWeight` is the cumulated path weight $\prod f(\mathbf{v},\mathbf{i})/p_{\text{sampling}}$ where f is the BSDF at the hit points and p_{sampling} the probability distribution function used for sampling. Note that neither BRDF nor incident direction information are stored. This is a consequence of our path combination strategy. Using the global radius update (Equation (1)) it is not necessary to store photon statistics per hit point.

We use the same hash map with a similar data structure in our SPPM implementation.

5. Light Tracing Projection

A projection in our context is equal to a next event estimation from the light-path to the sensor. While many of the bi-directional algorithms include this type of path, we have not found the required formula in the current literature. In Section 5.1 a formula which computes the light transport from an incident photon on a surface to a pinhole camera is derived. It is then extended to larger filter kernels in Section 5.2.

Before projection, the pixel coordinate of the path vertex is computed and an occlusion ray is cast. Since this is one of the most time consuming steps in our GPU PCLT implementation we propose a biased optimization in Section 5.3.

5.1. Visible Radiance from the Sensor

First, details on nomenclature can be found in Figure 4. All quantities with an index s are measured at the surface and all with an index p at the pixel (or pinhole respectively). Especially, these are: θ_p the angle between the view direction and the direction through the pixel center, ω_p the solid angle of the pixel and θ_s the angle between surface normal and the connection to the pinhole.

Beginning with the definition of the BSDF as the ratio between

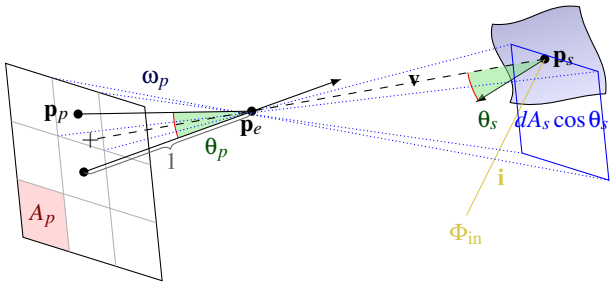


Figure 4: *Projection to a pin-hole camera with focal length 1.0 located at $\mathbf{p}_e$. All pixels on the virtual plane have the same area A_p. The projection of $\mathbf{p}_s$ falls on pixel $\mathbf{p}_p$ for which we know the solid angle ω_p. Φ_{in} is the flux of a photon coming from direction $\mathbf{i}$.*

excident radiance L and incident irradiance E we can derive the outgoing flux Φ_s into observer direction $\mathbf{v}$:

$$f(\mathbf{i},\mathbf{v}) = \frac{dL(\mathbf{v})}{dE(\mathbf{i})} = \frac{d^2\Phi_s}{d\Phi_{\text{in}} d\omega_s \cos\theta_s}$$
$$\Rightarrow d^2\Phi_s = f(\mathbf{i},\mathbf{v}) d\Phi_{\text{in}} d\omega_s \cos\theta_s, \tag{2}$$

where $d\omega_s$ is the solid angle from the surface to the observer. For the pinhole this quantity is aligned with the direction $\mathbf{v}$ and goes to zero (and is therefore not existent/included in the Figure). However, this is no problem as it cancels out in the ongoing derivation.

Interpreting the projection as a single path tracing sample we want to calculate the radiance at the surface

$$L_s = \frac{d^2\Phi_s}{d\omega_s dA_s \cos\theta_s}.$$

Inserting Equation 2 gives

$$L_s = \frac{d\Phi_{\text{in}} f(\mathbf{i},\mathbf{v}) \cos\theta_s}{dA_s \cos\theta_s} \tag{3}$$

with $d\omega_s$ already canceled out. Now we can approximate the visible area $dA_s \cos\theta_s$ from the pixel, dependent on the pixel's solid angle ω_p, with $\omega_p \|\mathbf{p}_e - \mathbf{p}_s\|^2$ and insert that into equation 3:

$$L_s \approx \frac{d\Phi_{\text{in}} f(\mathbf{i},\mathbf{v}) \cos\theta_s}{\omega_p \|\mathbf{p}_e - \mathbf{p}_s\|^2}.$$

