Real-time Path-Tracing Denoising

Vincent Higginson - June 2023

Supervisors: *Christophe De Vleeschouwer, Antoine Legrand*
Reader: *Benoît Macq*

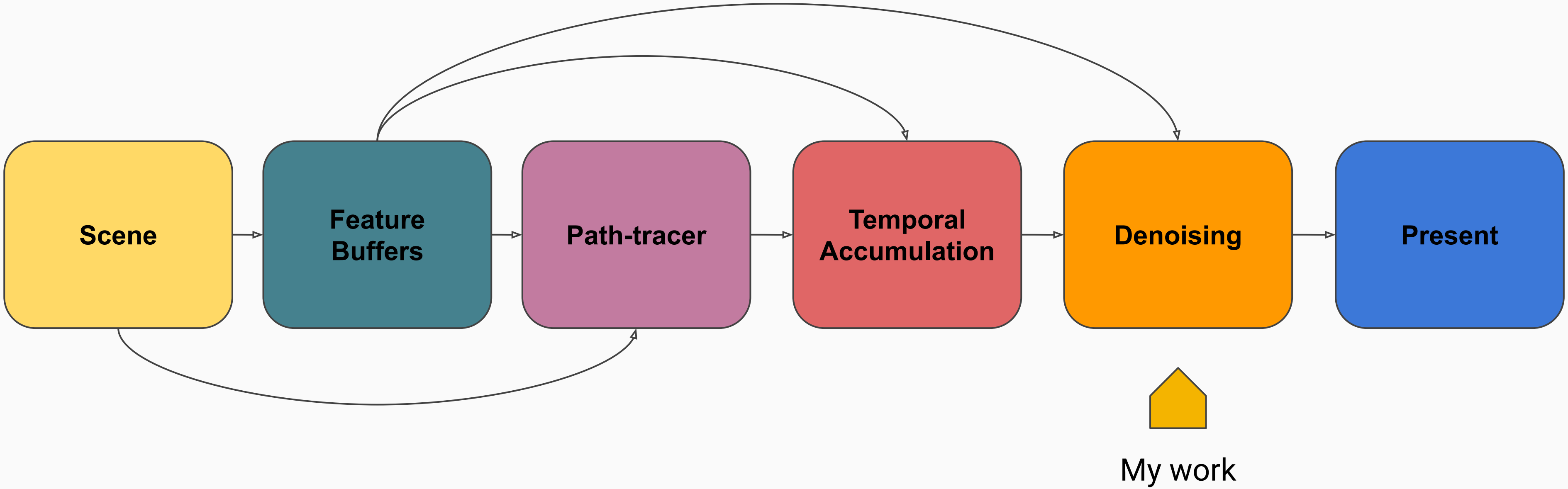
Objectives

Physically based path-tracing benefits
from a versatile, and powerful pipeline

We want to leverage it in a real-time renderer by

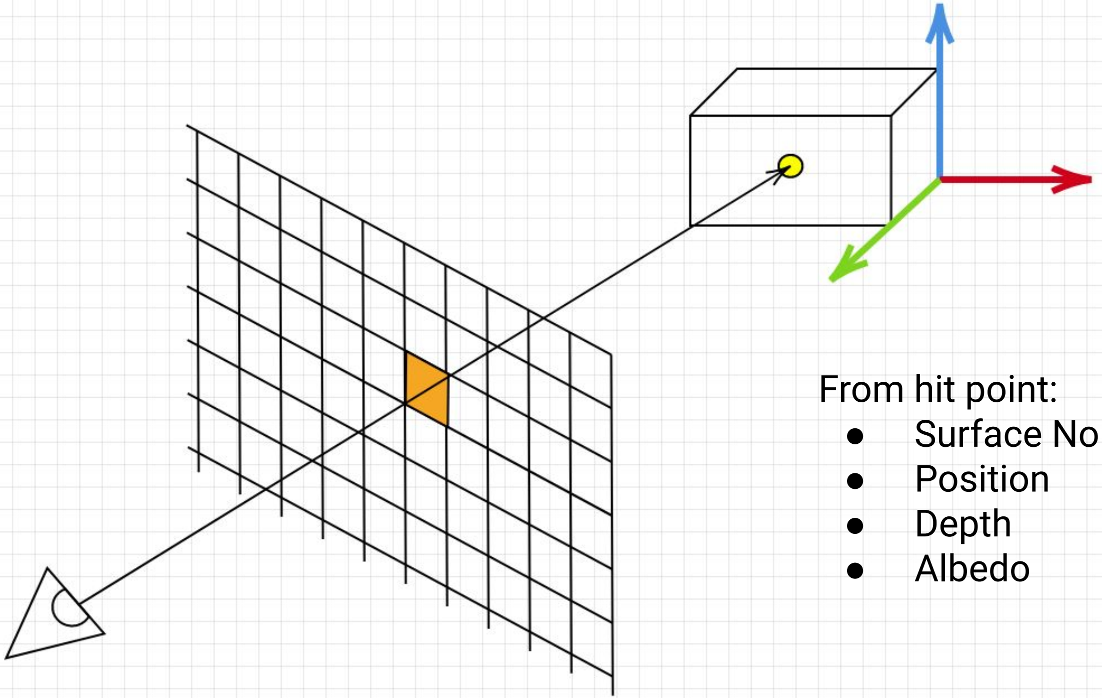
- Applying specific constraints
- Adding new ***reconstruction filters*** (purpose of this master thesis)

