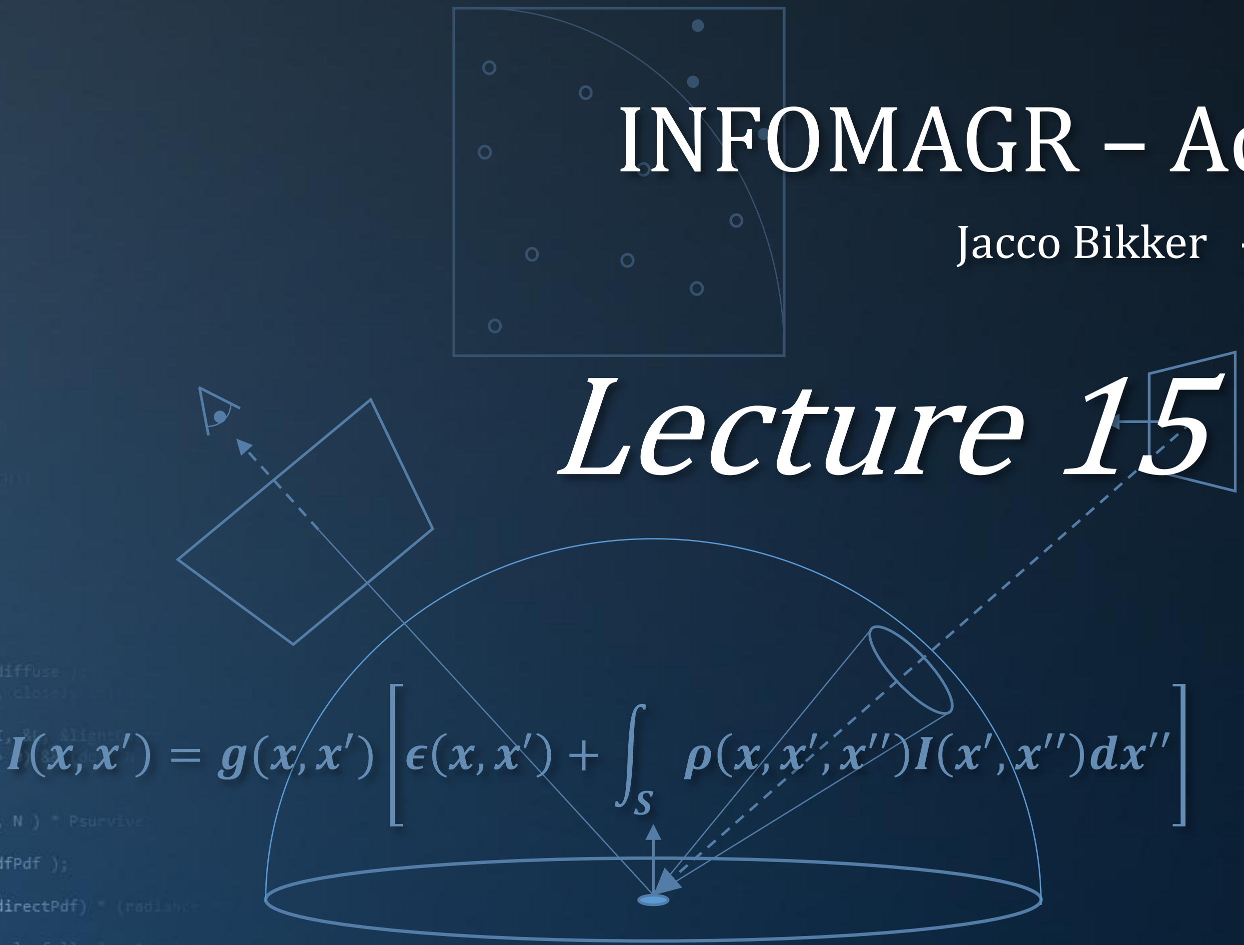
INFOMAGR – Advanced Graphics

Jacco Bikker - November 2021 - February 2022

Lecture 15 - “Filtering”



Welcome!

Today's Agenda:

- Introduction
- Ingredients
- Future Work

Previously in Advanced Graphics…

Last Time

Towards Noise-free Path Tracing

“Work smarter, not harder”: generate better samples / send rays where they matter.

Extreme case: ReSTIR, which spends a lot of effort on deciding where to send a shadow ray.

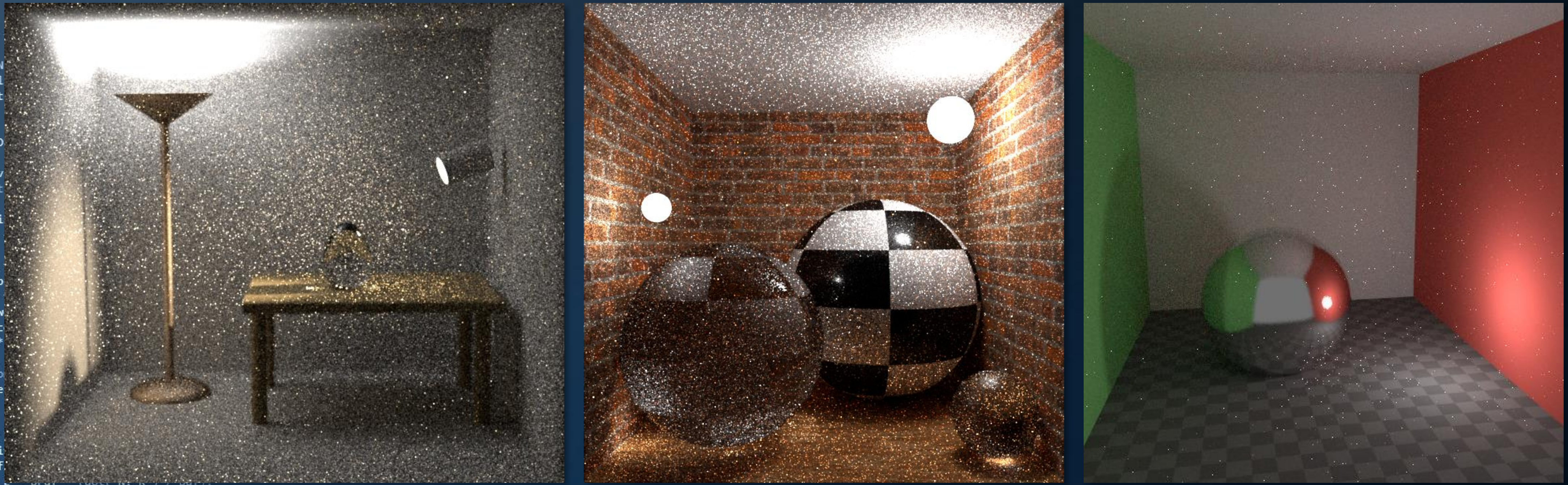
“Rearchitecting Spatiotemporal Resampling for Production”, Wyman & Panteleev, 2021.

Last Time

We tried everything

...But with an 8spp budget, it's still noisy.

- There is somewhat uniform noise left
- 'Fireflies' indicate presence of 'improbable paths'.

