Photon Splatting Using a View-Sample Cluster Hierarchy

P. Moreau[1] E. Sintorn[2] V. Kämpe[2] U. Assarsson[2]
M. Doggett[1]

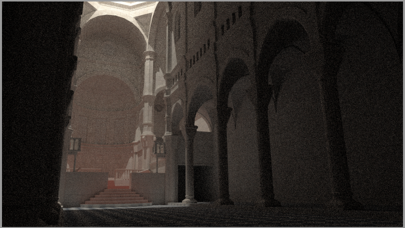
[1]Lund University, Sweden
[2]Chalmers University of Technology, Sweden

High Performance Graphics 2016

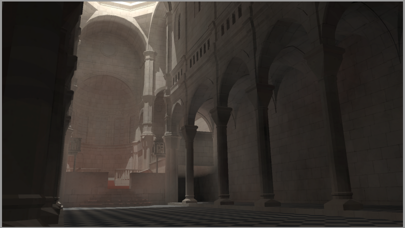
Motivations

Figure: Screenshot from Battlefield 1 (Release Oct. 2016)

Motivations



(a) Sibenik rendered using path tracing (with Embree in 15s).



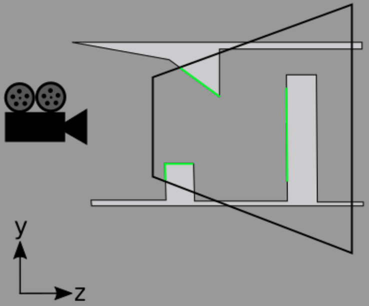
(b) Sibenik rendered using photon splatting (with our cluster-trivial method in 33 ms for 200k photons).

Previous Work: Tiled Photon Splatting from *Toward Practical Real-Time Photon Mapping: Efficient GPU Density Estimation* by Mara et al.

Tiled Photon Splatting from Mara et al. [MLM13]

Algorithm's steps

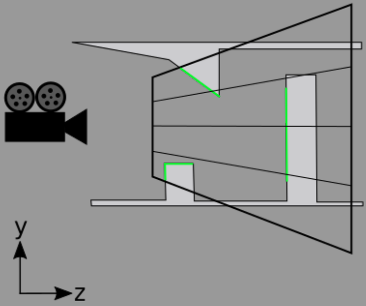
1. Generate the photon map;
2. Rasterize the scene;
3. Tile the view frustrum;
4. Compute a bounding-box per tile;
5. Test all photons against the tiles: append photon to the tile's photon list if intersect;
6. Per tile, shade all contained view-samples using its photon list.



Tiled Photon Splatting from Mara et al. [MLM13]

Algorithm's steps

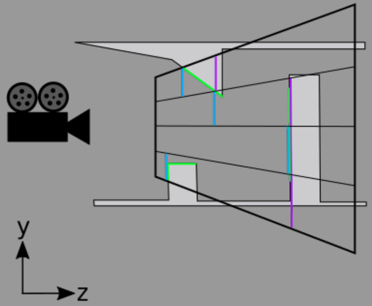
1. Generate the photon map;
2. Rasterize the scene;
3. Tile the view frustrum;
4. Compute a bounding-box per tile;
5. Test all photons against the tiles: append photon to the tile's photon list if intersect;
6. Per tile, shade all contained view-samples using its photon list.



Tiled Photon Splatting from Mara et al. [MLM13]

Algorithm's steps

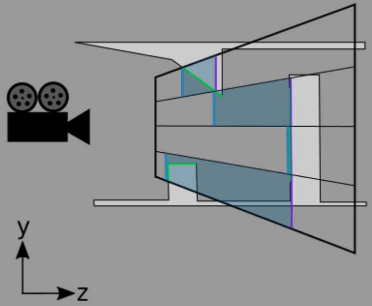
1. Generate the photon map;
2. Rasterize the scene;
3. Tile the view frustrum;
4. Compute a bounding-box per tile;
5. Test all photons against the tiles: append photon to the tile's photon list if intersect;
6. Per tile, shade all contained view-samples using its photon list.



Tiled Photon Splatting from Mara et al. [MLM13]

Algorithm's steps

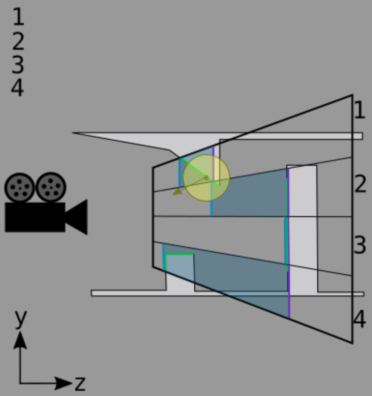
1. Generate the photon map;
2. Rasterize the scene;
3. Tile the view frustrum;
4. Compute a bounding-box per tile;
5. Test all photons against the tiles: append photon to the tile's photon list if intersect;
6. Per tile, shade all contained view-samples using its photon list.



Tiled Photon Splatting from Mara et al. [MLM13]

Algorithm's steps

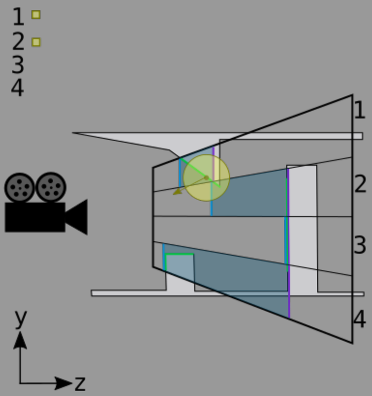
1. Generate the photon map;
2. Rasterize the scene;
3. Tile the view frustrum;
4. Compute a bounding-box per tile;
5. Test all photons against the tiles: append photon to the tile's photon list if intersect;
6. Per tile, shade all contained view-samples using its photon list.



Tiled Photon Splatting from Mara et al. [MLM13]

Algorithm's steps

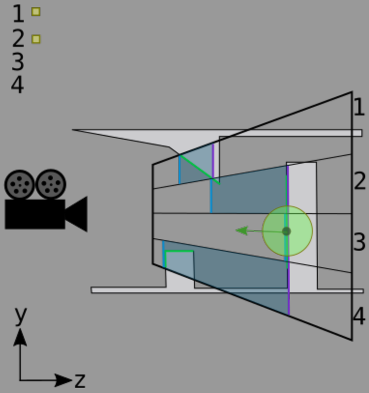
1. Generate the photon map;
2. Rasterize the scene;
3. Tile the view frustrum;
4. Compute a bounding-box per tile;
5. Test all photons against the tiles: append photon to the tile's photon list if intersect;
6. Per tile, shade all contained view-samples using its photon list.



Tiled Photon Splatting from Mara et al. [MLM13]

Algorithm's steps

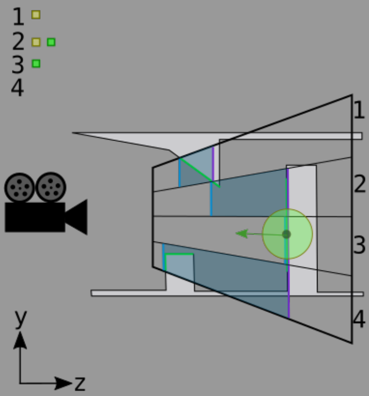
1. Generate the photon map;
2. Rasterize the scene;
3. Tile the view frustrum;
4. Compute a bounding-box per tile;
5. Test all photons against the tiles: append photon to the tile's photon list if intersect;
6. Per tile, shade all contained view-samples using its photon list.



