Gradient Estimation for Real-Time Adaptive Temporal Filtering

Onur Basci

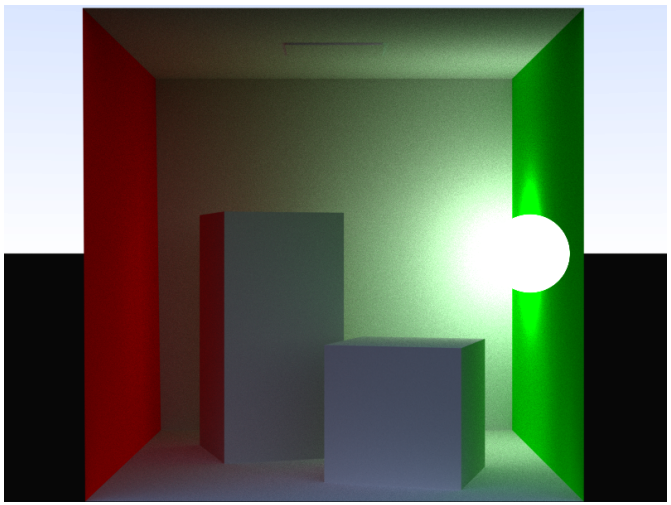
Sorbonne Université

February 6, 2025

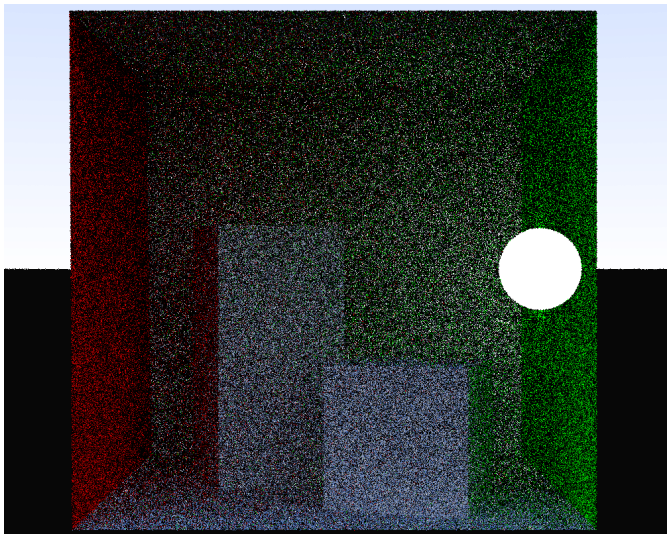
Contents

Path tracing

- Produces highly realistic results with enough samples per pixel
- Models global illumination and soft shadows
- Generates very noisy results with small sample count
- In real-time, we can only cast 1 ray per pixel