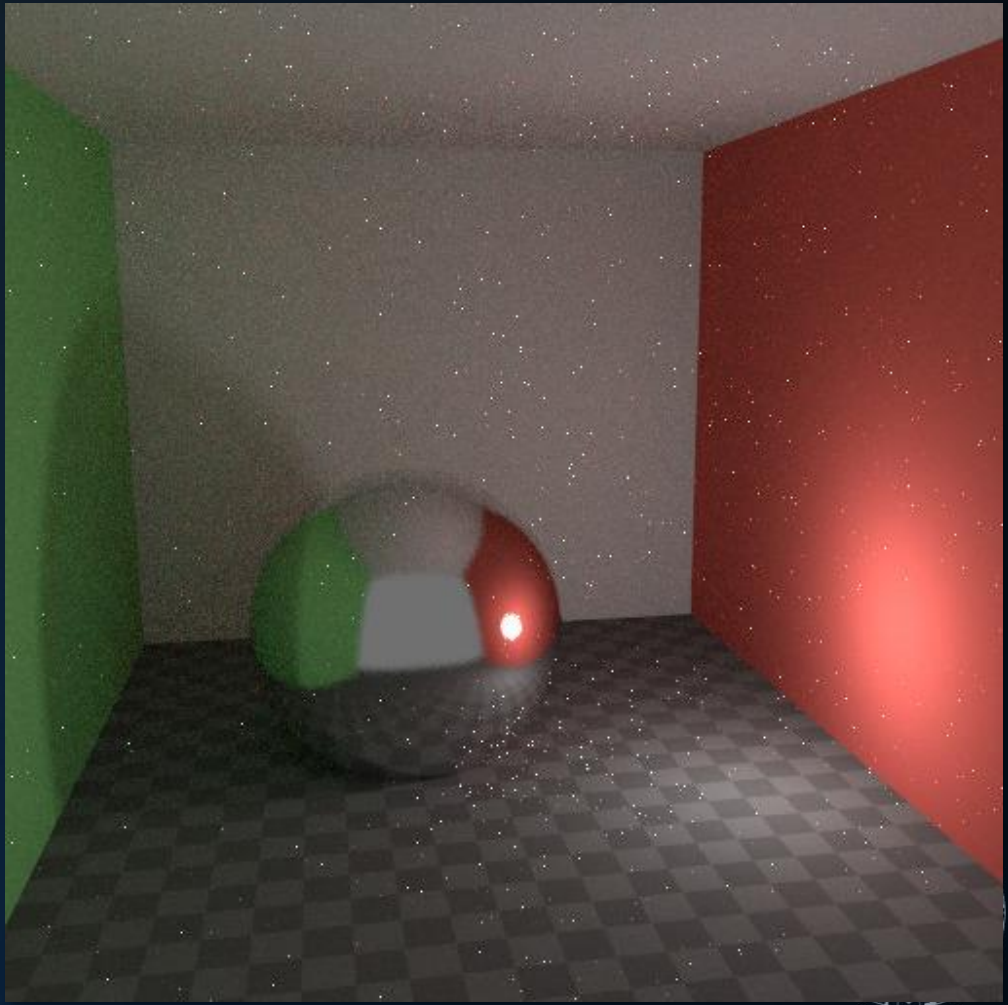


Last Time

We tried everything

…But with an 8spp budget, it’s still noisy.

- There is somewhat uniform noise left
- ‘Fireflies’ indicate presence of ‘improbable paths’.



Last Time

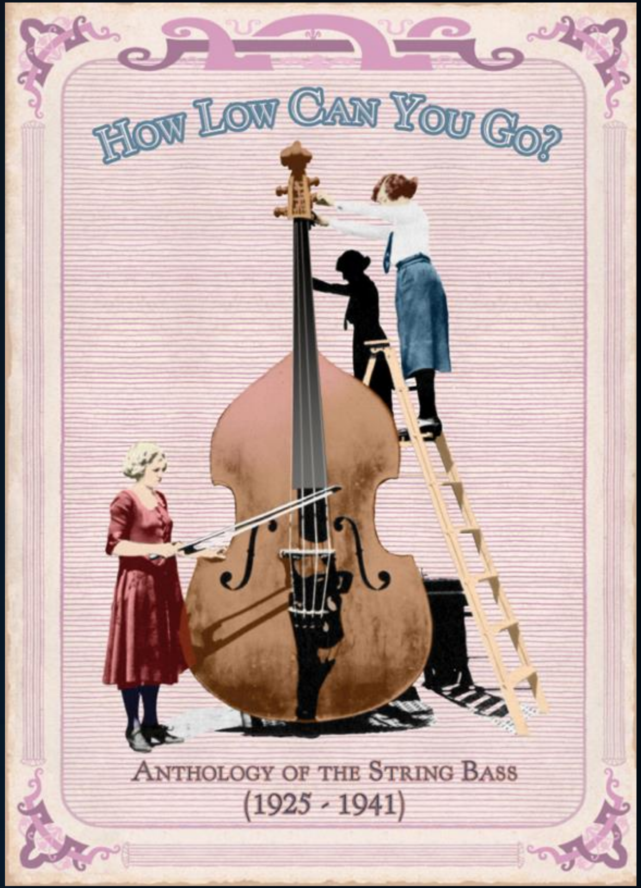
Suppressing Fireflies

“A firefly is easily recognized in the final image: it is a pixel with a value that differs significantly from its neighbors.”

- Is this always true?
- How to fix it?
- Is that still correct?

Firefly suppression introduces bias in our estimator.

- Spread out the removed energy over the image / neighborhood
- Just wait it out (additional samples will improve the average)
- Do some adaptive sampling (detect high variance)
- **Just accept it.**



Last Time

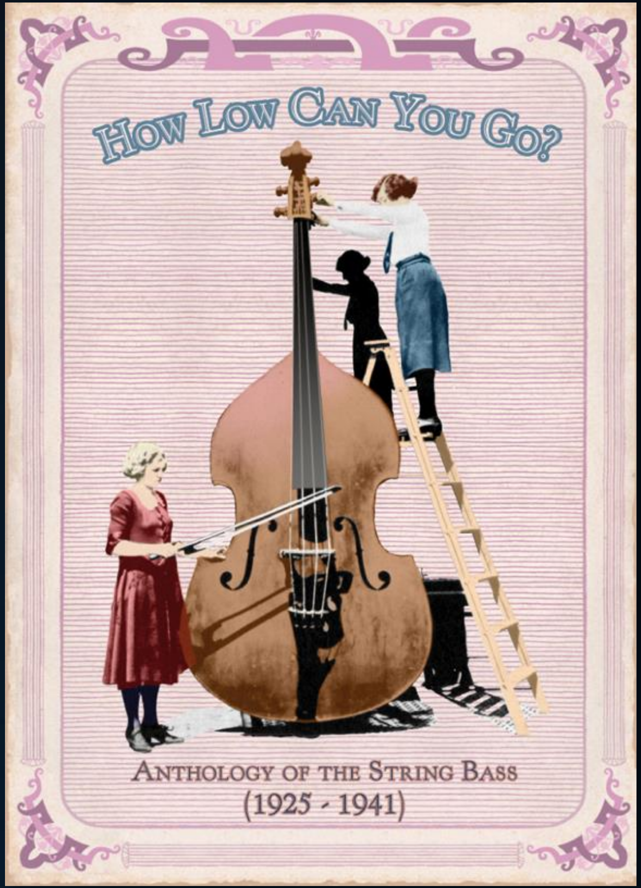
Suppressing Fireflies

"A firefly is easily recognized in the final image: it is a pixel with a value that differs significantly from its neighbors."