Tiled Photon Splatting from Mara et al. [MLM13]

Algorithm's steps

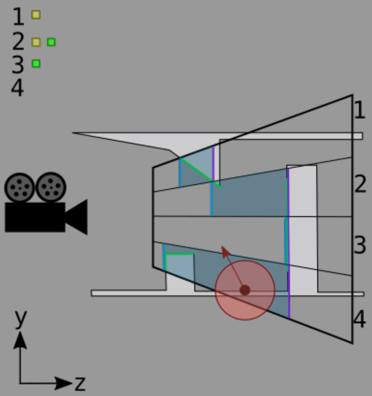
1. Generate the photon map;
2. Rasterize the scene;
3. Tile the view frustrum;
4. Compute a bounding-box per tile;
5. Test all photons against the tiles: append photon to the tile's photon list if intersect;
6. Per tile, shade all contained view-samples using its photon list.



Tiled Photon Splatting from Mara et al. [MLM13]

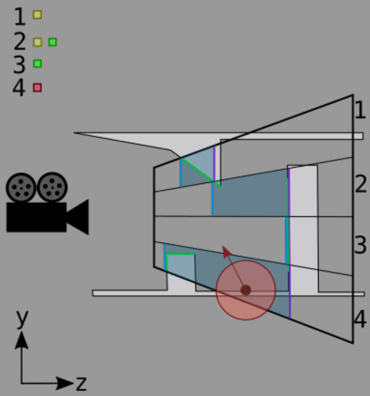
Algorithm's steps

1. Generate the photon map;
2. Rasterize the scene;
3. Tile the view frustrum;
4. Compute a bounding-box per tile;
5. Test all photons against the tiles: append photon to the tile's photon list if intersect;
6. Per tile, shade all contained view-samples using its photon list.



Algorithm's steps

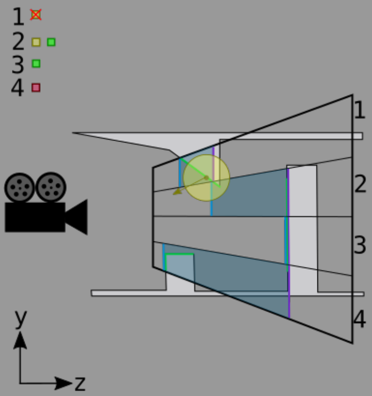
1. Generate the photon map;
2. Rasterize the scene;
3. Tile the view frustrum;
4. Compute a bounding-box per tile;
5. Test all photons against the tiles: append photon to the tile's photon list if intersect;
6. Per tile, shade all contained view-samples using its photon list.



Tiled Photon Splatting from Mara et al. [MLM13]

Algorithm's steps

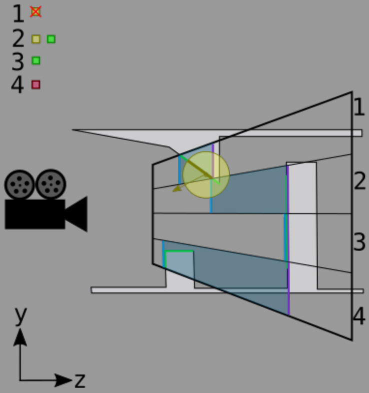
1. Generate the photon map;
2. Rasterize the scene;
3. Tile the view frustrum;
4. Compute a bounding-box per tile;
5. Test all photons against the tiles: append photon to the tile's photon list if intersect;
6. Per tile, shade all contained view-samples using its photon list.



Tiled Photon Splatting from Mara et al. [MLM13]

Algorithm's steps

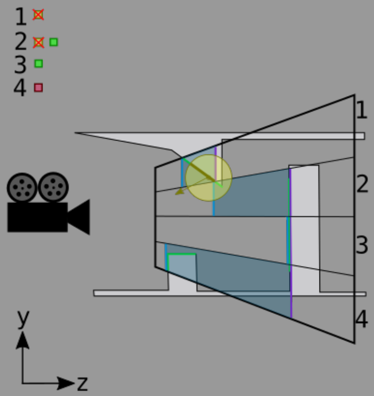
1. Generate the photon map;
2. Rasterize the scene;
3. Tile the view frustrum;
4. Compute a bounding-box per tile;
5. Test all photons against the tiles: append photon to the tile's photon list if intersect;
6. Per tile, shade all contained view-samples using its photon list.



Tiled Photon Splatting from Mara et al. [MLM13]

Algorithm's steps

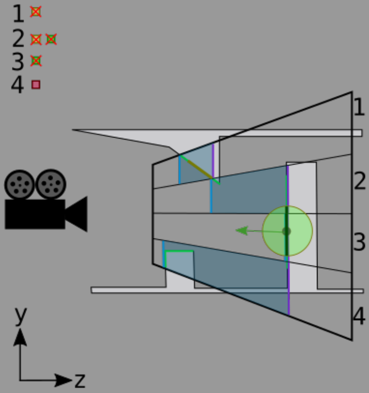
1. Generate the photon map;
2. Rasterize the scene;
3. Tile the view frustrum;
4. Compute a bounding-box per tile;
5. Test all photons against the tiles: append photon to the tile's photon list if intersect;
6. Per tile, shade all contained view-samples using its photon list.



Tiled Photon Splatting from Mara et al. [MLM13]

Algorithm's steps

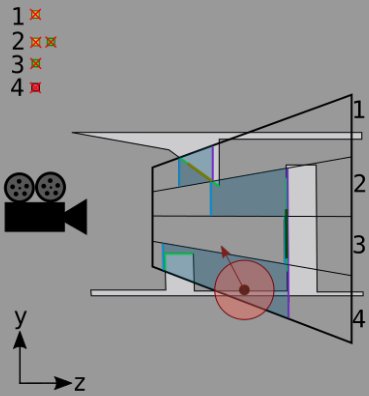
1. Generate the photon map;
2. Rasterize the scene;
3. Tile the view frustrum;
4. Compute a bounding-box per tile;
5. Test all photons against the tiles: append photon to the tile's photon list if intersect;
6. Per tile, shade all contained view-samples using its photon list.



Tiled Photon Splatting from Mara et al. [MLM13]

Algorithm's steps

1. Generate the photon map;
2. Rasterize the scene;
3. Tile the view frustrum;
4. Compute a bounding-box per tile;
5. Test all photons against the tiles: append photon to the tile's photon list if intersect;
6. Per tile, shade all contained view-samples using its photon list.

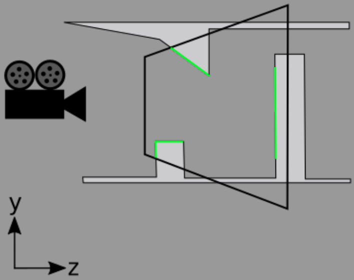


Previous Work: View-Sample Cluster Hierarchy from *Per-Triangle Shadow Volumes Using a View-Sample Cluster Hierarchy* by Sintorn et al.