Rendering Pipeline



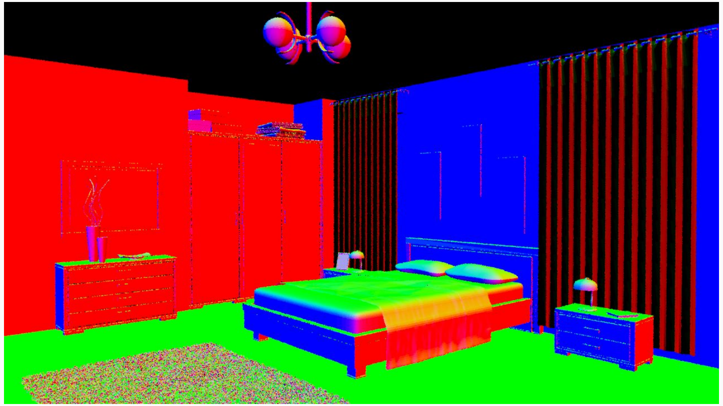
Feature buffers

Feature buffers—Origin



From hit point:
- Surface Normal
- Position
- Depth
- Albedo

Feature buffers—Data

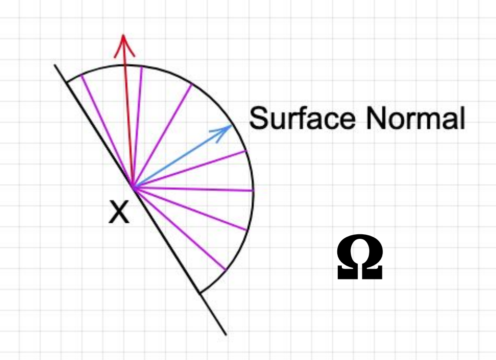


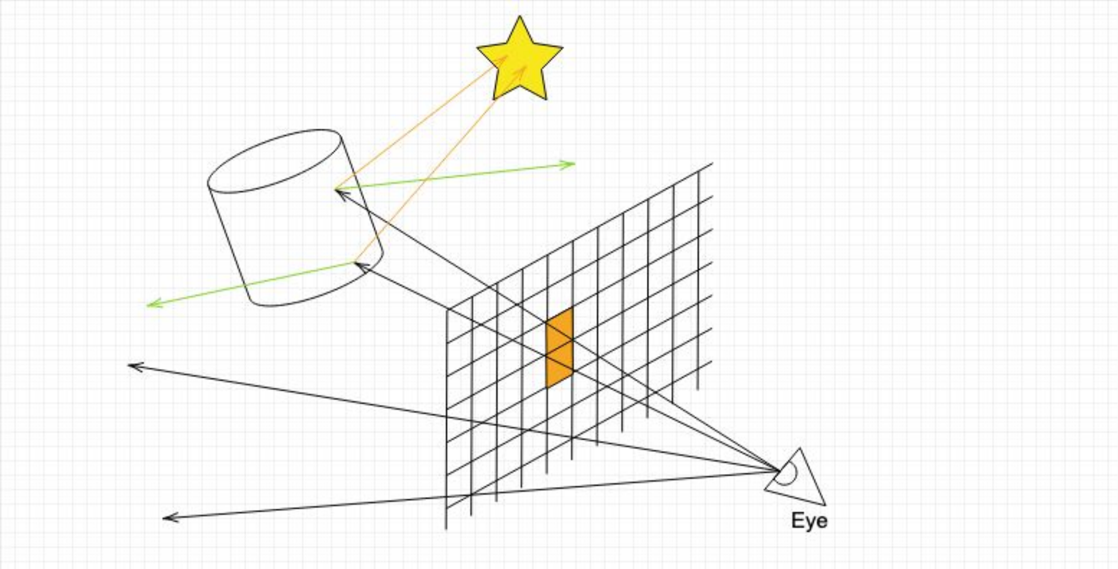
Path-Tracer

Path-Tracer

$$L_0(X,\widehat{\omega}_0) = L_e(X,\widehat{\omega}_0) + \left[\int_\Omega L_i(X,\widehat{\omega}_i) f(X,\widehat{\omega}_i,\widehat{\omega}_0) |\widehat{\omega}_i \cdot \widehat{n}|\, d\widehat{\omega}_i\right]$$

Rendering Equation - Kajiya 1986

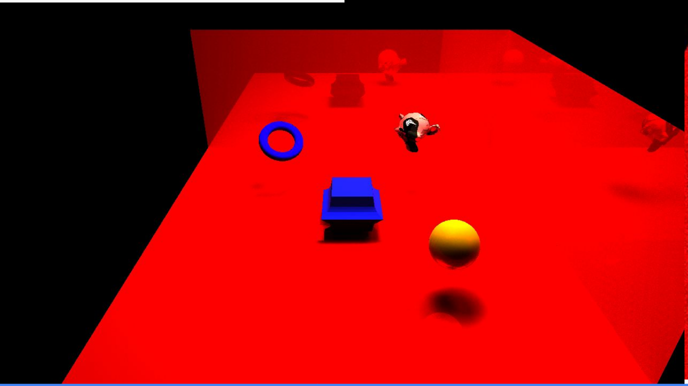


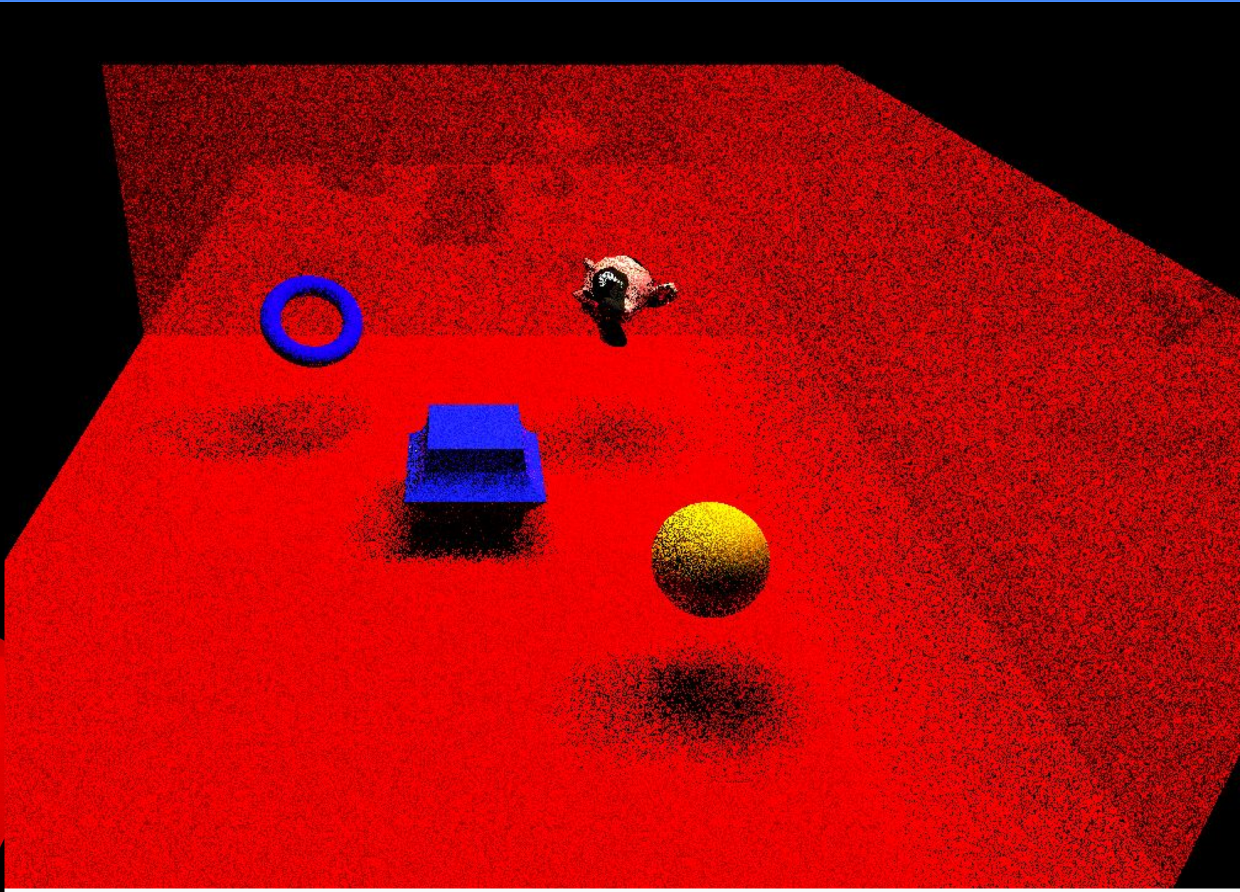


- Recursive
- Possibly infinite (depth, Ω)