To express the solid angle of a pixel ω_p, we describe the pixel's solid angle with respect to a virtual plane in a distance of 1.0 to the pinhole camera which is also shown in Figure 4. Applying simple trigonometry we get

$$\omega_p \approx \frac{A_p \cos\theta_p}{\|\mathbf{p}_e - \mathbf{p}_p\|^2} = A_p \cos^3\theta_p.$$

Therefore, the final result describing the influence of a photon to the radiance seen in the pixel is

$$L_s \approx \frac{\Phi_{\text{in}} f(\mathbf{i},\mathbf{v}) \cos\theta_s}{A_p \cos^3\theta_p \|\mathbf{p}_e - \mathbf{p}_s\|^2}. \tag{4}$$

5.2. Image Reconstruction Filters in Projection

Equation (4) is implementing a box filter per pixel. Other filters, using a custom weight $w(r)$ with r being the distance from the pixel

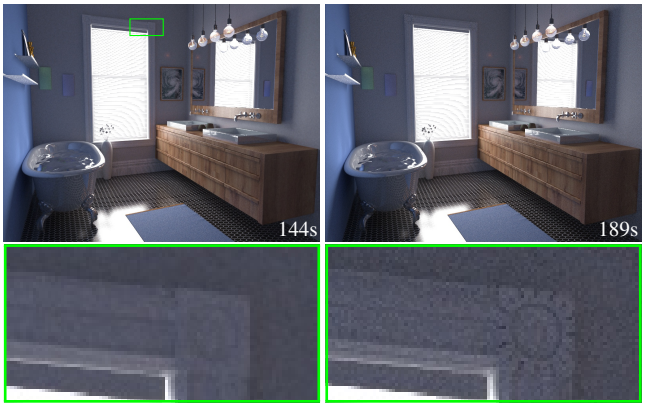


Figure 5: *Equal path count comparison between SPPM (left) and PCLT (right) (5000 iterations with an equal number of eye-paths and light-paths). Scene from PBRT book [PJH17].*

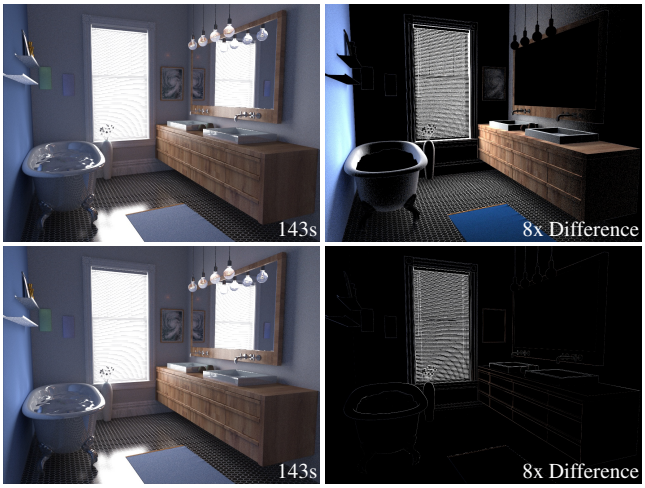


Figure 6: *Performance and error of z-buffer (first row) and plane (second row) based visibility test compared to Figure 5.*

center to the sample, can be integrated into projection by looping over a small neighborhood of pixels. In this case it is wrong to compute the solid angle of a pixel ω_p. Instead, we need the solid angle of the filter $\omega_w \approx A_w \cos^3 \theta_w$. Here, θ_w is the angle between view direction and sample direction (instead of the pixel center) and A_w the filter area in the image plane (similar to A_p).