View-Sample Cluster Hierarchy from Sintorn et al. [SKOA14]

Algorithm's steps

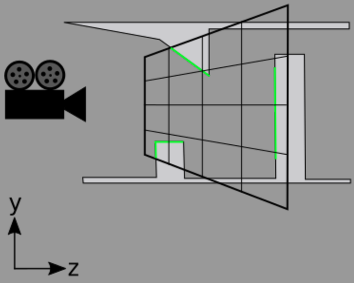
1. Rasterize the scene;
2. Divide the view frustrum in clusters;
3. Mark clusters containing view-samples;
4. Compute a bounding-box per cluster;
5. If current cluster is not the root node, compute its parent and restart from step 4.



View-Sample Cluster Hierarchy from Sintorn et al. [SKOA14]

Algorithm's steps

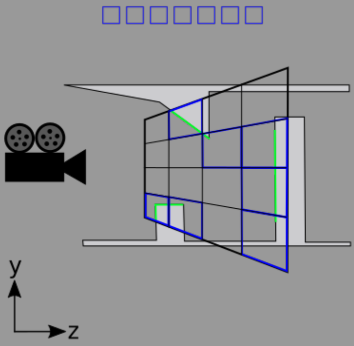
1. Rasterize the scene;
2. Divide the view frustrum in clusters;
3. Mark clusters containing view-samples;
4. Compute a bounding-box per cluster;
5. If current cluster is not the root node, compute its parent and restart from step 4.



View-Sample Cluster Hierarchy from Sintorn et al. [SKOA14]

Algorithm's steps

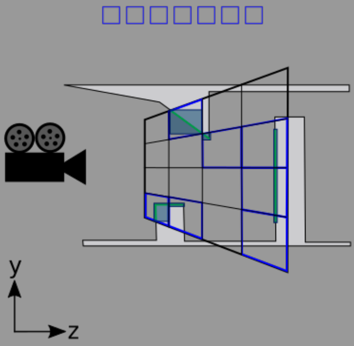
1. Rasterize the scene;
2. Divide the view frustrum in clusters;
3. Mark clusters containing view-samples;
4. Compute a bounding-box per cluster;
5. If current cluster is not the root node, compute its parent and restart from step 4.



View-Sample Cluster Hierarchy from Sintorn et al. [SKOA14]

Algorithm's steps

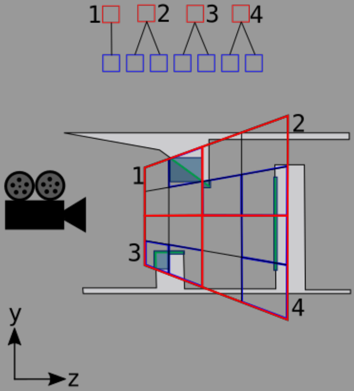
1. Rasterize the scene;
2. Divide the view frustrum in clusters;
3. Mark clusters containing view-samples;
4. Compute a bounding-box per cluster;
5. If current cluster is not the root node, compute its parent and restart from step 4.



View-Sample Cluster Hierarchy from Sintorn et al. [SKOA14]

Algorithm's steps

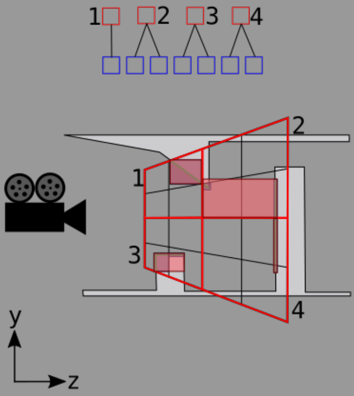
1. Rasterize the scene;
2. Divide the view frustrum in clusters;
3. Mark clusters containing view-samples;
4. Compute a bounding-box per cluster;
5. If current cluster is not the root node, compute its parent and restart from step 4.



View-Sample Cluster Hierarchy from Sintorn et al. [SKOA14]

Algorithm's steps

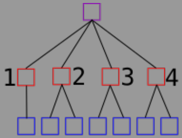
1. Rasterize the scene;
2. Divide the view frustrum in clusters;
3. Mark clusters containing view-samples;
4. Compute a bounding-box per cluster;
5. If current cluster is not the root node, compute its parent and restart from step 4.

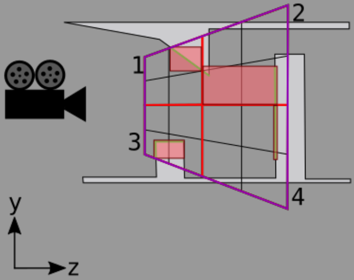


View-Sample Cluster Hierarchy from Sintorn et al. [SKOA14]

Algorithm's steps

1. Rasterize the scene;
2. Divide the view frustrum in clusters;
3. Mark clusters containing view-samples;
4. Compute a bounding-box per cluster;
5. If current cluster is not the root node, compute its parent and restart from step 4.

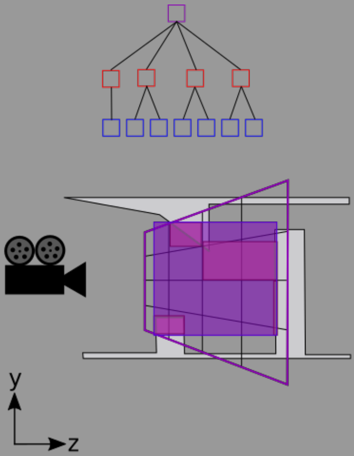




View-Sample Cluster Hierarchy from Sintorn et al. [SKOA14]

Algorithm's steps

1. Rasterize the scene;
2. Divide the view frustrum in clusters;
3. Mark clusters containing view-samples;
4. Compute a bounding-box per cluster;
5. If current cluster is not the root node, compute its parent and restart from step 4.

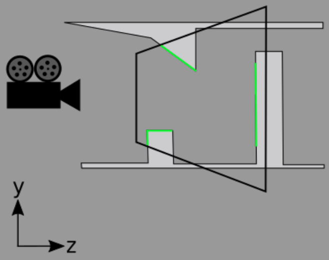


Contributions: Basic Algorithm

Contribution: Basic Algorithm

Algorithm's steps

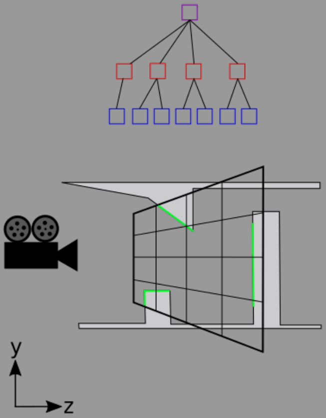
1. Generate the photon map;
2. Rasterize the scene;
3. Generate the cluster hierarchy;
4. Test all photons against the hierarchy; append photon to the cluster's photon list if intersect;
5. Per cluster, shade all contained view-samples using its photon list.