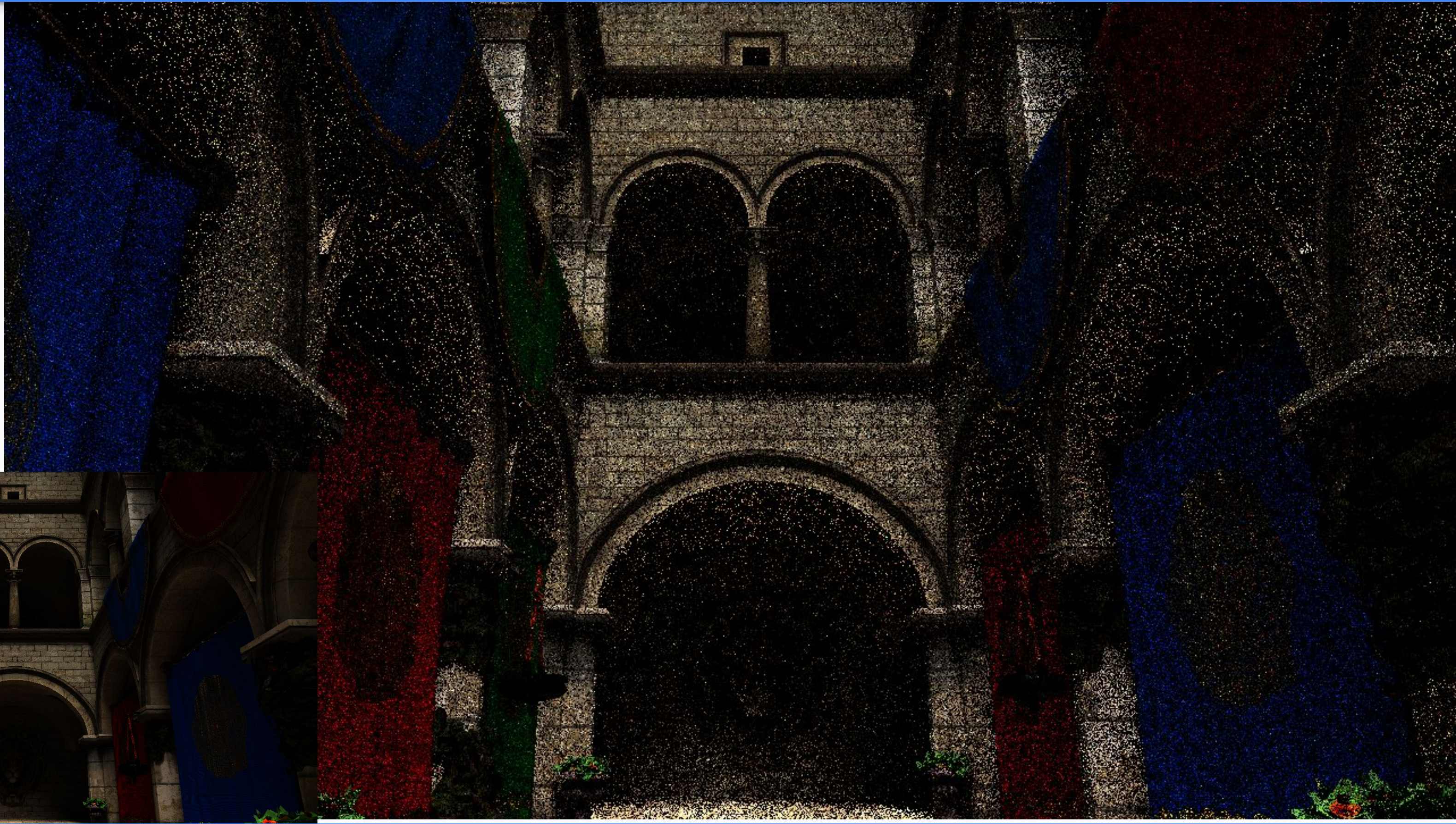
"Approximation" → Monte Carlo
When to stop?