5.3. Faster Occlusion Tests

It is possible to speed up the visibility test by replacing the ray cast with a z-buffer comparison. Unfortunately, this also introduces a bias which cannot be removed [HBGM11]. We used a stochastic z-buffer which is generated by the first pass together with the pixel cache. Instead of the pixel center depth like in [HBGM11], a sampled depth anywhere in the pixel is produced. A projection is accepted if the point is closer to the camera than the current depth value plus a small offset. This approach yields a systematic energy loss on planar surfaces as shown in Figure 6 (first row).

In a second approach we stored position and geometry normal at the first hit point and performed half space tests with this plane. Projections are accepted if the distance to the plane is greater than $-z \cdot \varepsilon$, where the view depth z accounts for the greater pixel footprints in the distance. The parameter ε depends on resolution and field-of-view. In our examples we use $\varepsilon = 5 \cdot 10^{-4}$. While this optimization is still biased, the visible error is very small, as the second row in Figure 6 shows.

6. Results

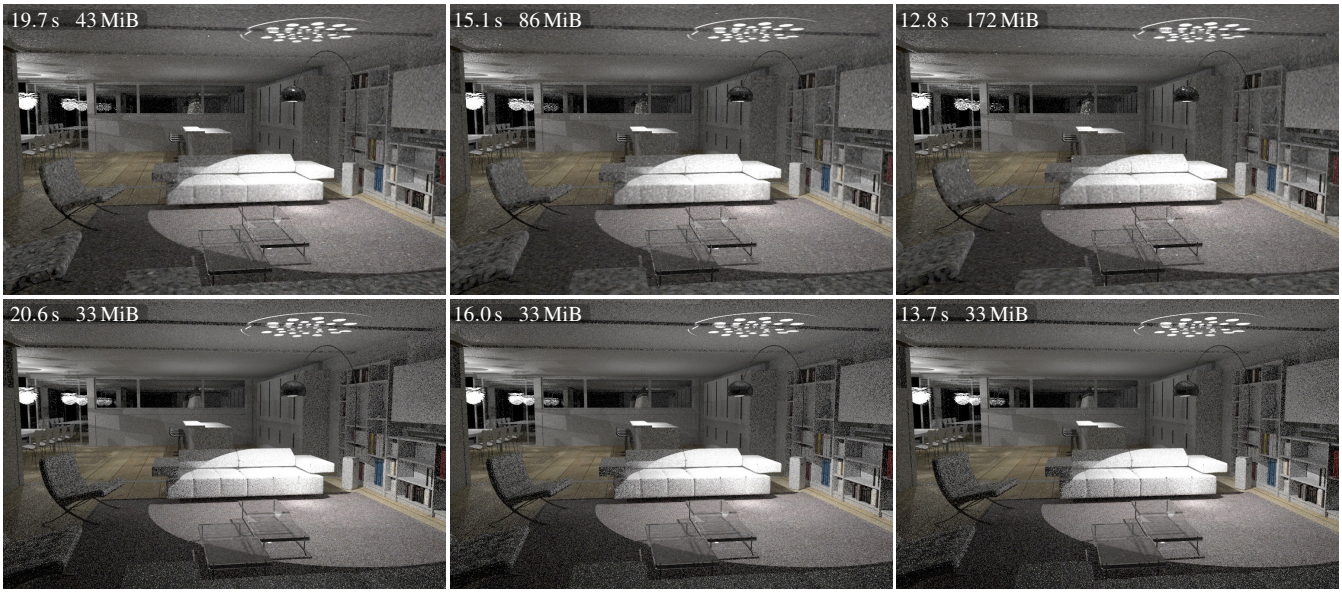
A first comparison of our method can be found in the teaser (Figure 1) which shows a diffuse BunnyDuck in a glass container and a mirror. PCLT reaches the quality of SPPM but with sharper caustics. While all methods should converge to the same output, LT, PT and BPT fail to produce many types of paths. These have very small probabilities and are practically impossible. For the same reason PCLT misses some multi-specular reflections. In the scene, the floor has a rough specular layer. A resulting reflection of the light source is not visible, because this pure specular path is not reproduced by LT in practice. SPPM produces correct results, except the missing direct light source reflections on the mirror and the cube (bright dots in all other images). In a comparison between SPPM and our PCLT, PCLT generates the sharper image.