Contribution: Basic Algorithm

Algorithm's steps

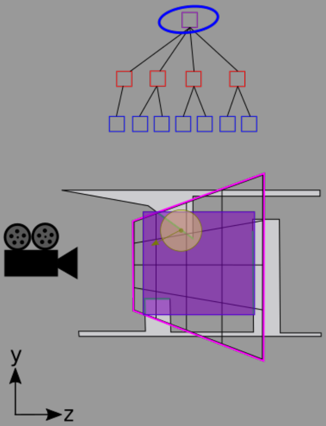
1. Generate the photon map;
2. Rasterize the scene;
3. Generate the cluster hierarchy;
4. Test all photons against the hierarchy; append photon to the cluster's photon list if intersect;
5. Per cluster, shade all contained view-samples using its photon list.



Contribution: Basic Algorithm

Algorithm's steps

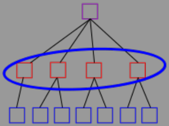
1. Generate the photon map;
2. Rasterize the scene;
3. Generate the cluster hierarchy;
4. Test all photons against the hierarchy; append photon to the cluster's photon list if intersect;
5. Per cluster, shade all contained view-samples using its photon list.

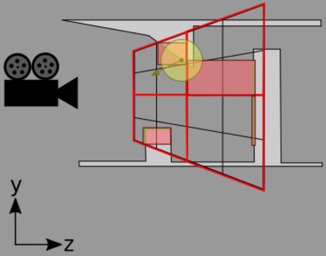


Contribution: Basic Algorithm

Algorithm's steps

1. Generate the photon map;
2. Rasterize the scene;
3. Generate the cluster hierarchy;
4. Test all photons against the hierarchy; append photon to the cluster's photon list if intersect;
5. Per cluster, shade all contained view-samples using its photon list.

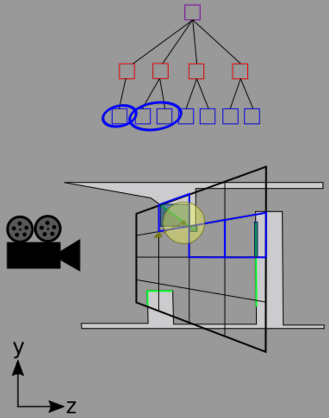




Contribution: Basic Algorithm

Algorithm's steps

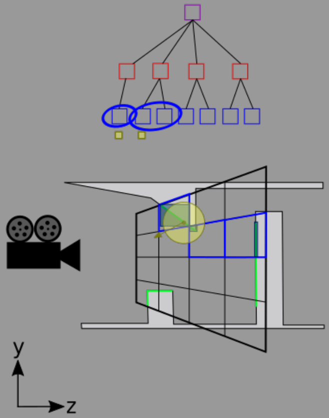
1. Generate the photon map;
2. Rasterize the scene;
3. Generate the cluster hierarchy;
4. Test all photons against the hierarchy; append photon to the cluster's photon list if intersect;
5. Per cluster, shade all contained view-samples using its photon list.



Contribution: Basic Algorithm

Algorithm's steps

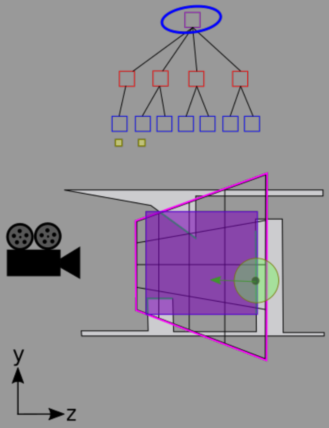
1. Generate the photon map;
2. Rasterize the scene;
3. Generate the cluster hierarchy;
4. Test all photons against the hierarchy; append photon to the cluster's photon list if intersect;
5. Per cluster, shade all contained view-samples using its photon list.



Contribution: Basic Algorithm

Algorithm's steps

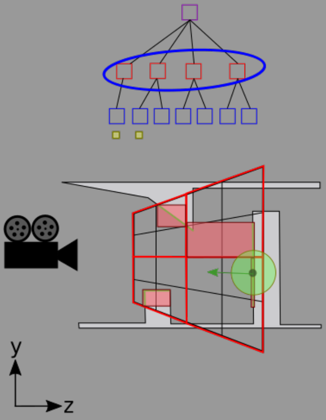
1. Generate the photon map;
2. Rasterize the scene;
3. Generate the cluster hierarchy;
4. Test all photons against the hierarchy; append photon to the cluster's photon list if intersect;
5. Per cluster, shade all contained view-samples using its photon list.



Contribution: Basic Algorithm

Algorithm's steps

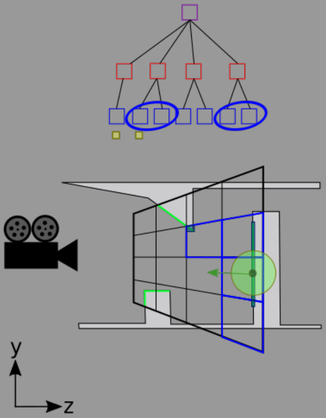
1. Generate the photon map;
2. Rasterize the scene;
3. Generate the cluster hierarchy;
4. Test all photons against the hierarchy; append photon to the cluster's photon list if intersect;
5. Per cluster, shade all contained view-samples using its photon list.



Contribution: Basic Algorithm

Algorithm's steps

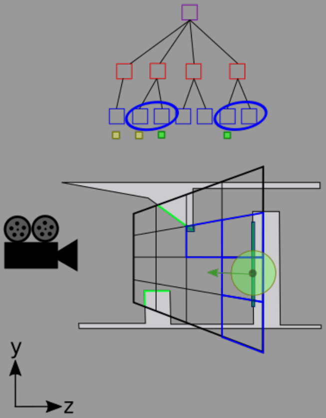
1. Generate the photon map;
2. Rasterize the scene;
3. Generate the cluster hierarchy;
4. Test all photons against the hierarchy; append photon to the cluster's photon list if intersect;
5. Per cluster, shade all contained view-samples using its photon list.



Contribution: Basic Algorithm

Algorithm's steps

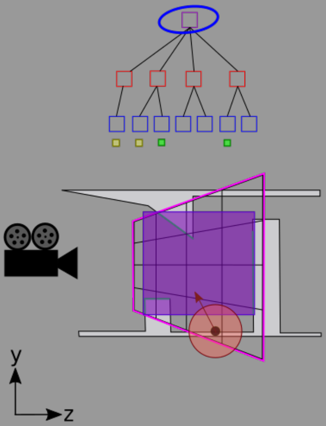
1. Generate the photon map;
2. Rasterize the scene;
3. Generate the cluster hierarchy;
4. Test all photons against the hierarchy; append photon to the cluster's photon list if intersect;
5. Per cluster, shade all contained view-samples using its photon list.



Contribution: Basic Algorithm

Algorithm's steps

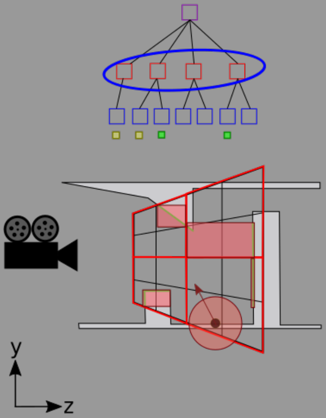
1. Generate the photon map;
2. Rasterize the scene;
3. Generate the cluster hierarchy;
4. Test all photons against the hierarchy; append photon to the cluster's photon list if intersect;
5. Per cluster, shade all contained view-samples using its photon list.