Path-Tracer—Results with applied constraints

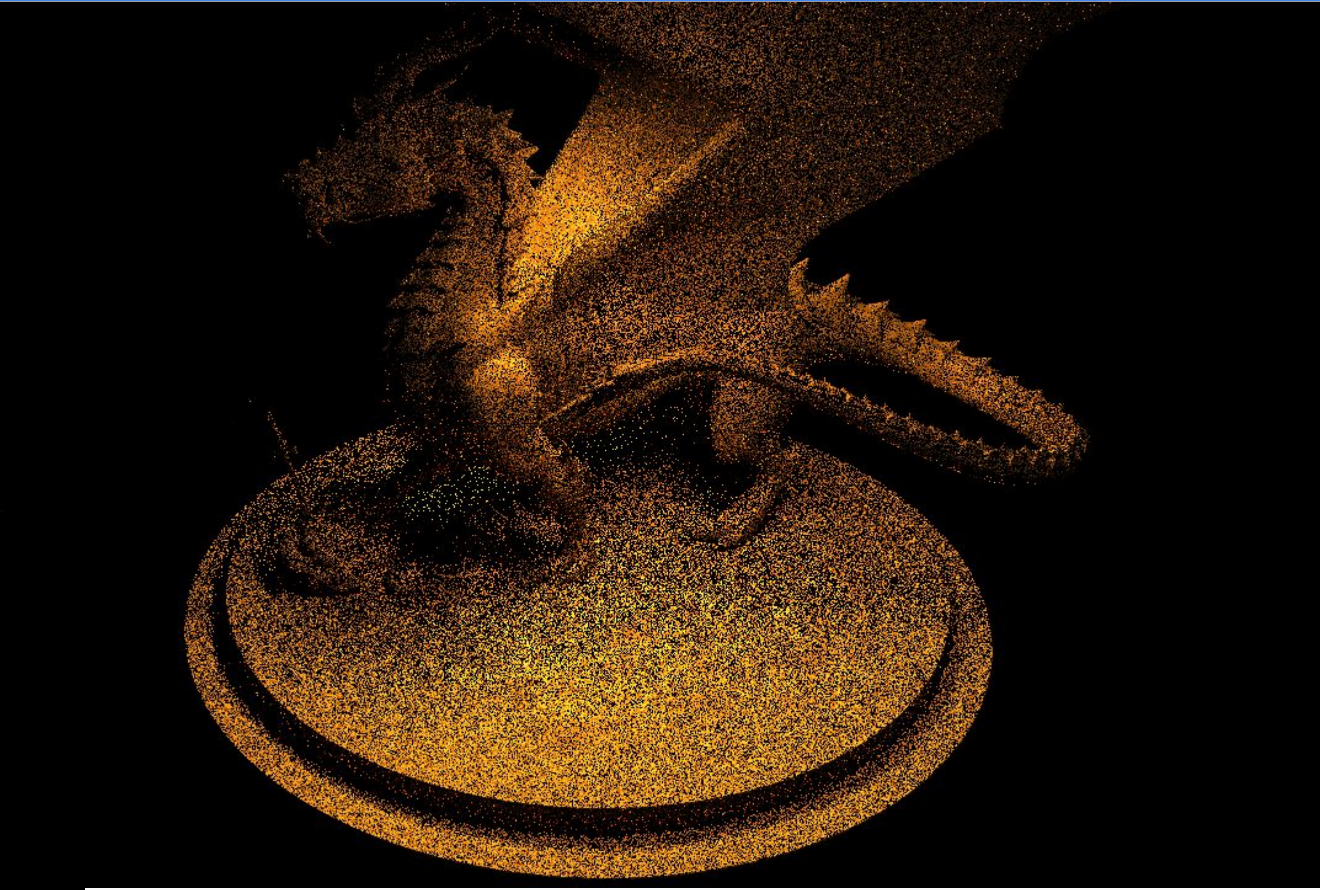




Path-Tracer—Results with applied constraints



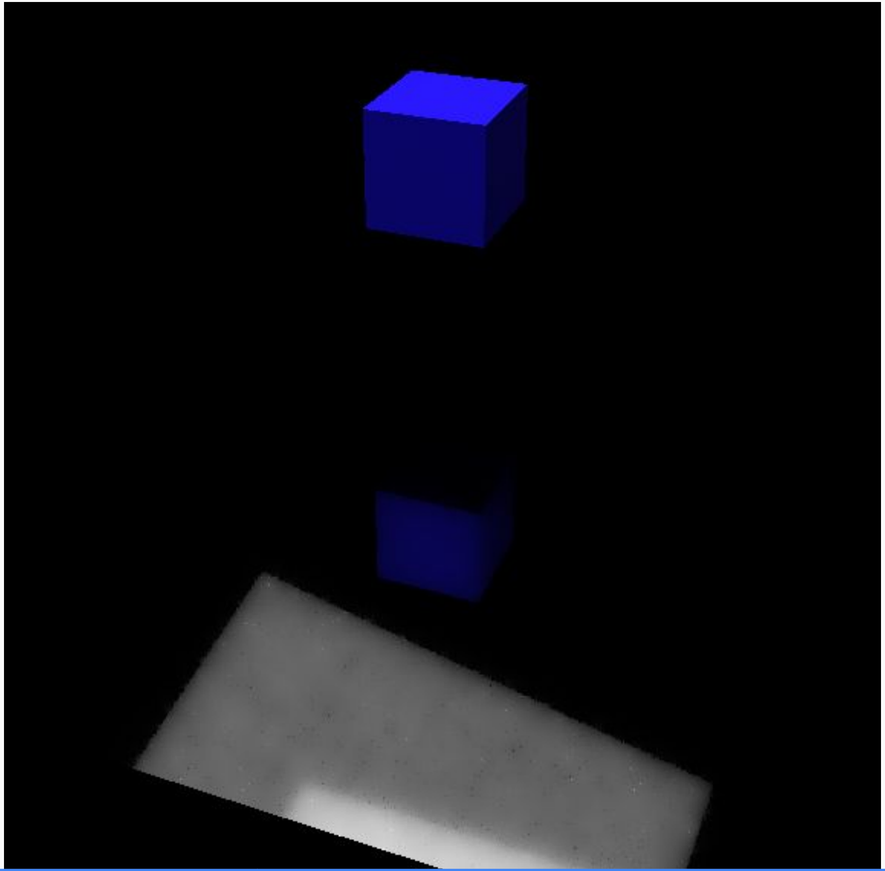
Path-Tracer—Results with applied constraints

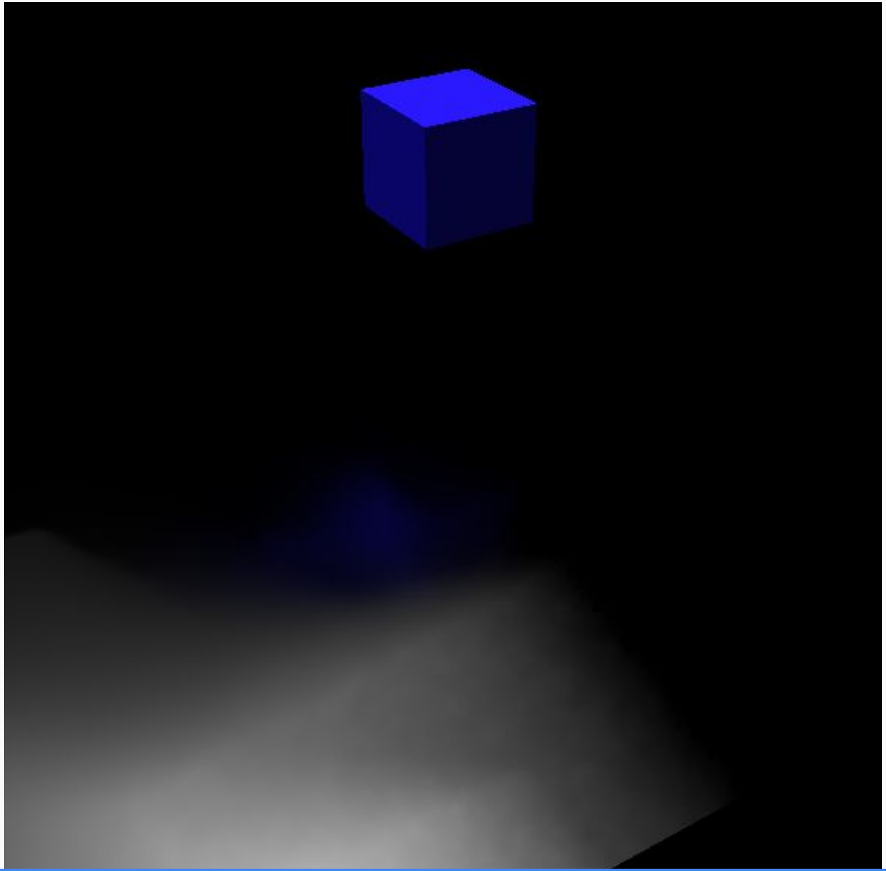


Temporal Accumulation

$$C'_i(p) = \alpha \times C_i(p) + (1 - \alpha) \times C'_{i-1}(\overleftarrow{p})$$