Better approach: *clamp*.*

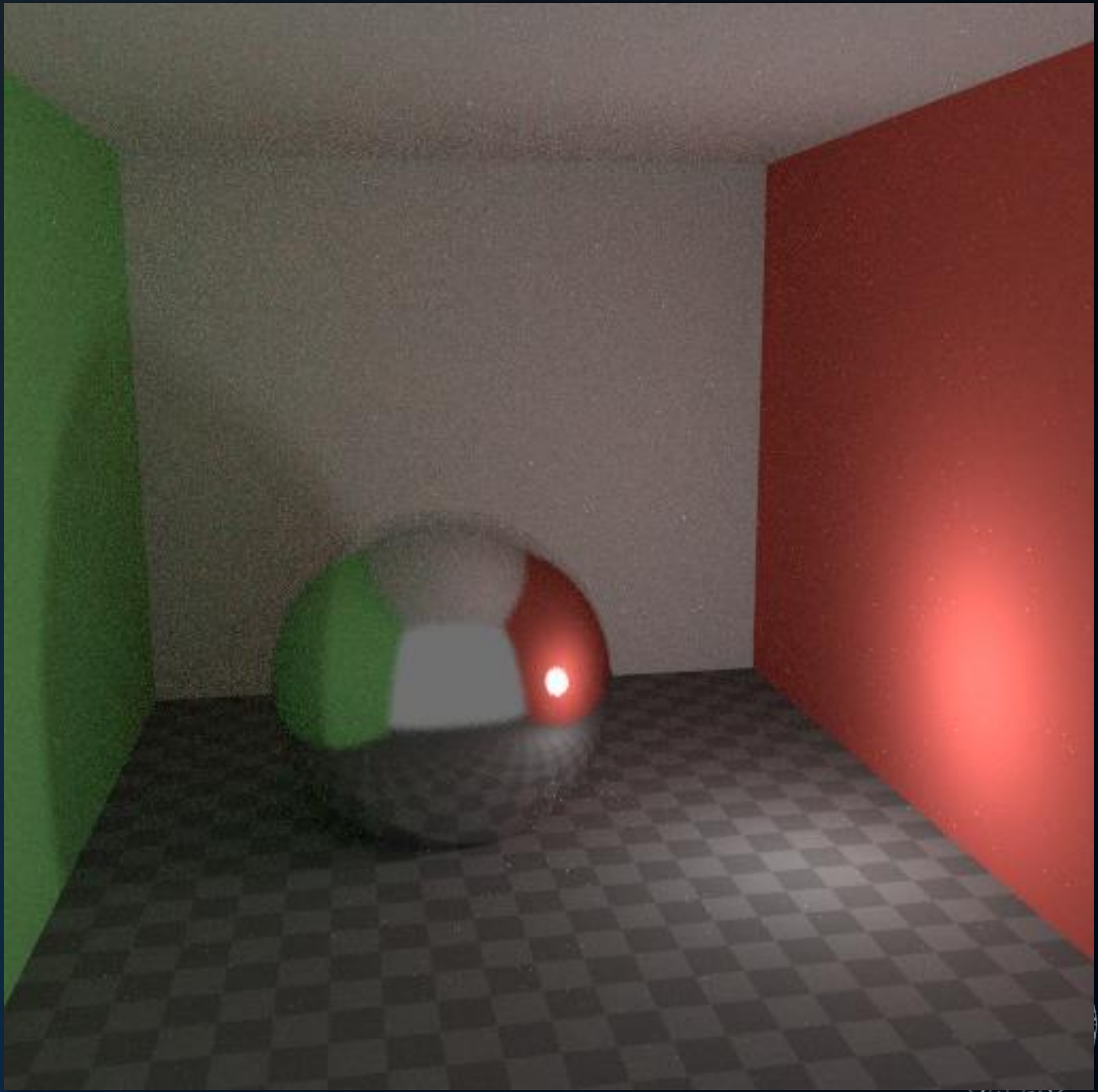
e.g., in Lighthouse 2:

```
#define CLAMPINTENSITY( E ) \
   if (dot( E, E ) > 25) E = 5 * normalize( E );
```



*: The Iray Light Transport Simulation and Rendering System, Section 5.5. Keller et al., 2017.

Last Time



Today's Agenda:

- Noise
- Ingredients
- Future Work

Ingredients

Reducing the Problem - Filtering

Core idea:

Exploit the fact that illumination is typically low-frequent:
Nearby pixels tend to converge to similar values, so we should be able to use information gathered for one pixel to improve the estimate of the next.

Essentially, we are increasing the number of samples per pixel, by including the neighbors.

Note:

Unless neighboring pixels actually converge to the same value, filtering introduces bias.

Filtering thus trades variance for bias.

Ingredients

1 kernels

Filter kernels

For the actual filtering, we apply a kernel.

```
Pixel FilteredValue( i_x, i_y, halfWidth )
   sum = 0
   summedWeight = 0
   for j_x = i_x - halfWidth to i_x + halfWidth
      for j_y = i_y - halfWidth to i_y + halfWidth
         sum += ReadPixel( j_x, j_y ) * weight( j_x, j_y )
         summedWeight += weight( j_x, j_y )
   return sum / summedWeight
```

$$\hat{c}_i = \frac{\sum_{j \in \mathcal{N}_i} c_j w(i,j)}{\sum_{j \in \mathcal{N}_i} w(i,j)}$$

Ingredients

1 kernels

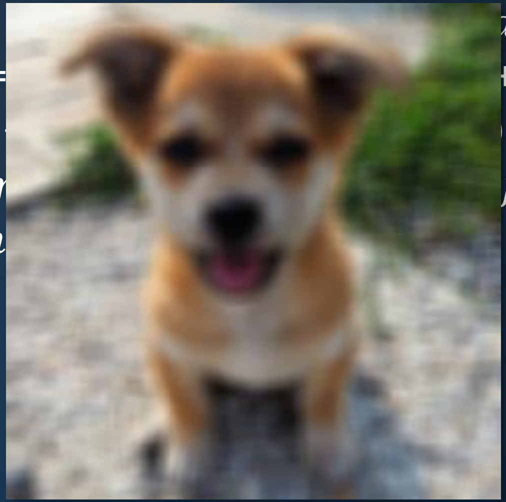
Filter kernels

For the actual filtering, we apply a kernel.