The equal iteration comparison in Figure 5 compares only SPPM and PCLT because the other methods do not achieve the full path spectrum. Again, PCLT produces sharper results on diffuse and mixed material surfaces. However, PCLT still uses gathering for reflective and refractive surfaces. Therefore, the images in the mirror are very similar in both pictures. Since many scenes mainly consist of diffuse and mixed materials, our algorithm can improve their rendering results.

Comparing the timings in Figure 5 reveals that our algorithm is 31% slower than SPPM. The reason is the additional visibility test during projection. Therefore, we rendered the same image again using the z-buffer and plane-based tests which yields the results in Figure 6. With this optimization, our algorithm has the same speed as SPPM for the bathroom scene. However, using a simple z-buffer comparison leads to a systematic loss of energy on planar surfaces, whereas plane-based tests produce small errors along edges. Nevertheless, the error from plane-based test is hardly visible and this optimization should be used in practice.

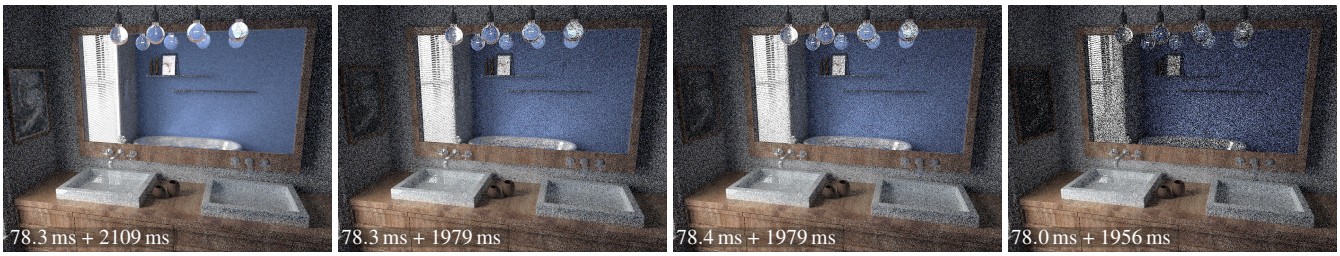
Figure 7 demonstrates that using more light-paths than eye-paths is more performant while yielding very similar results. Both, PCLT and SPPM, are significantly faster when using more photons per iteration. For PCLT we used the plane-based visibility test to achieve comparable timings. It also shows that memory costs are increasing for SPPM when distributing more photons while the costs for PCLT remain constant. However, the memory for the pixel cache increases with image resolution. The map would need 82 MiB in case of FullHD resolution. To compensate this, pixels could be stored stochastically once the buffer is full.

The effect and performance of stochastic hash mapping is visualized in Figure 8. The first thing to be noticed is that photons



200 iterations with 2^{18} photons each | 100 iterations with 2^{19} photons each | 50 iterations with 2^{20} photons each

Figure 7: *Equal light path count comparison. Top row: SPPM, bottom row: PCLT. Using more photons per iteration improves performance for both renderers, while having almost the same quality. Memory requirements increase for SPPM only. Scene from PBRT book [PJH17].*



Full hash map 33 MiB | Stochastic hash map 33 MiB | Stochastic hash map 10 MiB | Stochastic hash map 1 MiB

Figure 8: *Our hash map compared to stochastic hash mapping. All images are rendered with 8 iterations. The timings are eye-pass and light-pass respectively. The main difference in timing results from the number of radiance estimates in the light-pass.*

do not spread over multiple pixel anymore. This is because in the mirror neighboring pixels have path-endpoints which fall into the same hash grid cell. Therefore, at most one pixel in each cell-neighborhood will survive the stochastic collision handling. Further collisions are no issue before forcing the map memory down too much (see 1 MiB case). With respect to timings the two maps are very similar. The only reason why the leftmost rendering, using our full map, is slower, is that more pixels are found in the radiance estimate. Hence, more projections are performed. For reference, if nothing is stored or searched in the pixel cache the timings are 77.4 ms + 1882 ms. I.e. building and searching particles in the hash map has negligible costs compared to the tracing.