Temporal Accumulation—Problems

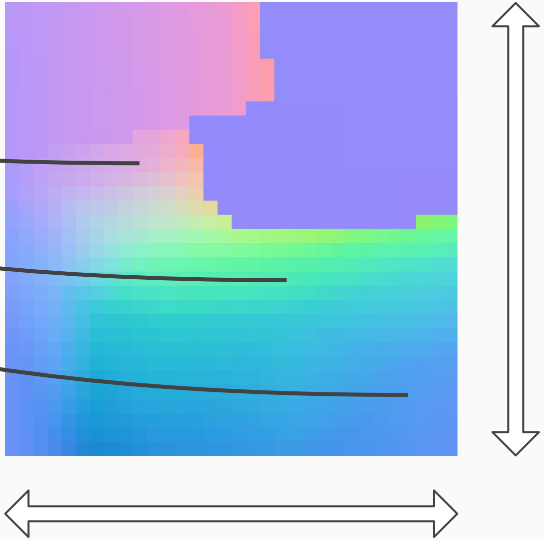




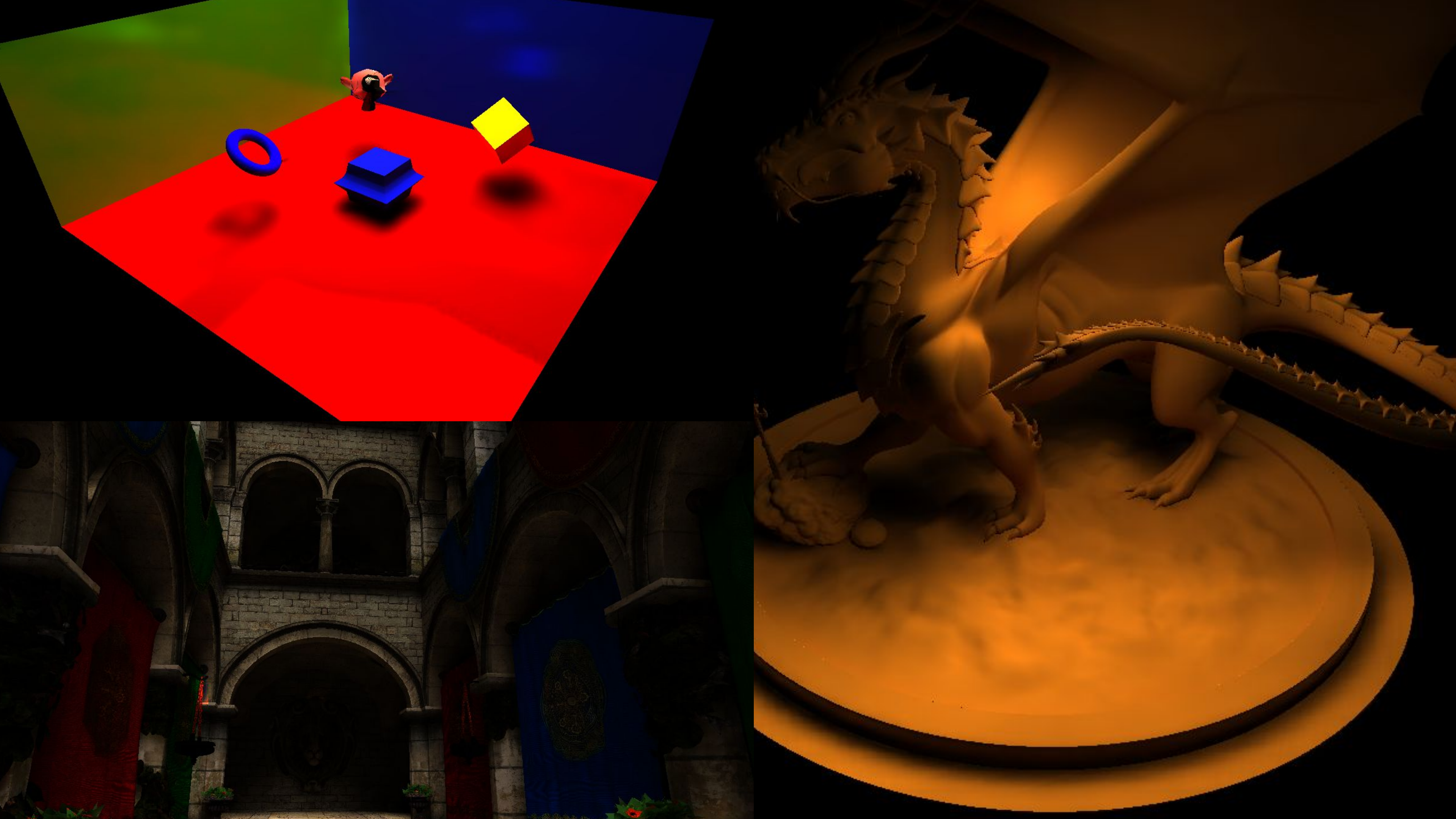
Denoising Methods

Special form of the least square problem

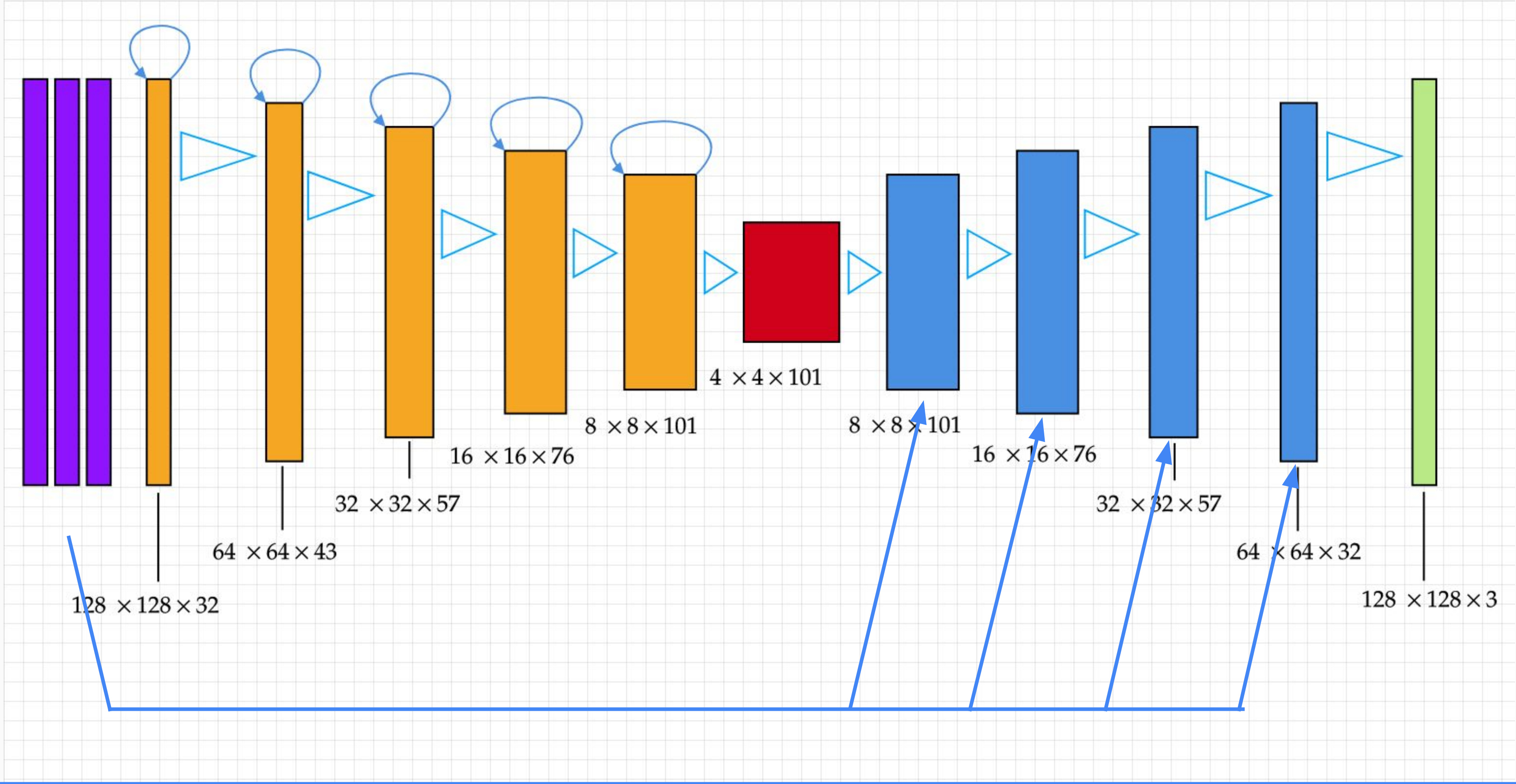
$$\tilde{T}\,T = \begin{bmatrix} \boxed{N_x(p_1)} & N_y(p_1) & N_z(p_1) & \cdots & {}^{(c)}(p_1) \\ \boxed{N_x(p_2)} & N_y(p_2) & N_z(p_2) & \cdots & {}^{(c)}(p_2) \\ \boxed{N_x(p_3)} & N_y(p_3) & N_z(p_3) & \cdots & {}^{(c)}(p_3) \\ \cdots & \cdots & \cdots & \cdots & \end{bmatrix}$$