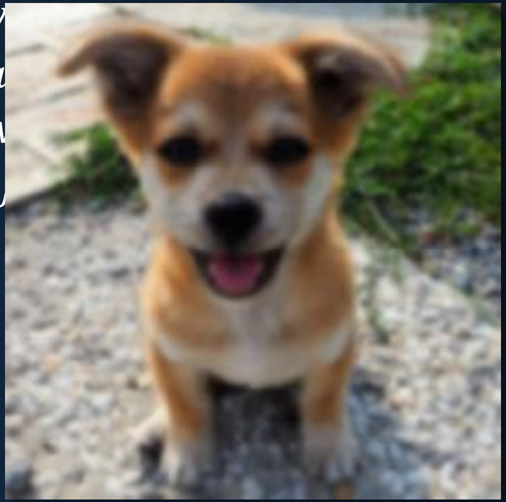
$$
\begin{array}{l}
\texttt{Pixel FilteredValue(}\ i_x,\ i_y,\ halfWidth\ \texttt{)} \\
\quad sum = 0 \\
\quad summedWeight = 0
\end{array}
$$

$$\hat{c}_i = \frac{\sum_{j \in \mathcal{N}_i} c_j w(i,j)}{\sum_{j \in \mathcal{N}_i} w(i,j)}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

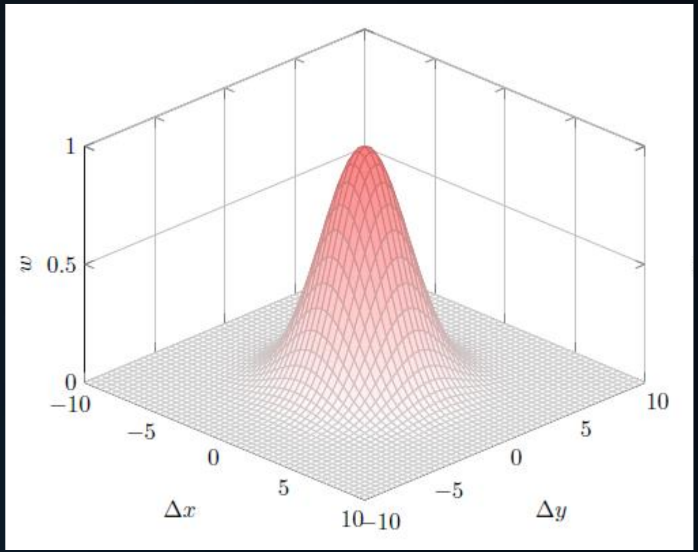


$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



$$\frac{1}{22}\begin{bmatrix} 1 & 3 & 1 \\ 3 & 6 & 3 \\ 1 & 3 & 1 \end{bmatrix}$$

Ingredients



1 kernels

Filter kernels

For the actual filtering, we apply a kernel.

```
Pixel FilteredValue( i_x, i_y, halfWidth )
    sum = 0
    summedWeight = 0
    for j_x = i_x - halfWidth to i_x + halfWidth
        for j_y = i_y - halfWidth to i_y + halfWidth
            sum += ReadPixel( j_x, j_y ) * weight( j_x, j_y )
            summedWeight += weight( j_x, j_y )
    return sum / summedWeight
```

Here, **weight** or w is the weight function. We could simply use the Gaussian kernel:

$$w(i,j) = \exp\left(\frac{-\|p_i - p_j\|^2}{2\sigma_d^2}\right),$$

where p_i and p_j are screen space positions and σ_d is the spatial standard deviation of the Gaussian kernel.

Ingredients

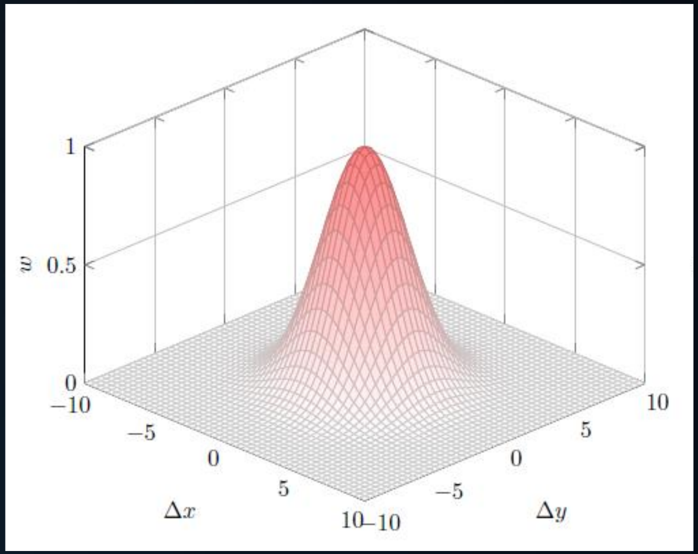
1 kernels

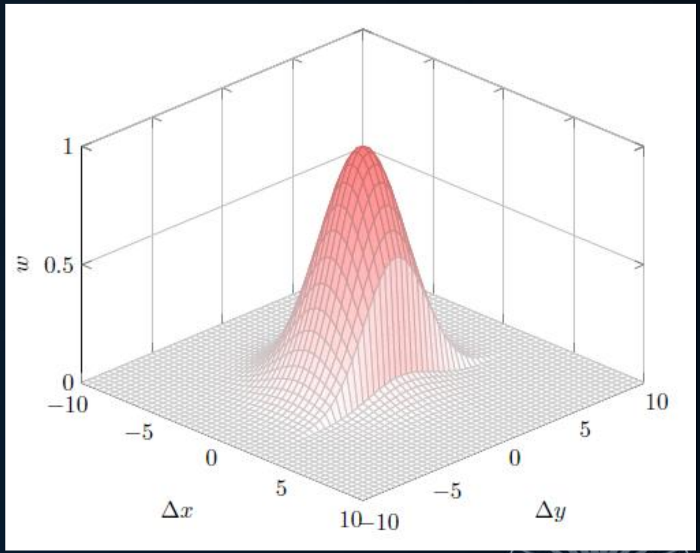
Filter kernels

A Gaussian filter (as well as other low-pass filters) blurs out high frequency details.

We can improve on this using a non-linear bilateral filter*.

$$w(i,j) = \exp\left(\frac{-\left\|p_i - p_j\right\|^2}{2\sigma_d^2}\right) \times \exp\left(\frac{-\left\|c_i - c_j\right\|^2}{2\sigma_r^2}\right)$$





*: Tomasi & Manduchi, Bilateral filtering for gray and color images. ICCV ’98.

Ingredients

1 kernels

Filter kernels

The bilateral filter takes the color of nearby pixels into account. We can take this further, by taking an arbitrary set of features into account.

The cross bilateral filter*:

$$w(i,j) = \exp\left(\frac{-\left\|p_i - p_j\right\|^2}{2\sigma_d^2}\right) \times \prod_{k=1}^{K} \exp\left(\frac{-\left\|f_{k,i} - f_{k,j}\right\|^2}{2\sigma_k^2}\right)$$

Here, $f_{k,i}$ is the k'th feature vector at pixel i and σ_k is the bandwidth parameter for feature k.

Note that we can use noise-free features to smooth noisy features.
Example of a low-noise feature: normals at the primary intersection point.
Example of a noisy feature: indirect illumination at the primary intersection point.

*: Eisemann & Durand. Flash photography enhancement via intrinsic relighting. ACM Trans. Graph. 23, 3 (Aug. 2004).

Ingredients

1 kernels

Filter kernels, digest

Filtering adds samples to a pixel by ‘borrowing’ them from neighbors.
Filtering trades variance for bias.