8192 samples per pixel

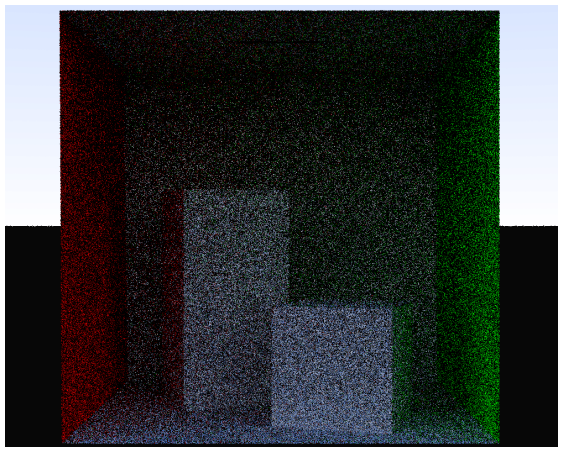


1 sample per pixel

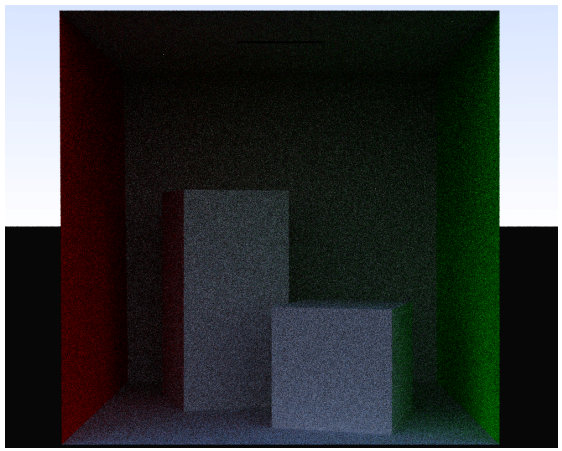
Exponential Moving Average

- In real time application, coherence between frames is high
- Idea is to use previous frame information
- The previous frame is reprojected and blended with the current frame.

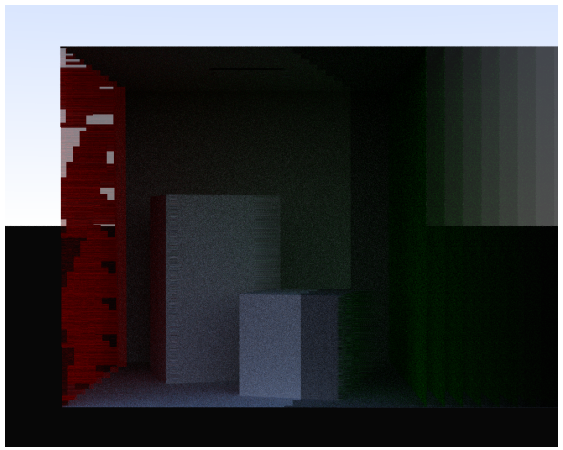
$$\hat{c}_i(x) = \alpha c_i(x) + (1-\alpha)\hat{c}_{i-1}(\overleftarrow{x})$$