Contribution: Basic Algorithm

Algorithm's steps

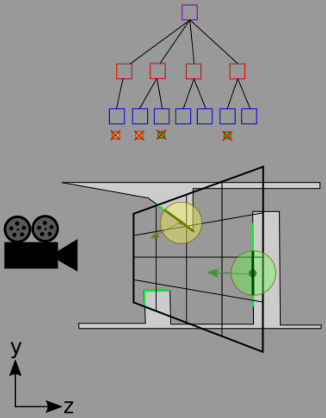
1. Generate the photon map;
2. Rasterize the scene;
3. Generate the cluster hierarchy;
4. Test all photons against the hierarchy; append photon to the cluster's photon list if intersect;
5. Per cluster, shade all contained view-samples using its photon list.



Contribution: Basic Algorithm

Algorithm's steps

1. Generate the photon map;
2. Rasterize the scene;
3. Generate the cluster hierarchy;
4. Test all photons against the hierarchy; append photon to the cluster's photon list if intersect;
5. Per cluster, shade all contained view-samples using its photon list.



Results: Basic Algorithm Efficiency

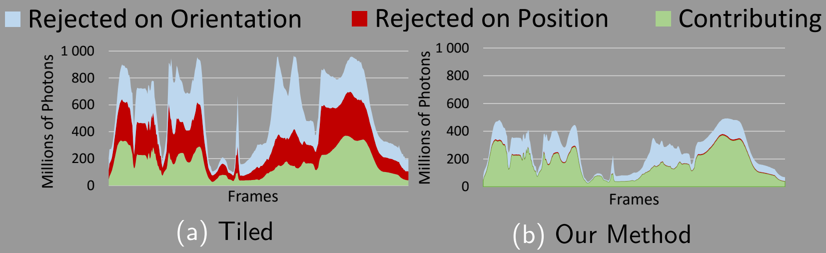


Figure: Contributing photons versus non-contributing photons during shading, for a fly-through in Sponza using 10k photons of radius 4.

Our method

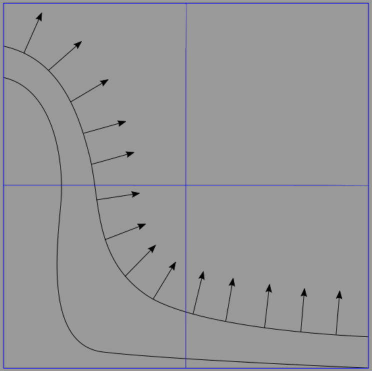
- Rejects less photons based on position;
- Fetches half as many photons from memory.

Contributions: Normal-Cone Hierarchy Optimisation

Contribution: Constructing the Normal-Cone Hierarchy

Algorithm's steps

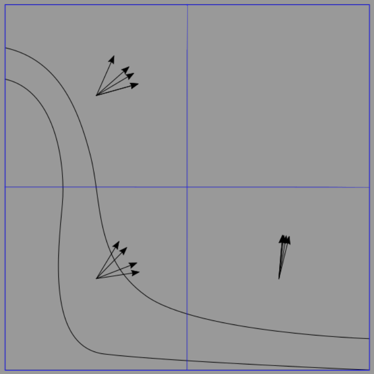
1. Per cluster, compute its normal-cone;
2. If current cluster is not the root node, move one level up in the hierarchy and restart from step 1.



Contribution: Constructing the Normal-Cone Hierarchy

Algorithm's steps

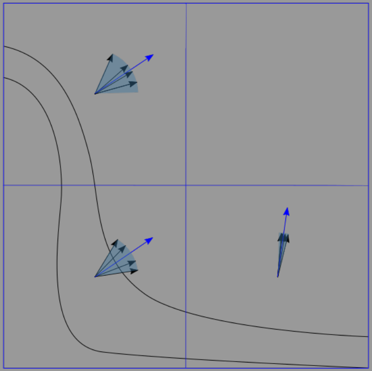
1. Per cluster, compute its normal-cone;
2. If current cluster is not the root node, move one level up in the hierarchy and restart from step 1.



Contribution: Constructing the Normal-Cone Hierarchy

Algorithm's steps

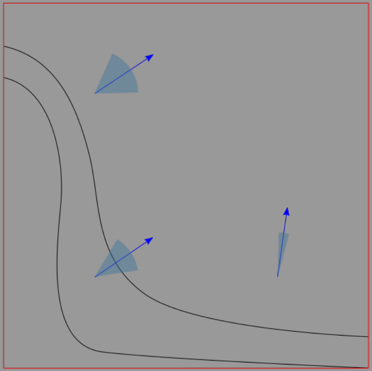
1. Per cluster, compute its normal-cone;
2. If current cluster is not the root node, move one level up in the hierarchy and restart from step 1.



Contribution: Constructing the Normal-Cone Hierarchy

Algorithm's steps

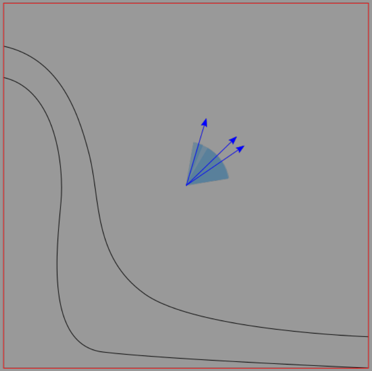
1. Per cluster, compute its normal-cone;
2. If current cluster is not the root node, move one level up in the hierarchy and restart from step 1.



Contribution: Constructing the Normal-Cone Hierarchy

Algorithm's steps

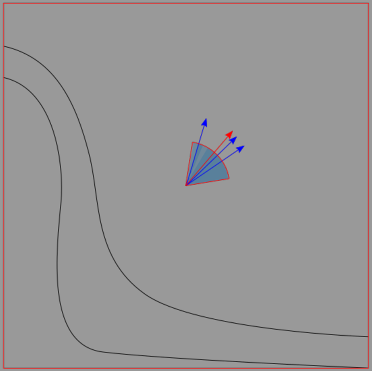
1. Per cluster, compute its normal-cone;
2. If current cluster is not the root node, move one level up in the hierarchy and restart from step 1.



Contribution: Constructing the Normal-Cone Hierarchy

Algorithm's steps

1. Per cluster, compute its normal-cone;
2. If current cluster is not the root node, move one level up in the hierarchy and restart from step 1.



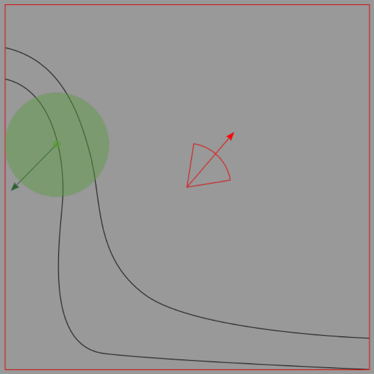
Contribution: Using the Normal-Cone Hierarchy

Trivial reject

If the photon's direction is at more than $\frac{\pi}{2}$ from all normals contained in the normal cone: stop traversing and drop the photon.