Summarizing the results from Figures 1, 5 and 7, our method always produces the higher quality results in equal time comparisons. Dependent on the z-buffer optimization this ratio can be improved even more, losing the unbiasedness of the projection.

Eventually, PCLT shares the strengths and weaknesses of all light tracing-based methods which includes all photon mapping techniques. In Figure 9 we compare the best non-projection/gathering-based techniques to our algorithm for two extreme scenarios. Scenario one contains many (69) light sources. In such a case, next event estimation in PT or BPT often fails due to occlusion. Contrary, an LT-based method distributes the photons close to the light sources which produces much better results. For the same reason only few photons of far distant light sources reach visible regions which is shown in the second scenario. However, in both scenarios the PCLT is faster than the compared methods for the 1000 iterations. In an equal time comparison it would be even better in the first case and not as bad in the second one.

A pattern in the noise can be observed in the front region of the PCLT day light image, which we identified as a floating point issue. The generated directions at the light source are not perfectly uniform which is visible on this distance.

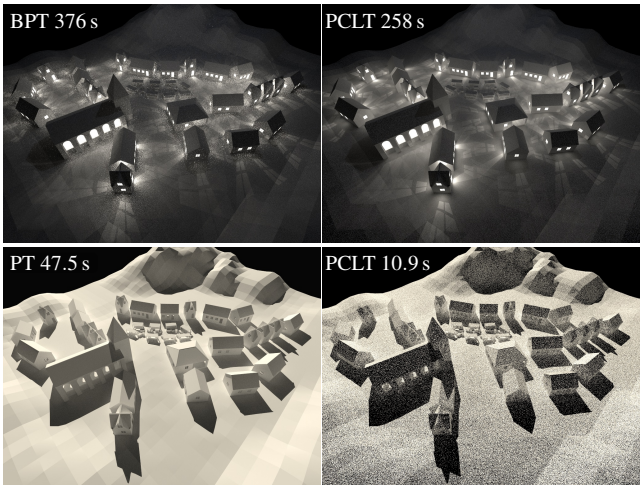


Figure 9: *Strengths and weaknesses of light tracing-based renderers. The upper row shows a scenario with 69 indoor light sources, which is bad for PT based methods because many of the next event connections are occluded. On the other hand, LT has problems with very distant light sources as shown the second row. All images are computed with 1000 iterations.*

7. Conclusions and Future Work

We presented PCLT, a simple light and eye path combination strategy which is well suited for GPU implementations. In general it is very similar to SPPM, but produces sharper images and requires less memory for high photon throughputs. Unlike more complex methods (VCM/UPS, UPG), it does not require multiple importance sampling and difficult probability estimations. High variance noise only remains for pure specular paths.

However, our method shares many of the problems of the other methods. For a very distant light source only a few rays reach the view frustum, like in all photon mapping or light tracing approaches. Further, situations, in which light source and visible parts are only connected by a small gap, produce only a few contributing paths. Both cases are handled similarly by different methods and could be improved by guidance. Here, it might be possible to find specialized solutions for PCLT. Another problem are the high variance pure specular paths through light tracing. In some cases those paths can be produced better by path tracing (see teaser image) which could be incorporated using MIS. However, this would also require tracing more paths/doing connection tests, resulting in an strongly increased complexity.

Additionally, we proposed a lean parallel hash grid implementation which shows the same performance as a stochastic hash map but does not lose information. This hash map can also be used in other GPU-based implementations to reduce the variance from hash collisions (SPPM, VCM/UPS, UPG).

8. Acknowledgements

This work is partially supported by the German Research Foundation (DFG), Grant Nr. GR 3833/3-1.