We can improve the quality of the borrowed samples using a weight:

- Further away = less relevant
- Different normal, different material, … = less relevant

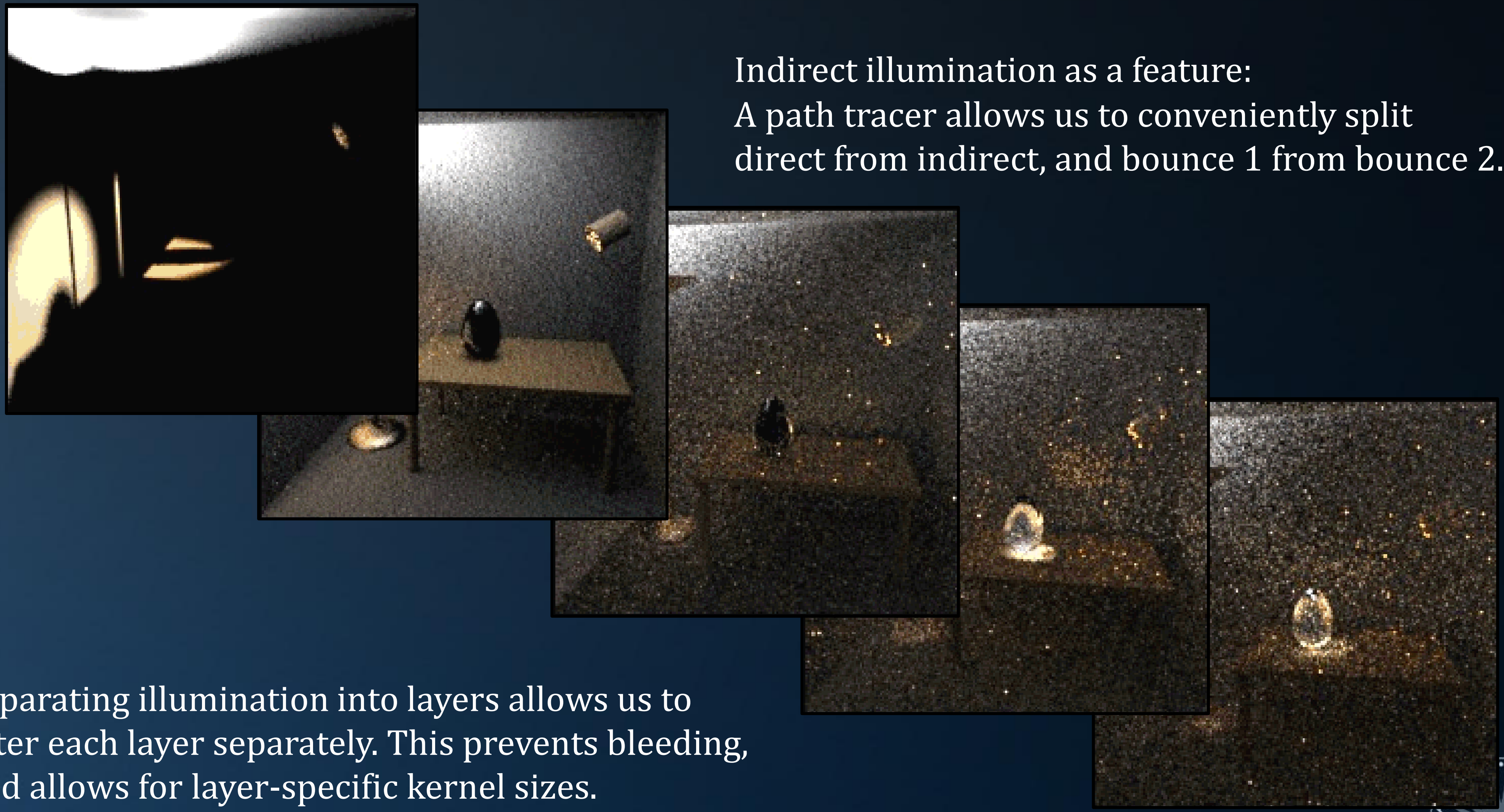
Some considerations:

- Should we take accumulated or individual samples from neighbors?
- Depth of field and AA seriously affect our options.

Ingredients

1 kernels

2 split



Indirect illumination as a feature:
A path tracer allows us to conveniently split direct from indirect, and bounce 1 from bounce 2.

Separating illumination into layers allows us to filter each layer separately. This prevents bleeding, and allows for layer-specific kernel sizes.

We can also separate albedo from illumination.

Ingredients

1 kernels

2 split

Separating albedo from illumination

Adding this separation to an existing renderer:

- store albedo at the primary intersection (simple material property);
- at the end of the pipeline: illumination = sample / max(epsilon, albedo).

Ingredients

1 kernels

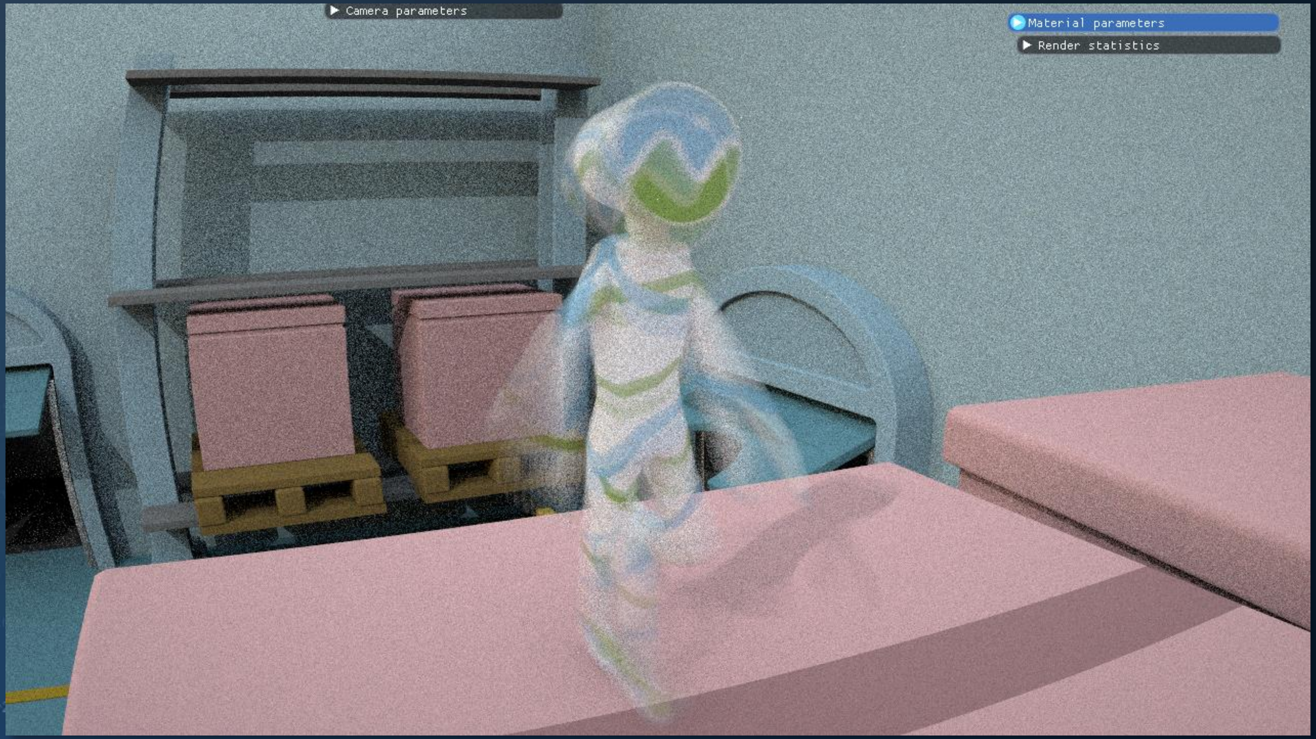
2 split

3 temporal

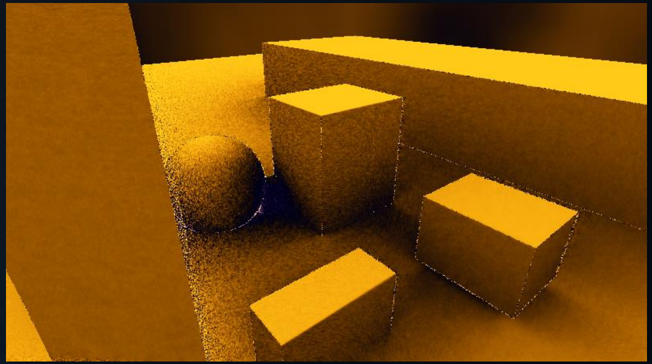
Reprojection

Core idea:

In an animation, samples taken for the previous frame are meaningful for the current frame. *We can supply the filter with more data by looking back in time.*



Ingredients



https://www.shadertoy.com/view/ldtGWl

1 kernels
2 split
3 temporal

Reprojection

Core idea:

In an animation, samples taken for the previous frame are meaningful for the current frame. *We can supply the filter with more data by looking back in time.*

Problem: in an animation, the camera and/or the geometry moves. We need to find the location of a pixel in the previous frame(s).

Solution: use the camera matrices.

$$M_{4x4}\begin{pmatrix}x_{world}\\y_{world}\\z_{world}\\1\end{pmatrix}=\begin{pmatrix}x_{screen}\\y_{screen}\\z_{screen}\\1\end{pmatrix} \rightarrow M_{4x4}^{-1}\begin{pmatrix}x_{screen}\\y_{screen}\\z_{screen}\\1\end{pmatrix}=\begin{pmatrix}x_{world}\\y_{world}\\z_{world}\\1\end{pmatrix}$$

(finally, apply the matrix of the previous frame to obtain the screen location in the previous frame.)

Ingredients

1 kernels

2 split

3 temporal

Reprojection

Reprojection using camera matrices:

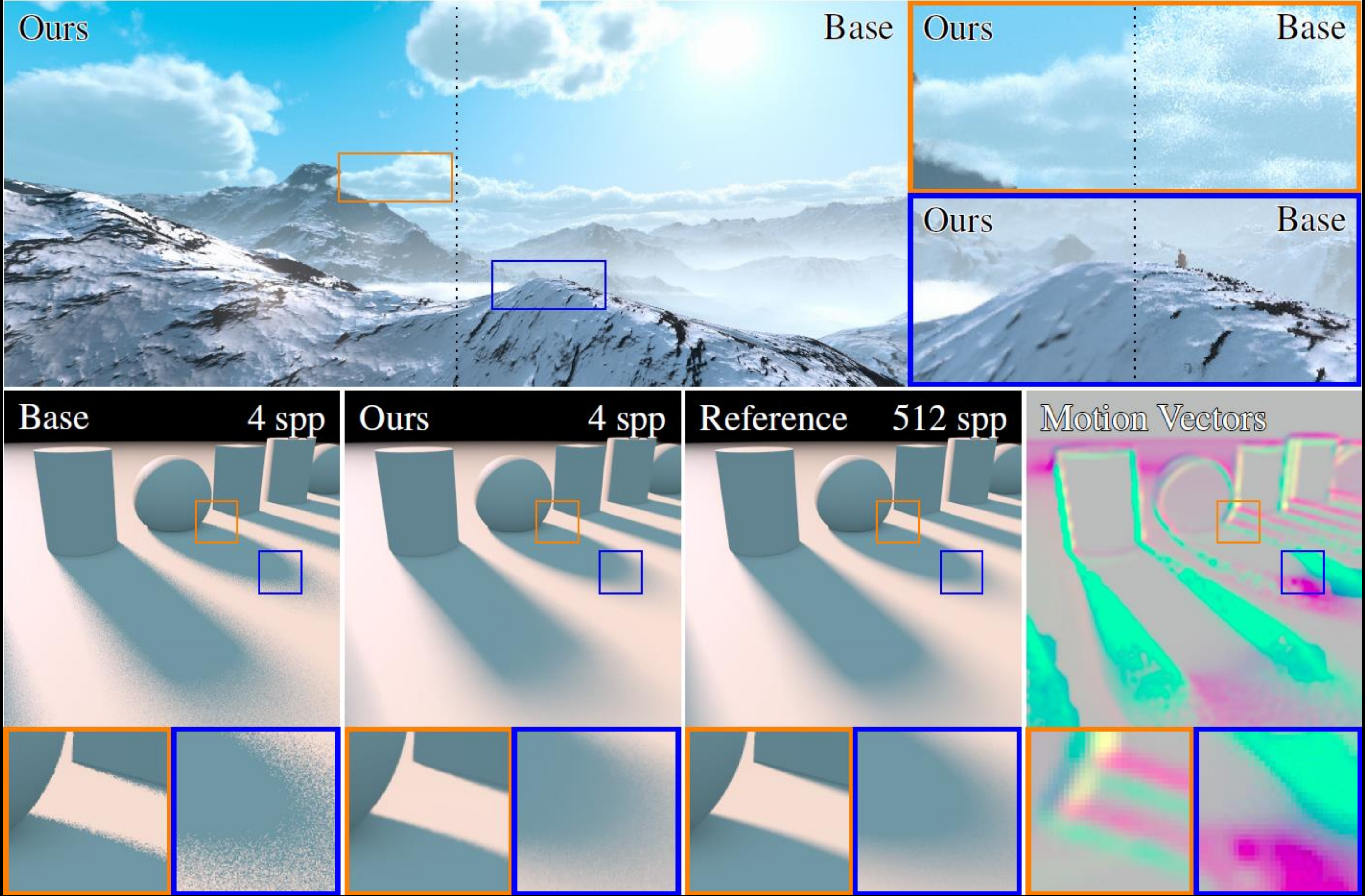
- fails if we have animation
- will not work with depth of field
- will not work with speculars.

A recent paper proposes an alternative*:

For each pixel (i,j), find the shift to similar pixels in the neighborhood by comparing a small patch of pixels around (i,j) to pixels at some distance.

Note: this idea is not new, but the paper makes it efficient using a hierarchical process, where down-sampled versions of the image are used to increase the size of the search window.

*: Fast Temporal Reprojection without Motion Vectors. Hanika & Tessari, 2021.


Ours
Base
Ours
Base
Ours
Base
Base
4 spp
Ours
4 spp
Reference
512 spp
Motion Vectors

Ingredients

1 kernels

2 split

3 temporal

Caching in world space

Instead of searching the current pixel in the previous frame in screen space, we can also maintain a cache in world space*.

Path space filtering:

- Store information in a 3D grid
- Map the grid cells to a hash map
- Update grid cells for each vertex that 'visits' it

Note that a single cell may still receive shading information for surfaces with different normals.

*: Binder et al., Massively Parallel Path Space Filtering, 2019.