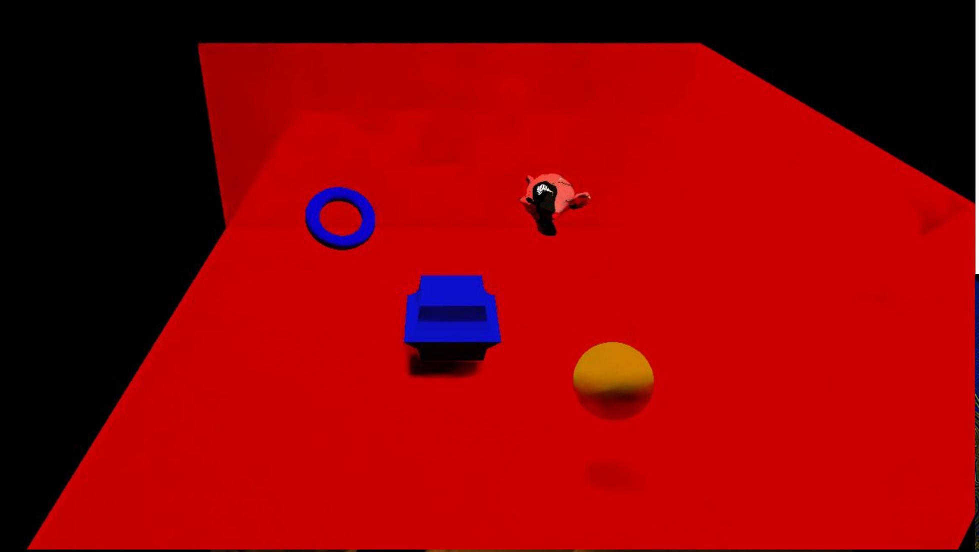


$$R\boxed{\widehat{\alpha}^{(c)}} = r^{(c)}$$

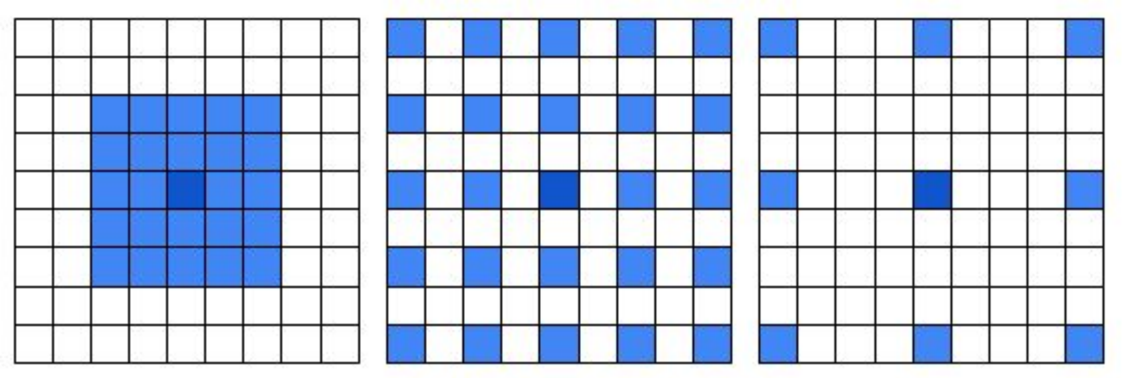


Machine-Learning with a Recurrent Auto-Encoder

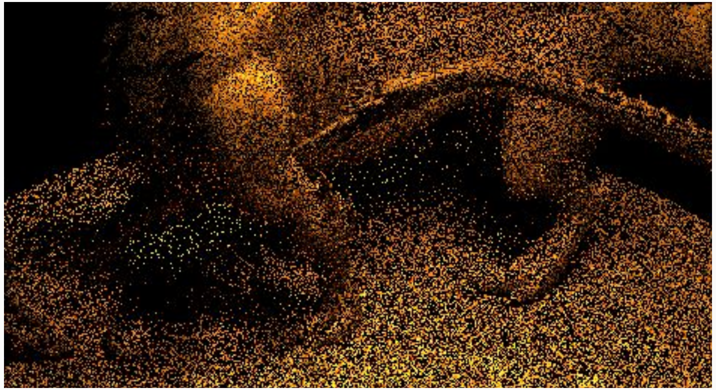




(Adaptive) Spatio-temporal Variance Guided Filtering

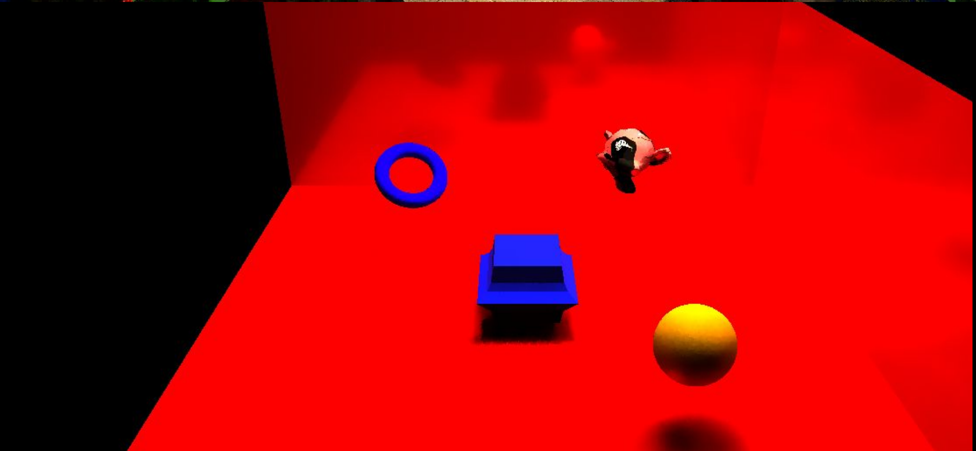


Weights computation with edge-avoiding functions
Related samples: 0.0
Unrelated samples: 1.0