Trivial accept (diffuse-only)

If the photon's direction is at less than $\frac{\pi}{2}$ from all normals contained in the normal cone, and it fully encloses the cluster: stop traversing and store the photon's energy.



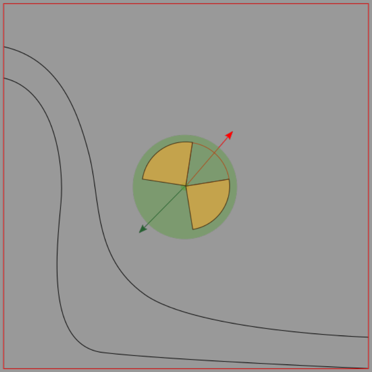
Contribution: Using the Normal-Cone Hierarchy

Trivial reject

If the photon's direction is at more than $\frac{\pi}{2}$ from all normals contained in the normal cone: stop traversing and drop the photon.

Trivial accept (diffuse-only)

If the photon's direction is at less than $\frac{\pi}{2}$ from all normals contained in the normal cone, and it fully encloses the cluster: stop traversing and store the photon's energy.



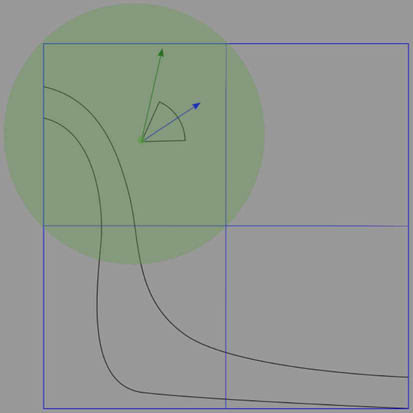
Contribution: Using the Normal-Cone Hierarchy

Trivial reject

If the photon's direction is at more than $\frac{\pi}{2}$ from all normals contained in the normal cone: stop traversing and drop the photon.

Trivial accept (diffuse-only)

If the photon's direction is at less than $\frac{\pi}{2}$ from all normals contained in the normal cone, and it fully encloses the cluster: stop traversing and store the photon's energy.



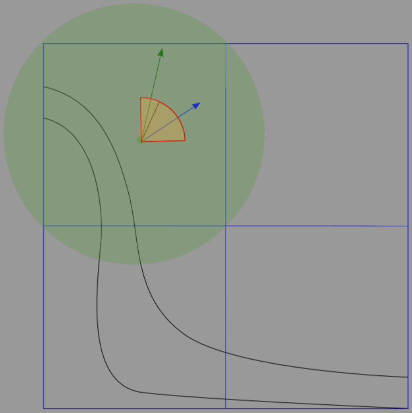
Contribution: Using the Normal-Cone Hierarchy

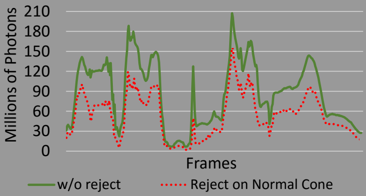
Trivial reject

If the photon's direction is at more than $\frac{\pi}{2}$ from all normals contained in the normal cone: stop traversing and drop the photon.

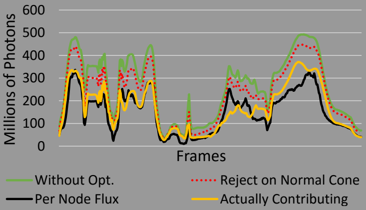
Trivial accept (diffuse-only)

If the photon's direction is at less than $\frac{\pi}{2}$ from all normals contained in the normal cone, and it fully encloses the cluster: stop traversing and store the photon's energy.





(a) Comparison of number of photons rejected due to orientation



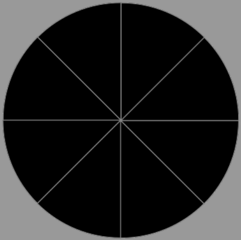
(b) Comparison of number of photons read

Contributions: Directional Radiant Intensity Accumulation

Contribution: Directional Radiant Intensity Accumulation

Algorithm's steps

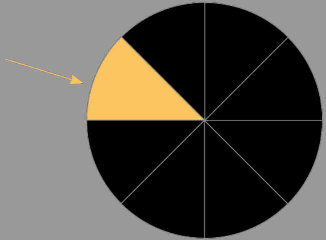
1. Rasterize the scene;
2. Generate the cluster hierarchy;
3. Test all photons against the hierarchy; accumulate incoming flux into a bin of the cluster if photon intersects;
4. Per cluster, shade all view-samples by weighting the accumulated flux of the cluster and its parents.



Contribution: Directional Radiant Intensity Accumulation

Algorithm's steps

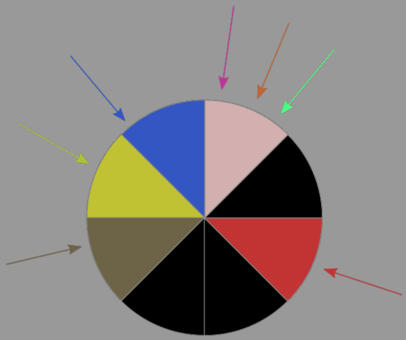
1. Rasterize the scene;
2. Generate the cluster hierarchy;
3. Test all photons against the hierarchy; accumulate incoming flux into a bin of the cluster if photon intersects;
4. Per cluster, shade all view-samples by weighting the accumulated flux of the cluster and its parents.



Contribution: Directional Radiant Intensity Accumulation

Algorithm's steps

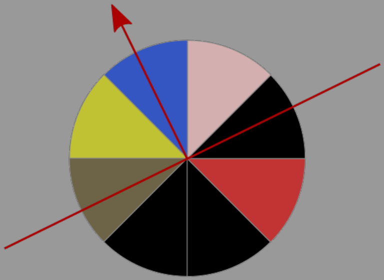
1. Rasterize the scene;
2. Generate the cluster hierarchy;
3. Test all photons against the hierarchy; accumulate incoming flux into a bin of the cluster if photon intersects;
4. Per cluster, shade all view-samples by weighting the accumulated flux of the cluster and its parents.



Contribution: Directional Radiant Intensity Accumulation

Algorithm's steps

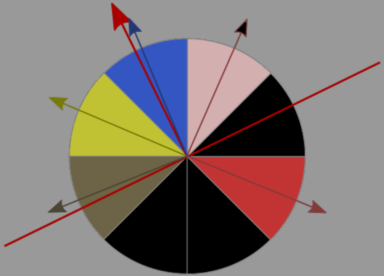
1. Rasterize the scene;
2. Generate the cluster hierarchy;
3. Test all photons against the hierarchy; accumulate incoming flux into a bin of the cluster if photon intersects;
4. Per cluster, shade all view-samples by weighting the accumulated flux of the cluster and its parents.