References

[Alc11] ALCANTARA D. A. F.: *Efficient Hash Tables on the GPU*. PhD thesis, University of California at Davis, 2011. URL: http://idav.ucdavis.edu/~dfalcant/research.php. 3, 4

[Arv86] ARVO J.: Backward Ray Tracing. In *Computer Graphics (Proc. SIGGRAPH)* (1986), pp. 259–263. 2

[ASA*09] ALCANTARA D. A., SHARF A., ABBASINEJAD F., SENGUPTA S., MITZENMACHER M., OWENS J. D., AMENTA N.: Real-time Parallel Hashing on the GPU. *ACM Transactions on Graphics (TOG) 28*, 5 (Dec. 2009), 154:1–154:9. URL: http://doi.acm.org/10.1145/1618452.1618500. 3

[Chr99] CHRISTENSEN P. H.: Faster Photon Map Global Illumination. *Journal of Graphics Tools (JGT) 4*, 3 (1999), 1–10. URL: http://dx.doi.org/10.1080/10867651.1999.10487505. 2

[DLW93] DUTRÉ P., LAFORTUNE E. P., WILLEMS Y. D.: Monte Carlo Light Tracing with Direct Computation of Pixel Intensities. In *Proc. of Computational Graphics and Visualisation Techniques* (Dec. 1993), pp. 128–137. URL: https://lirias.kuleuven.be/handle/123456789/132745. 2

[GKDS12] GEORGIEV I., KŘIVÁNEK J., DAVIDOVIČ T., SLUSALLEK P.: Light Transport Simulation with Vertex Connection and Merging. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 192:1–192:10. URL: http://doi.acm.org/10.1145/2366145.2366211. 2

[HBGM11] HENRICH N., BAERZ J., GROSCH T., MÜLLER S.: Accelerating Path Tracing by Eye-Path Reprojection. In *International Congress on Graphics and Virtual Reality (GRVR)* (2011). URL: http://www.rendering.ovgu.de/rendering_media/downloads/publications. 2, 5

[Hec90] HECKBERT P. S.: Adaptive Radiosity Textures for Bidirectional Ray Tracing. *Computer Graphics (Proc. SIGGRAPH) 24*, 4 (Sept. 1990), 145–154. URL: http://doi.acm.org/10.1145/97880.97895. 2

[HEV*16] HERHOLZ S., ELEK O., VORBA J., LENSCH H., KŘIVÁNEK J.: Product Importance Sampling for Light Transport Path Guiding. *Computer Graphics Forum (CGF) 35*, 4 (2016), 67–77. URL: http://dx.doi.org/10.1111/cgf.12950. 2

[HHS05] HAVRAN V., HERZOG R., SEIDEL H.-P.: Fast Final Gathering via Reverse Photon Mapping. *Computer Graphics Forum (CGF) 24*, 3 (2005), 323–332. URL: http://dx.doi.org/10.1111/j.1467-8659.2005.00857.x. 1, 2

[HJ09] HACHISUKA T., JENSEN H. W.: Stochastic Progressive Photon Mapping. *ACM Transactions on Graphics (TOG) 28*, 5 (Dec. 2009), 141:1–141:8. URL: http://doi.acm.org/10.1145/1618452.1618487. 1, 2

[HJ10] HACHISUKA T., JENSEN H. W.: Parallel Progressive Photon Mapping on GPUs. In *ACM SIGGRAPH Asia Sketches* (2010), ACM, p. 54. URL: http://doi.acm.org/10.1145/1899950.1900004. 3

[HOJ08] HACHISUKA T., OGAKI S., JENSEN H. W.: Progressive Photon Mapping. *ACM Transactions on Graphics (TOG) 27 number 5* (2008), 130. URL: http://doi.acm.org/10.1145/1409060.1409083. 1, 2

[HP02] HEY H., PURGATHOFER W.: Importance Sampling with Hemispherical Particle Footprints. In *Proc. of Spring Conference on Computer Graphics* (2002), SCCG '02, ACM, pp. 107–114. URL: http://doi.acm.org/10.1145/584458.584476. 2