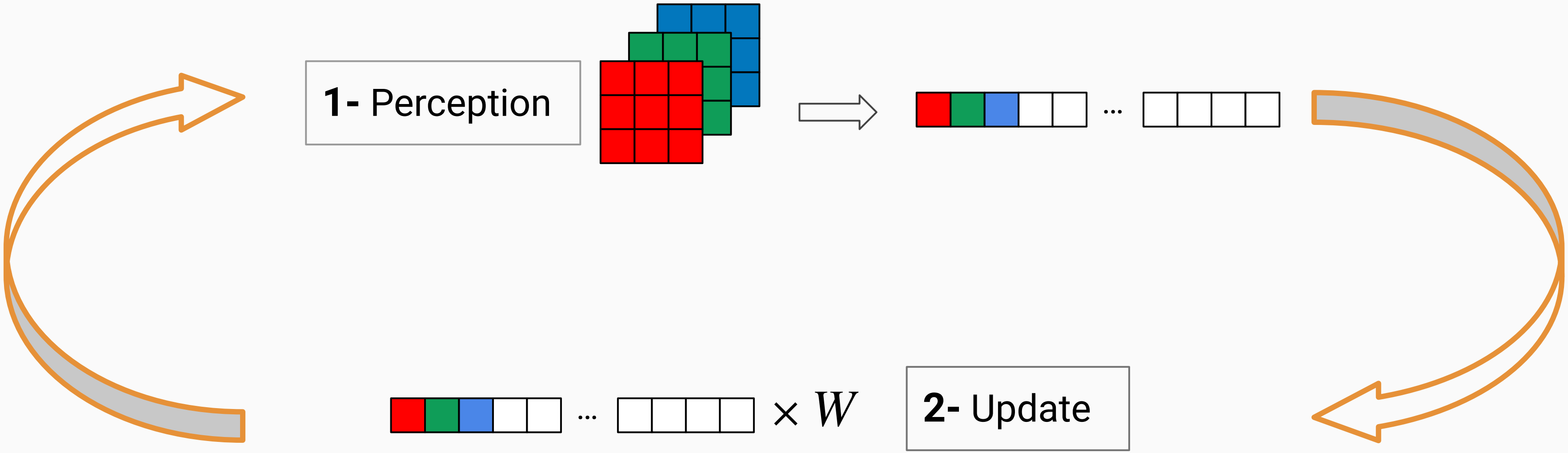


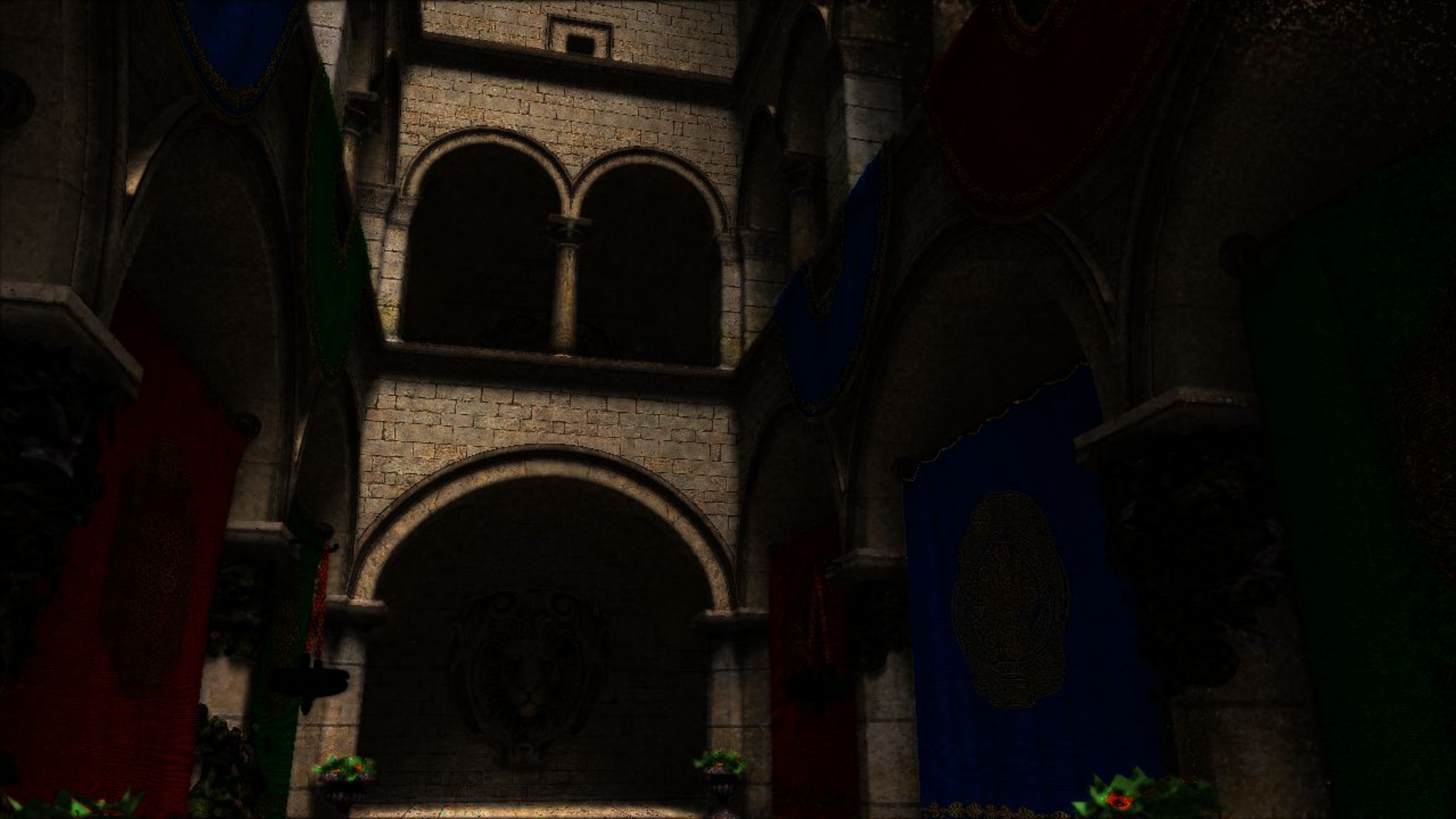
$$C'_i(p) = \boxed{\alpha} \times C_i(p) + (1 - \boxed{\alpha}) \times C'_{i-1}(\overleftarrow{p})$$

Was fixed, now *variable* per pixel
Based on the luminance variation



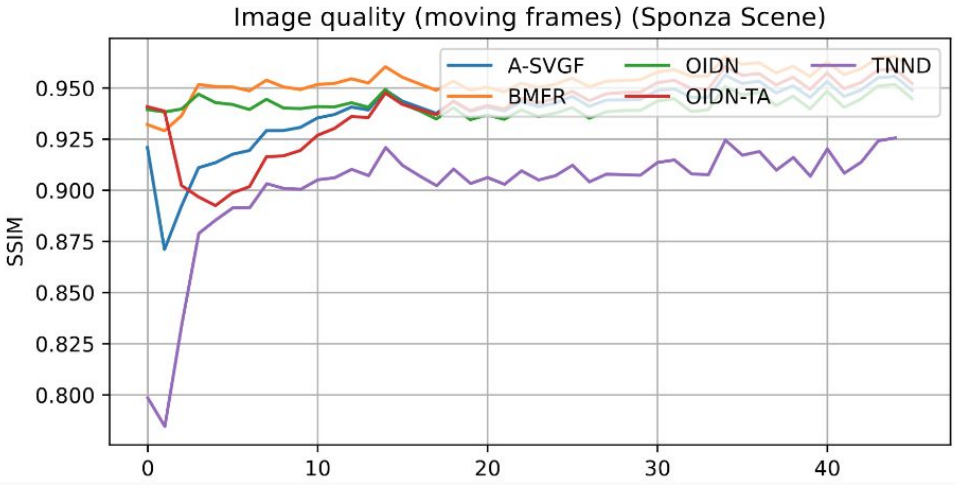
Tiny Neural Network Denoiser - Cellular Automata