Massively Parallel Path Space Filtering

Nikolaus Binder, NVIDIA — Sascha Fricke, University of Braunschweig — Alexander Keller, NVIDIA

Figure 1: Path tracing at one path per pixel (left) in combination with hashed path space filtering (middle) very closely approximates the reference solution using 1024 paths per pixel (right). The filtering takes about 1.5 ms in HD resolution. Scene credit: Amazon Lumberyard Bistro, Open Research Content Archive [8].

ABSTRACT

15 Feb 2019

Ingredients

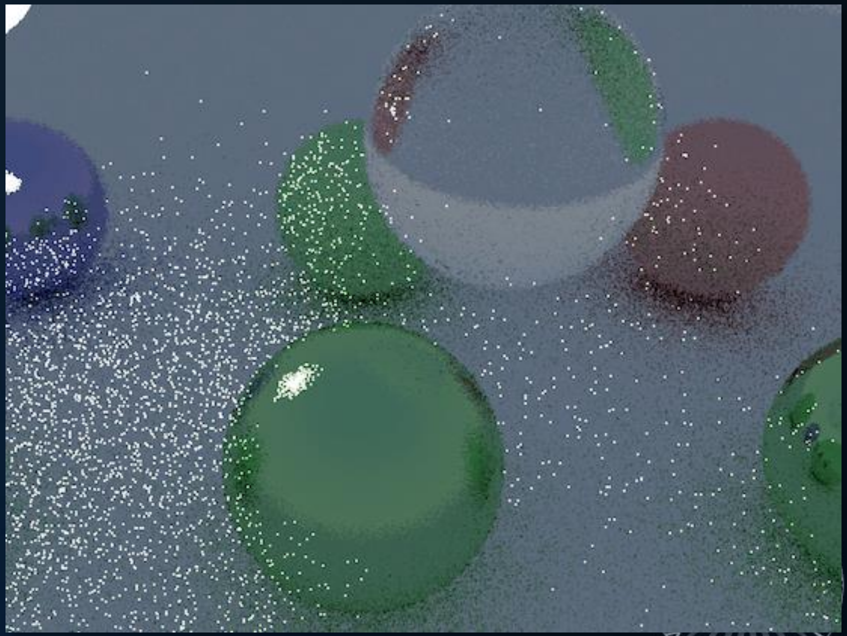
1 kernels
2 split
3 temporal
4 adaptive

Adaptive Sampling

Some pixels need more samples than others.
(to reach a certain variance level)

Adaptive Sampling* aims to estimate which pixels still need work.

Note that reliable variance estimation requires more than a few samples; adaptive sampling is generally not applicable to realtime rendering.



*: A Survey of Adaptive Sampling in Realistic Image Synthesis, M. Sik, 2013.

Ingredients

1 kernels

2 split

3 temporal

4 adaptive

Variance-guided Filtering*

A variance estimate is also useful for steering the filter kernel size:

- A pixel with low variance can use a small kernel
  (which prevents overblurring)
- A pixel with high variance needs a larger kernel
  (to include more samples from neighbors)

SVGF combines bilateral filtering with variance guided kernel sizes and temporal reprojection.



*: Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination. Schied et al., 2017.

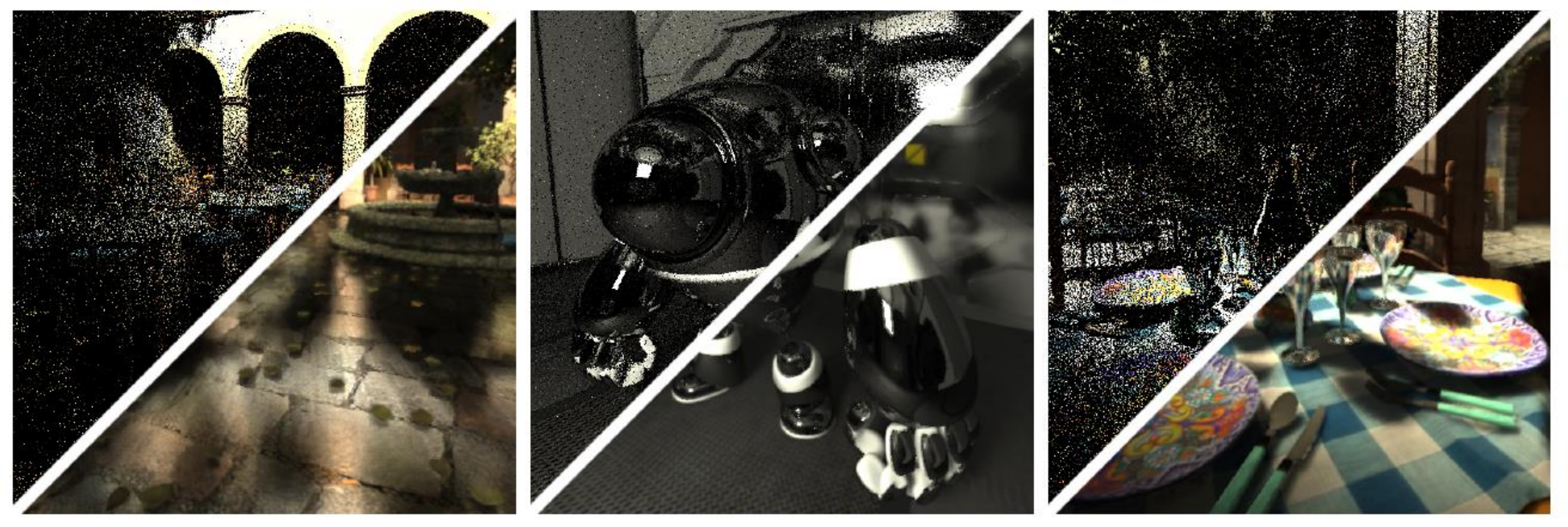
Ingredients

1 kernels

2 split

3 temporal

4 adaptive



https://dspace.library.uu.nl/bitstream/handle/1874/366198/Beyond%20SVGF.pdf

Ingredients

1 kernels
2 split
3 temporal
4 adaptive
5 learning

Machine Learning

Neural networks can be used to filter path tracing noise.

E.g., by learning optimal filter parameters:
A Machine Learning Approach for Filtering Monte Carlo Noise. Kalantari et al., 2015.