Contribution: Directional Radiant Intensity Accumulation

Algorithm's steps

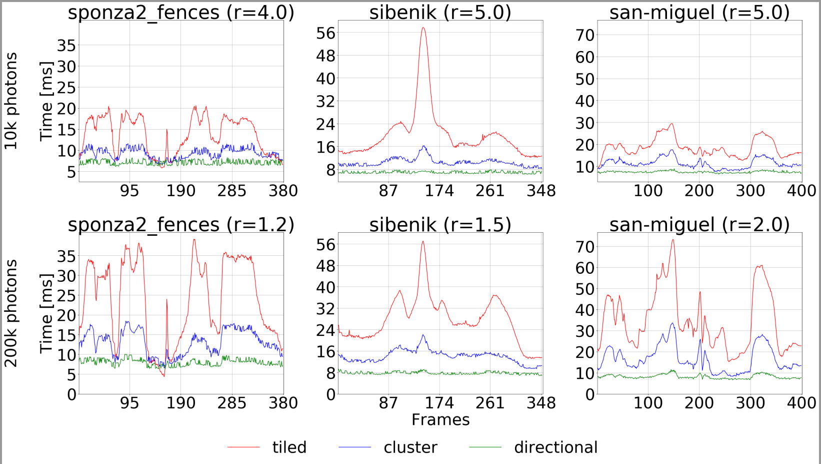
1. Rasterize the scene;
2. Generate the cluster hierarchy;
3. Test all photons against the hierarchy; accumulate incoming flux into a bin of the cluster if photon intersects;
4. Per cluster, shade all view-samples by weighting the accumulated flux of the cluster and its parents.



$$I = a * 2*\overline{\cos}(\theta_a) + b * 2*\overline{\cos}(\theta_b) + c * 2*\overline{\cos}(\theta_c) + d * 2*\overline{\cos}(\theta_d) + e * 2*\overline{\cos}(\theta_e)$$

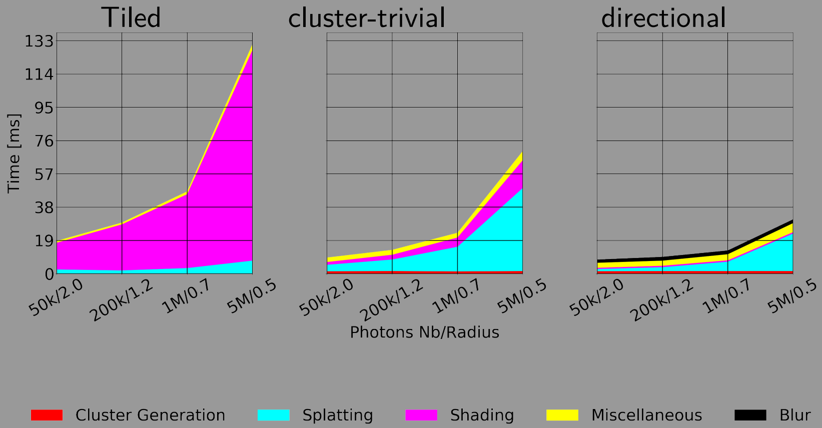
Results

Results: Fly-Through Splatting Time



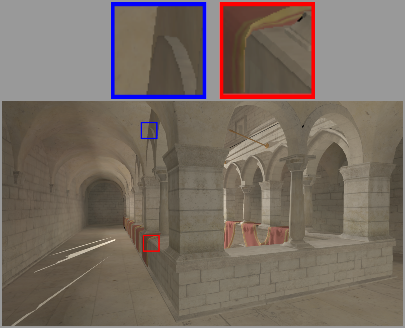
GPU: GTX Titan X; photons traced using OptiX

Results: Sponza Frame Time Breakdown

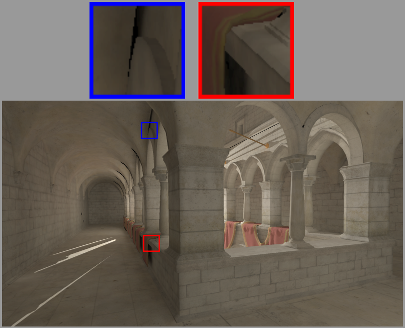


Miscellaneous consists of buffer clearing, texture mapping and unmapping.

Results: Low-Quality vs High-Quality Using cluster-trivial



(a) 10k photons of radius 4 in 10 ms (splatting only



(b) 50M photons of radius 0.2 in 290 ms (splatting only)

Results: Rendering Glossy Materials of Different Roughness

Total rendering time per frame: 34 ms (50k photons of radius 1).

Conclusion

Cluster-trivial: Efficient Splatting

- Tight spatial bounds: handles nicely views with depth complexity;
- Normal-Cones: allows for even more efficiency and performance;
- 2x faster on average than tiled.

Directional: Approximate but Fast

- Splatting: 8 ms (total frame time: 16 ms) for 10k photons;
- Splatting: 10 ms (total frame time: 26 ms) for 200k photons;
- View-independent performance.

Future Work

Improve Splatting Performance

Make a better use of shared memory by using it to store photons from higher levels' list.

Hybrid Path-Tracing/Photon-Splatting

Estimate a path per cluster, then place a photon on each path's first bounce with the path's carried energy.

Reflectance-Cone

In directional, replace unit sphere with a reflectance cone.

Thank You!

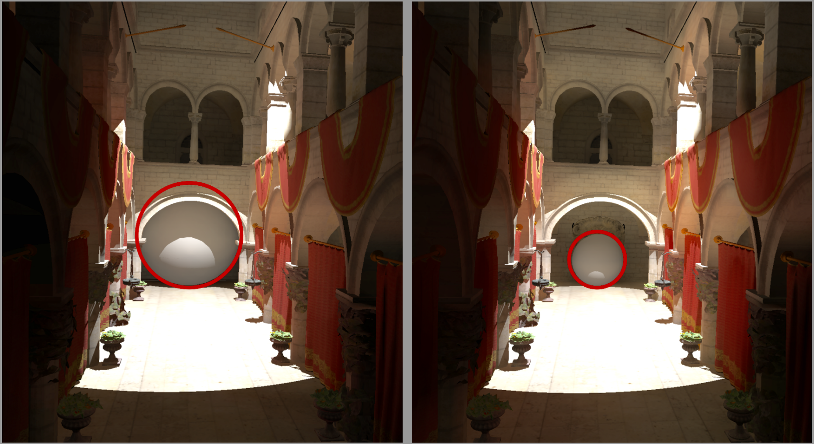
200k photons of radius 3

http://fileadmin.cs.lth.se/graphics/research/papers/2016/splatting/

Bibliography

- Michael Mara, David Luebke, and Morgan McGuire.
  Toward practical real-time photon mapping: Efficient gpu density estimation.
  In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '13, pages 71–78, New York, NY, USA, 2013. ACM.
- Wolfgang Stürzlinger and Rui Bastos.
  Interactive rendering of globally illuminated glossy scenes.
  In Julie Dorsey and Philipp Slusallek, editors, *Rendering Techniques '97*, Eurographics, pages 93–102. Springer Vienna, 1997.
- Erik Sintorn, Viktor Kämpe, Ola Olsson, and Ulf Assarsson.
  Per-triangle shadow volumes using a view-sample cluster hierarchy.
  In *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '14, pages 111–118, New York, NY, USA, 2014. ACM.