Path traced image with 1 spp



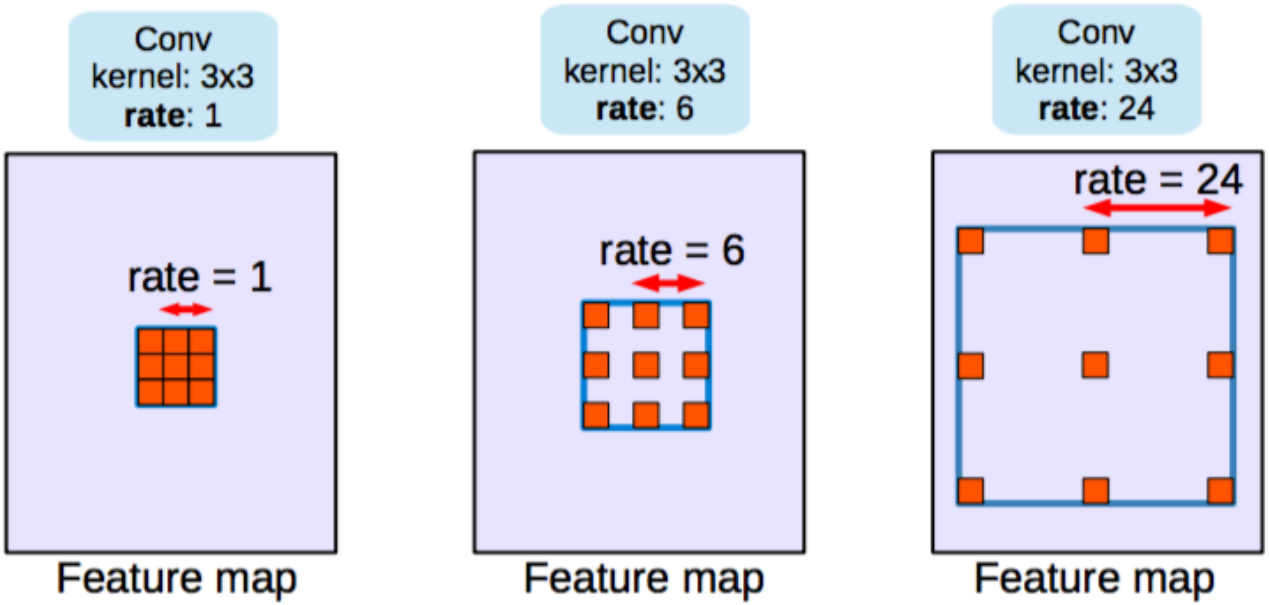
$\alpha = 0.2$



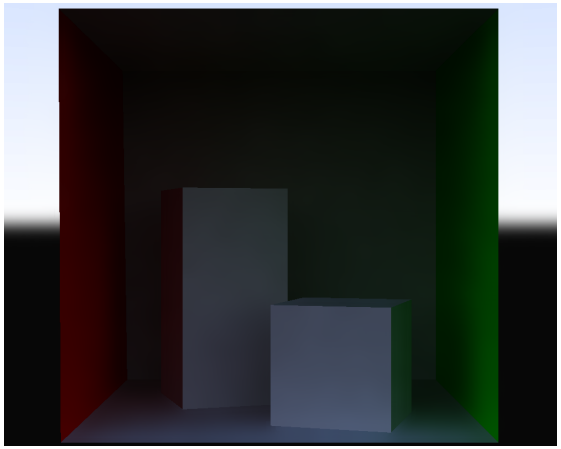
$\alpha = 0.05$ introduces ghosting

SVGF (Spatiotemporal Variance-Guided Filtering)

- EMA does not remove all noise
- We need spatial filtering on the path traced image
- We use an edge preserving A-Trous Wavelet Transform



SVGF



Alpha blending with SVGF filtering

$$\hat{l}^{(k+1)}(p) := \frac{\sum_{q\in\Omega} h^{(k)}(p,q)\cdot w^{(k)}(p,q)\cdot \hat{l}^{(k)}(q)}{\sum_{q\in\Omega} h^{(k)}(p,q)\cdot w^{(k)}(p,q)}$$

$$w^{(k)}(p,q) := w_z(p,q)\cdot w_n(p,q)\cdot w_l^{(k)}(p,q)$$

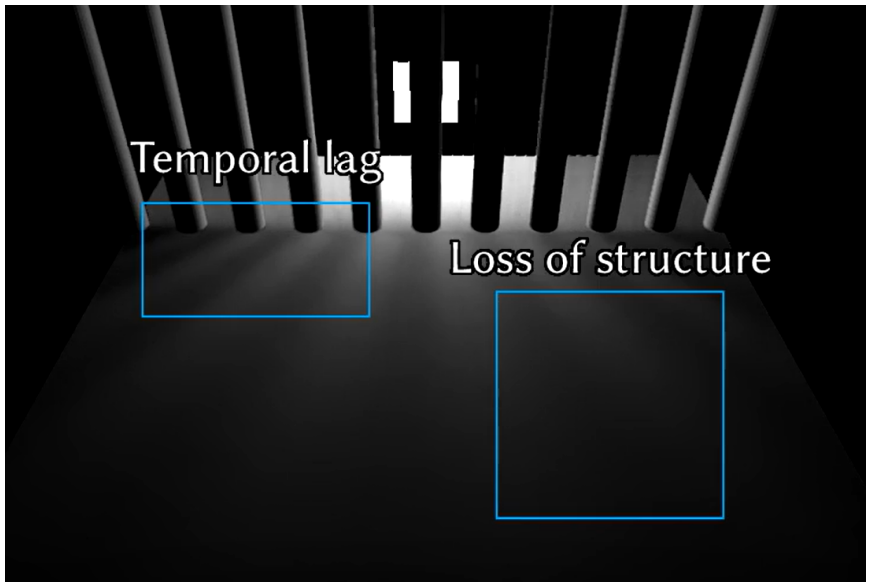
$$w_z(p,q) := \exp\left(-\frac{|z(p)-z(q)|}{\sigma_z\cdot|\langle\nabla z(p), p-q\rangle|}\right)$$