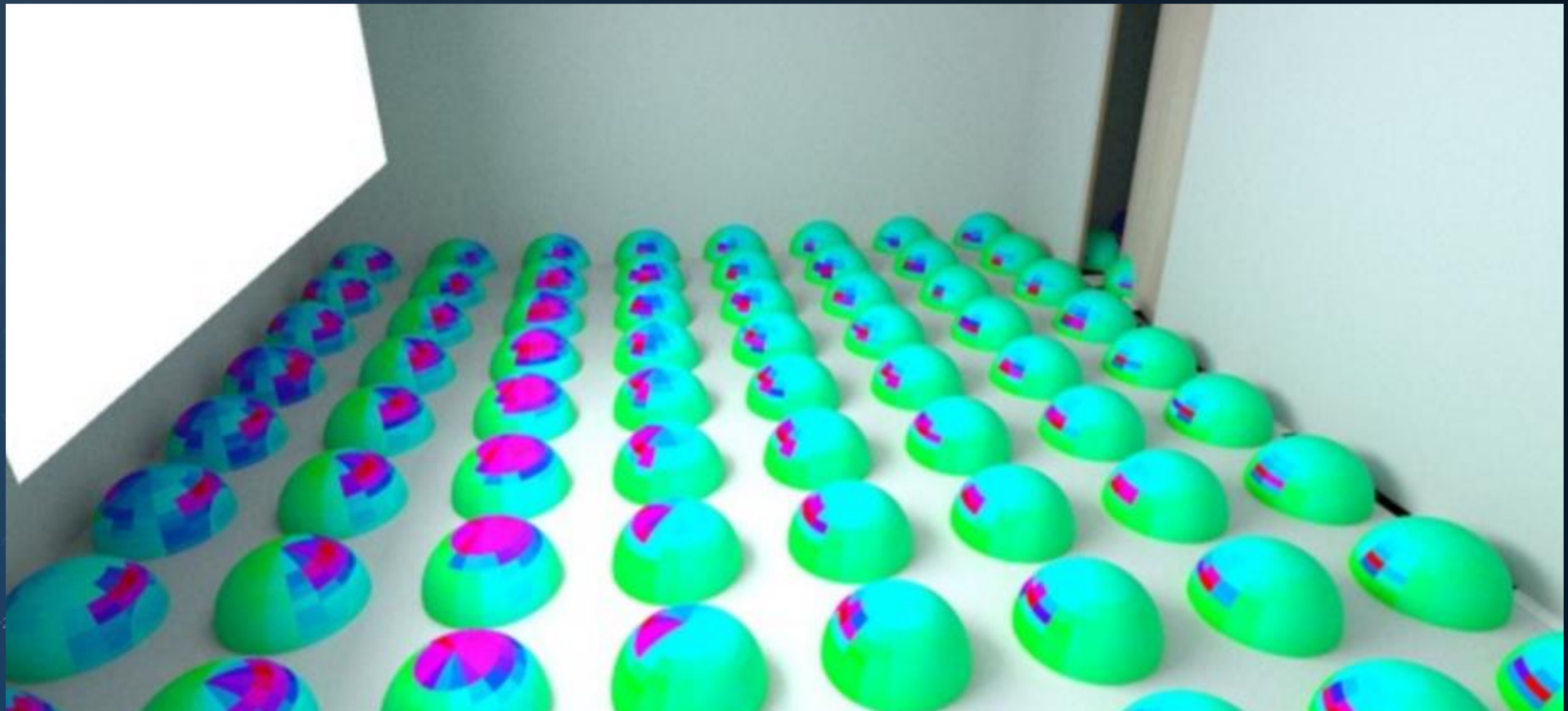
Ingredients

1 kernels
2 split
3 temporal
4 adaptive
5 learning

Machine Learning

Reinforcement Learning can be used to importance sample based on experience.

E.g., by learning light transport while rendering:
Learning Light Transport the Reinforced Way. Dahm & Keller, 2017.



Ingredients

1 kernels

2 split

3 temporal

4 adaptive

5 learning

Machine Learning

Reinforcement Learning can be used to importance sample based on experience.

Reinforcement Learning for rendering is often referred to as path guiding:
Path Guiding in Production. Vorba et al., 2019 (SIGGRAPH 2019 course).

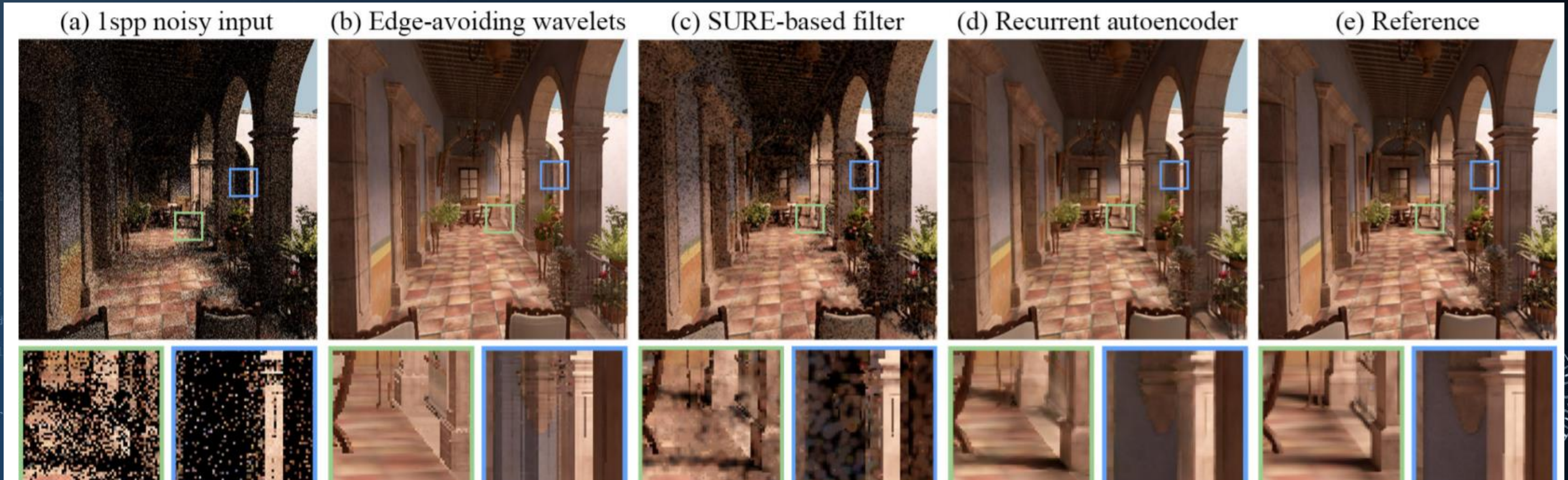
Ingredients

1 kernels
2 split
3 temporal
4 adaptive
5 learning

Machine Learning

And finally: convolutional neural networks.

Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings. Disney / Pixar, University of California: Bako et al., 2019.
Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder. NVIDIA, several universities: Chaitanya et al., 2017.



Today’s Agenda:

- Noise
- Ingredients
- Future Work

Digest

Filtering, practical

First of all, provide a high-quality render:

- Few samples can still be HQ samples
- Many filters get more expensive with high spp counts ➔ spend more time per sample

Prepare your input:

- Separate albedo and illumination
- Separate direct and indirect light
- Suppress outliers
- Supply ‘feature buffers’ for the bilateral kernels
- Use a pinhole camera - postpone AA / DOF
- Reproject; go temporal.

Filter:

- Some form of bilateral
- Steer kernel size with variance estimation
- Ideally: sample-based; pixel-based if this is too slow

Digest

Filtering, open problems

Not easy to do:

- DOF, AA
- Transparency

Considerations for real-time:

- Mind temporal stability
- Don’t make it too crisp
- Make some (uniform) noise a feature
- Consider using DLSS


Camera parameters
Material parameters
name: unnamed
R:255 G:255 B:255 color
R: 0 G: 0 B: 0 absorption
0.000 metallic
0.000 subsurface
0.000 specular
0.000 roughness
0.000 specularTin
0.000 anisotropic
0.000 sheen
0.000 sheenTint
0.000 clearcoat
0.000 clearcoatGl
0.000 transmissio
0.000 eta (1/ior)

Today’s Agenda:

- Noise
- Ingredients
- Future Work

INFOMAGR – Advanced Graphics

Jacco Bikker - November 2021 – February 2022

END of “Filtering”

next lecture: “Bits & Pieces, Exam Training”