Photons' Size Reference



(a) 10k photons, radius 4 (b) 200k photons, radius 2

Results: Memory Consumption Comparison

(MiB)	cluster-trivial (8×8)	Directional (8 regions)	Tiled (32×32)
Tiles z-Bounds	-	-	0.016
Cluster Hierarchy	97	97	-
Final Bounds	256	256	-
Normal Cone	24	-	-
Accum. Flux	73	582	-
Sub-Total	450	935	0.016
Jobs	2.4	2.4	-
Photons Array	60	-	28
Photon Map	0.16	0.16	0.16
Sub-Total	63	7.2	28
Total	513	942	28

Table: Memory-consumption breakdown at a resolution of 1080p and using 10k photons of radius 4 in Sponza.

Image Quality Using SSIM and PSNR: Sponza

Photons Nb,	Cluster		Directional		Tiled	
Radius	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR
10k , 4.0	90	25	91	27	90	24
50k , 2.0	91	26	90	25	92	27
200k , 1.2	93	30	92	27	93	30
1M, 0.7	94	32	92	27	94	32
5M, 0.5	94	32	92	27	94	32
50M, 0.2	95	33	89	25	95	33

Table: SSIM (in %) and PSNR (in dB) results for various setups across the three test scenes using the cluster-trivial and the directional methods against a path traced reference image generated using Embree.

Image Quality Using SSIM and PSNR: Sibenik

Photons Nb,	Cluster		Directional		Tiled	
Radius	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR
10k , 5.0	75	22	81	25	76	22
50k , 3.0	82	26	83	26	81	25
200k , 2.0	84	26	83	26	83	26
1M, 1.0	85	27	82	26	84	27
5M, 0.5	85	28	82	26	85	27
50M, 0.2	86	28	83	27	85	28

Table: SSIM (in %) and PSNR (in dB) results for various setups across the three test scenes using the cluster-trivial and the directional methods against a path traced reference image generated using Embree.

Image Quality Using SSIM and PSNR: San Miguel

Photons Nb, Radius	Cluster SSIM	Cluster PSNR	Directional SSIM	Directional PSNR	Tiled SSIM	Tiled PSNR
10k , 5.0	91	28	87	27	92	27
50k , 2.0	88	31	85	28	89	31
200k , 1.5	94	32	89	28	93	32
1M, 1.0	95	34	90	28	94	34
5M, 0.6	96	36	94	30	95	36
50M, 0.2	96	38	93	30	96	38

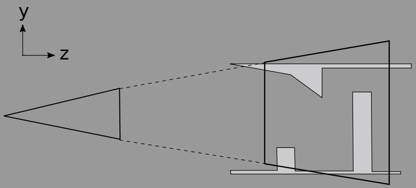
Table: SSIM (in %) and PSNR (in dB) results for various setups across the three test scenes using the cluster-trivial and the directional methods against a path traced reference image generated using Embree.

Appendix
Cluster Key Computation

Photon Splatting Using 3D Bounds

Presented by Stürzlinger et al. [SB97]

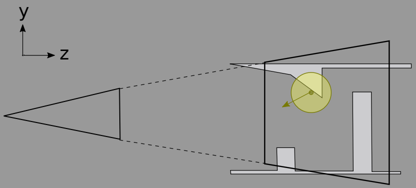
1. Rasterize scene
2. Rasterize photons (as 3D spheres) on top of scene's depthbuffer
3. Shade view-samples within the photon's sphere of influence



Photon Splatting Using 3D Bounds

Presented by Stürzlinger et al. [SB97]

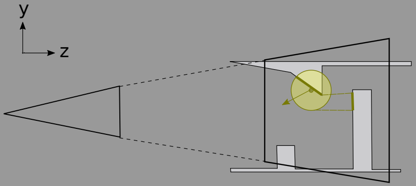
1. Rasterize scene
2. Rasterize photons (as 3D spheres) on top of scene's depthbuffer
3. Shade view-samples within the photon's sphere of influence



Photon Splatting Using 3D Bounds

Presented by Stürzlinger et al. [SB97]

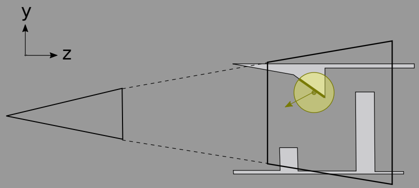
1. Rasterize scene
2. Rasterize photons (as 3D spheres) on top of scene's depthbuffer
3. Shade view-samples within the photon's sphere of influence



Photon Splatting Using 3D Bounds

Presented by Stürzlinger et al. [SB97]

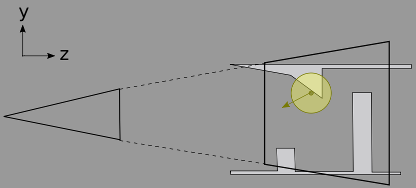
1. Rasterize scene
2. Rasterize photons (as 3D spheres) on top of scene's depthbuffer
3. Shade view-samples within the photon's sphere of influence



Photon Splatting Using 2.5D Bounds

Presented by Mara et al. [MLM13]

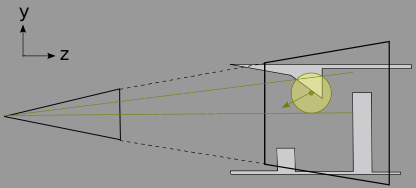
1. Rasterize scene
2. Compute sphere silhouette as seen from camera
3. Rasterize silhouette (as 2D polygon) on top of scene's depthbuffer
4. Shade view-samples within the photon's sphere of influence



Photon Splatting Using 2.5D Bounds

Presented by Mara et al. [MLM13]

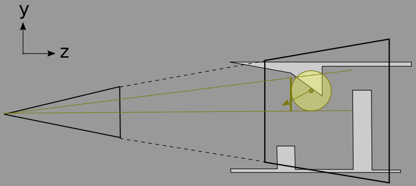
1. Rasterize scene
2. Compute sphere silhouette as seen from camera
3. Rasterize silhouette (as 2D polygon) on top of scene's depthbuffer
4. Shade view-samples within the photon's sphere of influence



Photon Splatting Using 2.5D Bounds

Presented by Mara et al. [MLM13]

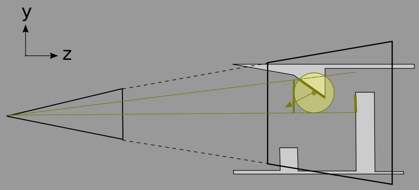
1. Rasterize scene
2. Compute sphere silhouette as seen from camera
3. Rasterize silhouette (as 2D polygon) on top of scene's depthbuffer
4. Shade view-samples within the photon's sphere of influence



Photon Splatting Using 2.5D Bounds

Presented by Mara et al. [MLM13]

1. Rasterize scene
2. Compute sphere silhouette as seen from camera
3. Rasterize silhouette (as 2D polygon) on top of scene's depthbuffer
4. Shade view-samples within the photon's sphere of influence



Photon Splatting Using 2.5D Bounds

Presented by Mara et al. [MLM13]

1. Rasterize scene
2. Compute sphere silhouette as seen from camera
3. Rasterize silhouette (as 2D polygon) on top of scene's depthbuffer
4. Shade view-samples within the photon's sphere of influence