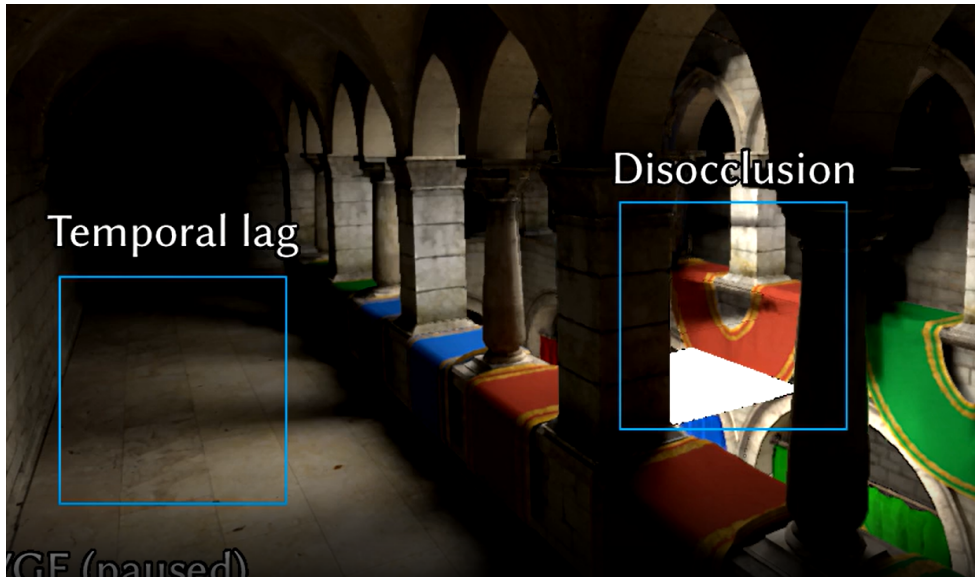
$$\mathsf{w}_n(p,q) := \max(0,\langle n(p), n(q)\rangle)^{\sigma_n}$$

$$\mathsf{w}_l^{(k)}(p,q) := \exp\left(-\frac{|\hat{i}^{(k)}(p)-\hat{i}^{(k)}(q)|}{\sigma_l\cdot\sqrt{g_{3\times3}(\mathsf{Var}(I^{(k)}(p))}}\right)$$

Adaptive Temporal Filtering

- SVGF creates great results for stable scenes
- introduces temporal lag and ghosting in scenes with dramatic changes





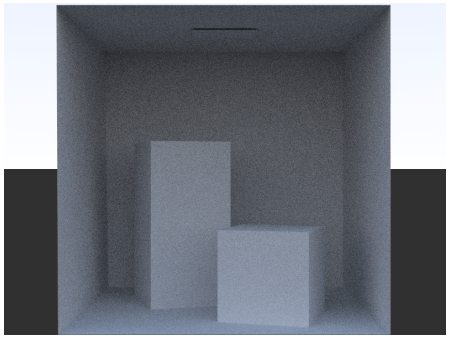
Adaptive Temporal Filtering

- Ghosting and temporal lag occurs due to fixed alpha
- Calculate adaptive alpha value per pixel with **temporal gradient**
- $\delta_{i,\vec{j}} = f_i(G_{i,j}) - f_{i-1}(\overleftarrow{G}_{i,j})$
- Linear interpolation between constant alpha and 1
- $\alpha_i(p) = (1 - \lambda(p)) \cdot \alpha + \lambda(p)$
- We are interested in absolute change
- $\lambda(p) = \min\left(1, \frac{|\hat{\delta}_i(p)|}{\hat{\Delta}_i(p)}\right).$

Implementation/Tools

- Vulkan is a low level graphics API with ray tracing extensions (Ray Query, Acceleration Structure)
- Path tracer algorithm is based from NVIDIA's mini path tracer tutorial
- I used NVIDIA's helper fonctions while working with Vulkan
- RenderDoc for debugging