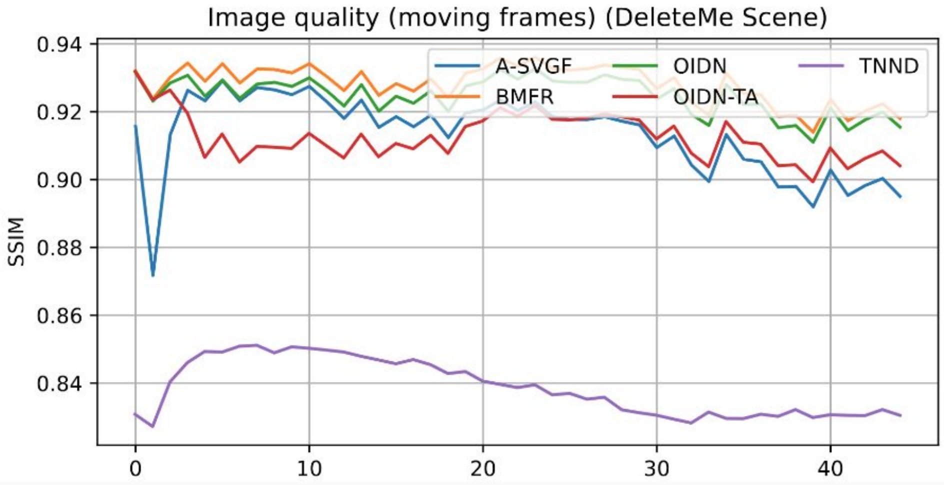


Comparison & Results

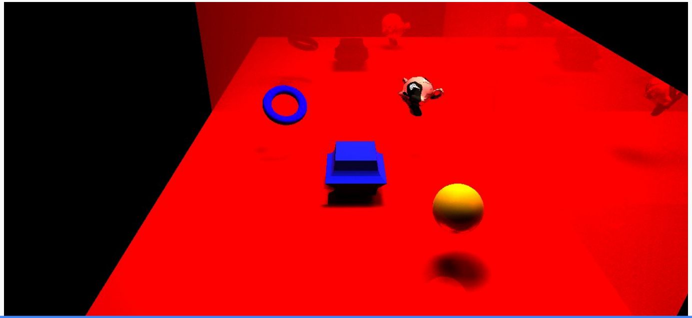
Comparison & Results

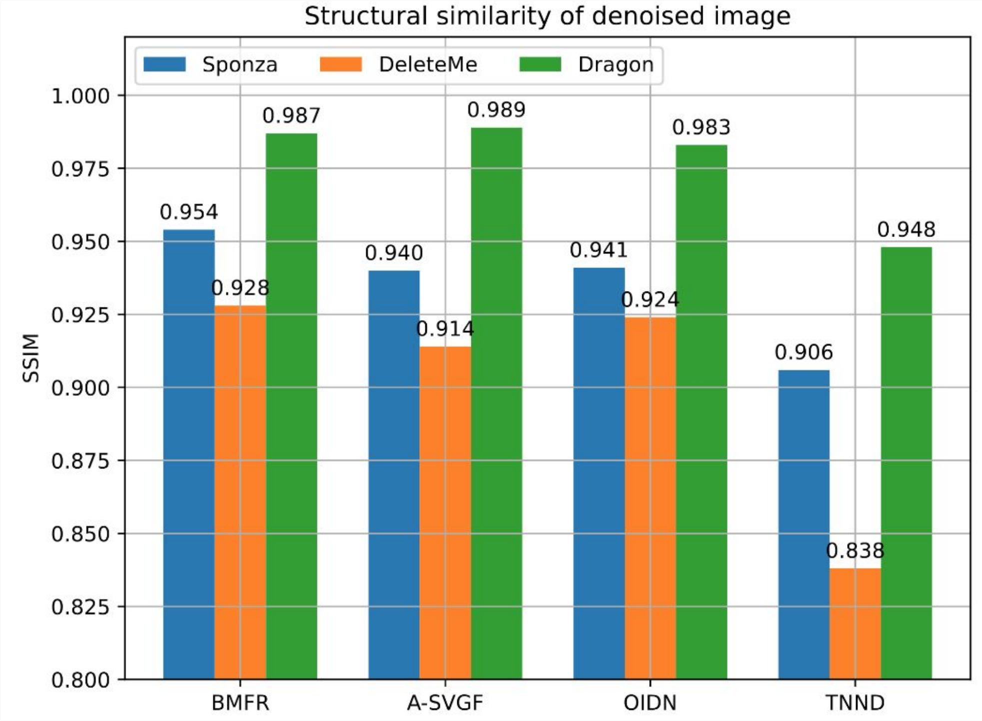


Sponza (no reflections)

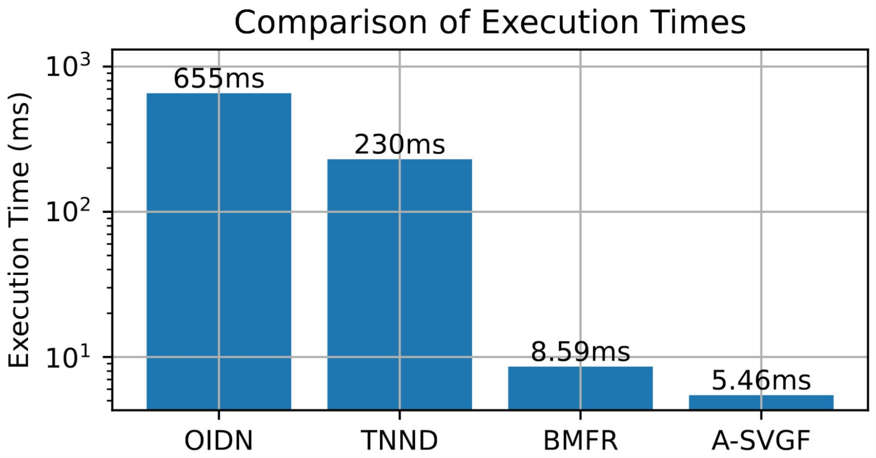


DeleteMe (reflections)





- Overall results very similar
- But computational complexity is not



Future Work

- TNND needs more work (GPU implementation, more diverse dataset, …)
- Some other techniques suggest moving the denoising further up in the pipeline
- Moving the whole pipeline on the GPU
- Implement compressed G-Buffer

Thank you!


BMFR