[HPJ12] HACHISUKA T., PANTALEONI J., JENSEN H. W.: A Path Space Extension for Robust Light Transport Simulation. *ACM Transactions on Graphics (TOG) 31*, 6 (Nov. 2012), 191:1–191:10. URL: http://doi.acm.org/10.1145/2366145.2366210. 2

[Jen95] JENSEN H. W.: Importance Driven Path Tracing Using the Photon Map. In *Proc. of Eurographics Workshop on Rendering (EGWR)* (1995), EGWR, Springer, pp. 326–335. URL: http://dx.doi.org/10.1007/978-3-7091-9430-0_31. 2

[Jen96] JENSEN H. W.: Global Illumination using Photon Maps. In *Proc. of Eurographics Workshop on Rendering (EGWR)* (1996), EGWR, Springer, pp. 21–30. URL: http://dl.acm.org/citation.cfm?id=275458.275461. 1, 2

[Jen01] JENSEN H. W.: *Realistic Image Synthesis Using Photon Mapping*, vol. 364. Ak Peters Natick, 2001. URL: http://graphics.ucsd.edu/~henrik/papers/book/. 2

[Kaj86] KAJIYA J. T.: The Rendering Equation. In *Computer Graphics (Proc. SIGGRAPH)* (1986), SIGGRAPH, ACM, pp. 143–150. URL: http://doi.acm.org/10.1145/15886.15902. 2

[KZ11] KNAUS C., ZWICKER M.: Progressive Photon Mapping: A Probabilistic Approach. *ACM Transactions on Graphics (TOG) 30*, 3 (2011), 25. URL: http://doi.acm.org/10.1145/1966394.1966404. 1, 2

[LW93] LAFORTUNE E. P., WILLEMS Y. D.: Bi-Directional Path Tracing. In *Proc. of Conference on Computational Graphics and Visualization Techniques* (1993), pp. 145–153. URL: https://lirias.kuleuven.be/handle/123456789/132773. 2

[PJH17] PHARR M., JAKOB W., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation*, 3 ed. Morgan Kaufmann, 2017. URL: http://www.pbrt.org/. 5, 6

[QSH*15] QIN H., SUN X., HOU Q., GUO B., ZHOU K.: Unbiased Photon Gathering for Light Transport Simulation. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 208:1–208:14. URL: http://doi.acm.org/10.1145/2816795.2818119. 1, 2

[Vea97] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1997. URL: http://graphics.stanford.edu/papers/veach_thesis/. 2

[VG95] VEACH E., GUIBAS L. J.: Bidirectional Estimators for Light Transport. In *Photorealistic Rendering Techniques*. Springer Berlin Heidelberg, 1995, pp. 145–167. URL: http://dx.doi.org/10.1007/978-3-642-87825-1_11. 2

[VG97] VEACH E., GUIBAS L. J.: Metropolis Light Transport. In *Proceedings of SIGGRAPH '97* (1997), SIGGRAPH, pp. 65–76. URL: http://dx.doi.org/10.1145/258734.258775. 2

[VK16] VORBA J., KŘIVÁNEK J.: Adjoint-Driven Russian Roulette and Splitting in Light Transport Simulation. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 1–11. URL: http://doi.acm.org/10.1145/2897824.2925912. 2

[VKŠ*14] VORBA J., KARLÍK O., ŠIK M., RITSCHEL T., KŘIVÁNEK J.: On-line Learning of Parametric Mixture Models for Light Transport Simulation. *ACM Transactions on Graphics (TOG) 33*, 4 (2014). URL: http://cgg.mff.cuni.cz/~jaroslav/papers/2014-onlineis/. 2

[Whi79] WHITTED T.: An Improved Illumination Model for Shaded Display. *Computer Graphics (Proc. SIGGRAPH) 13*, 2 (Aug. 1979), 14–. URL: http://doi.acm.org/10.1145/965103.807419. 2