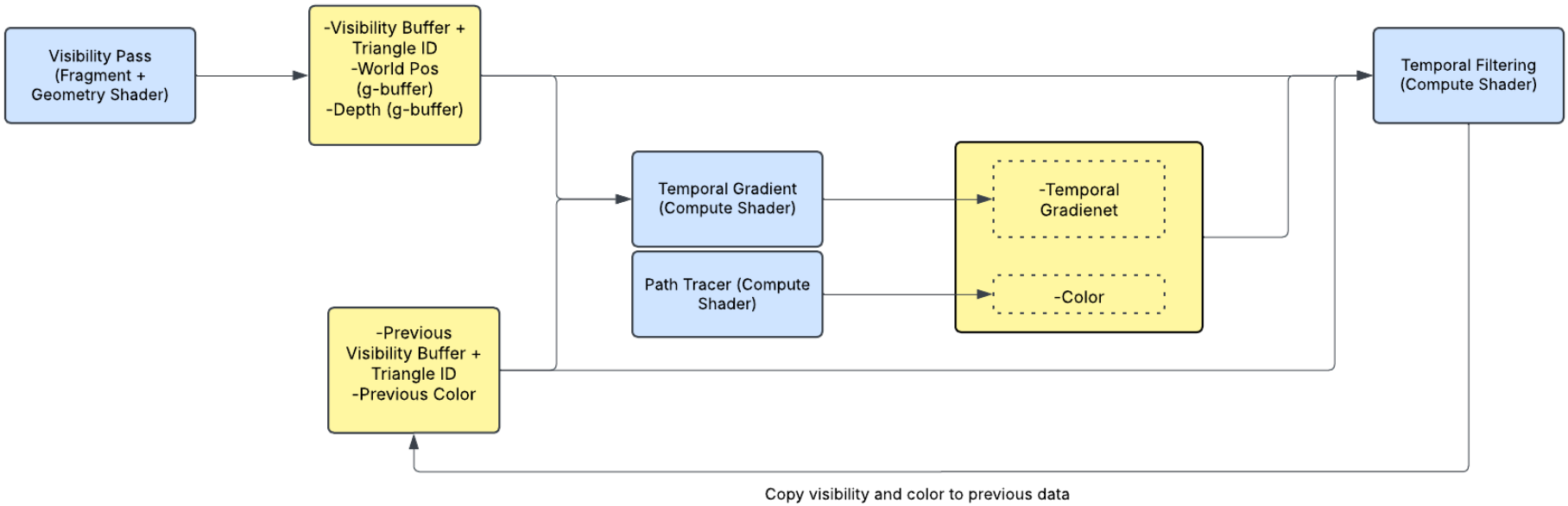
Vulkan API



vk mini path tracer

RendeDoc

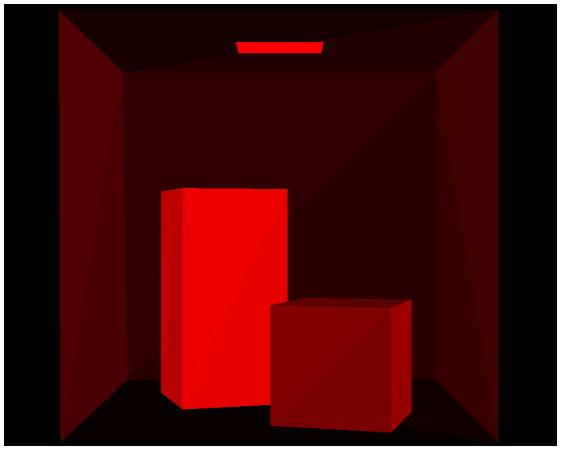
Implementation



The blue boxes represent shaders, while the yellow boxes represent the resources either used by or generated for the shaders, depending on the direction of the arrows

Visibility Pass

- The computation of temporal gradient requires back projection
- $\delta_{i,\vec{j}} = f_i(G_{i,j}) - f_{i-1}(\overleftarrow{G}_{i,j})$
- I store triangle ID in a visibility buffer for each pixel
- I store the triangle vertices for each triangle ID in a visibility lookup table
- Memory friendly approach since we don't store triangle vertices information for each pixel
- We also don't need the store normal information since we can deduce it from normalized cross product between vertices



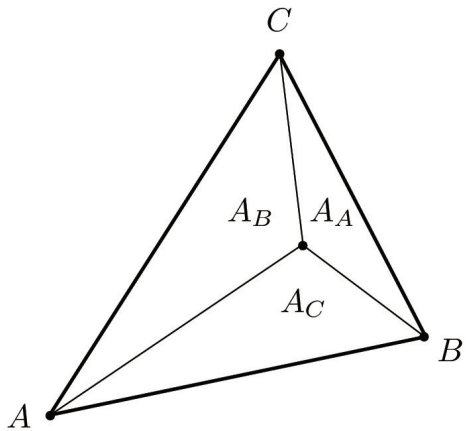
Visibility Buffer

Temporal Gradient

- $\delta_{i,\vec{j}} = f_i(G_{i,j}) - f_{i-1}(\overleftarrow{G}_{i,j})$
- Used defered shading with Blinn–Phong reflection model

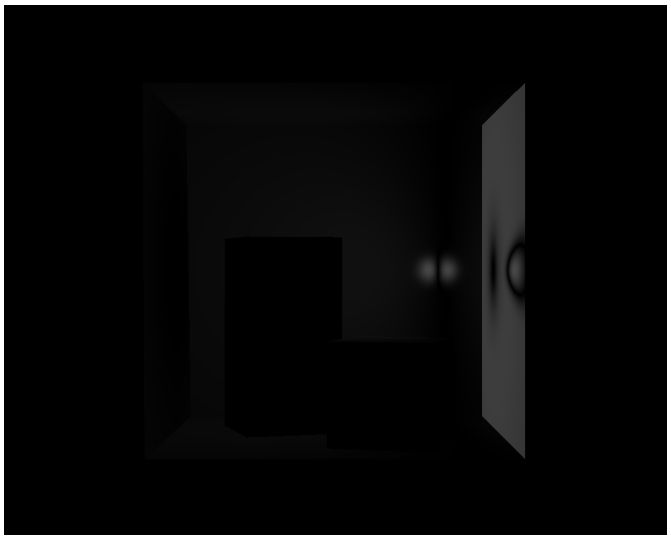
Barycentric Coordinates

Alternative geometric viewpoint — proportional areas



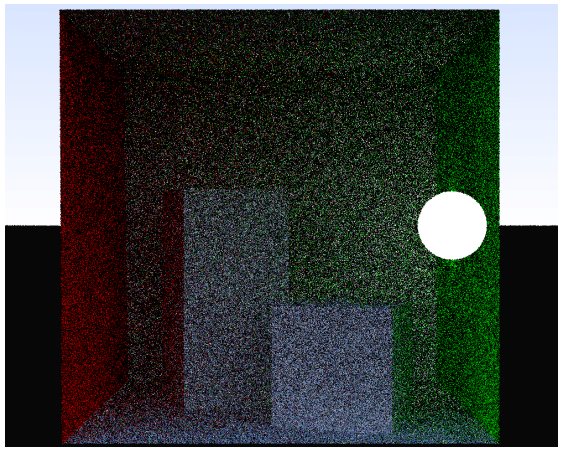
$$\alpha = \frac{A_A}{A_A + A_B + A_C}$$
$$\beta = \frac{A_B}{A_A + A_B + A_C}$$
$$\gamma = \frac{A_C}{A_A + A_B + A_C}$$

CS184/284A **Ren Ng**

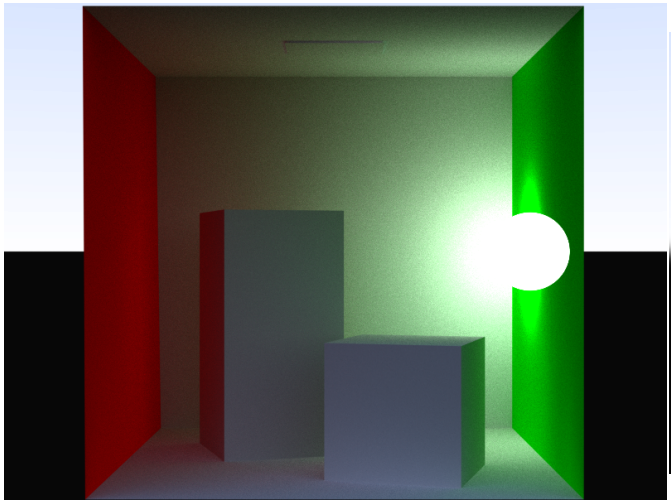


Temporal Gradient

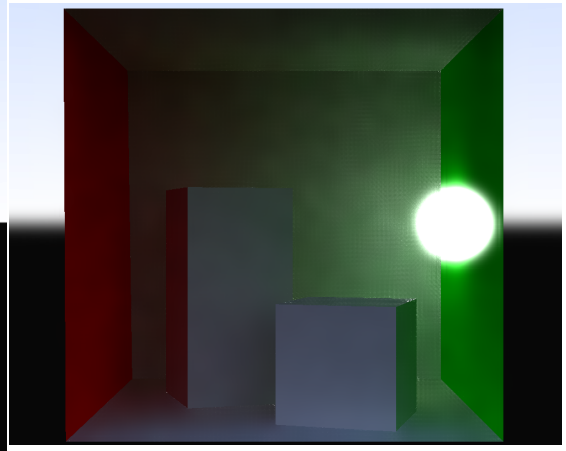
Results



Path traced image with 1 spp

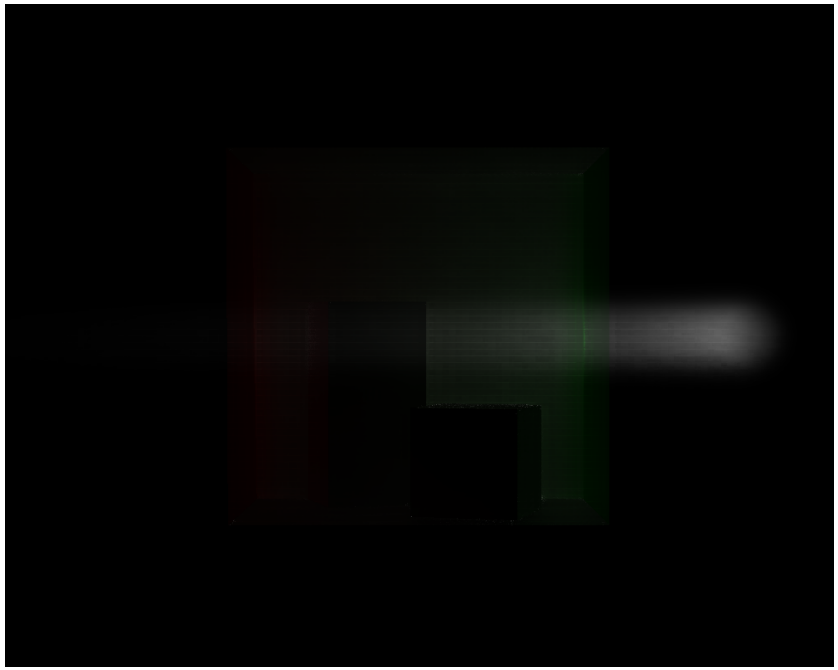


reference image / 8192 spp



Scene with A-SVGF filtering

Results



constant alpha per pixel

adaptive alpha per pixel

Thank you!