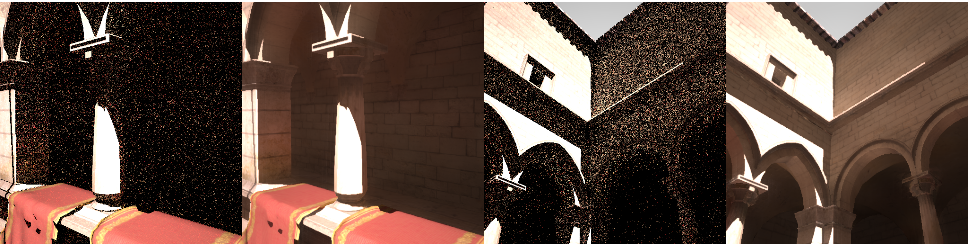
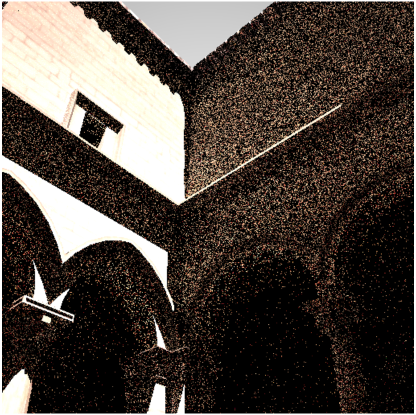
ulm university universität uulm



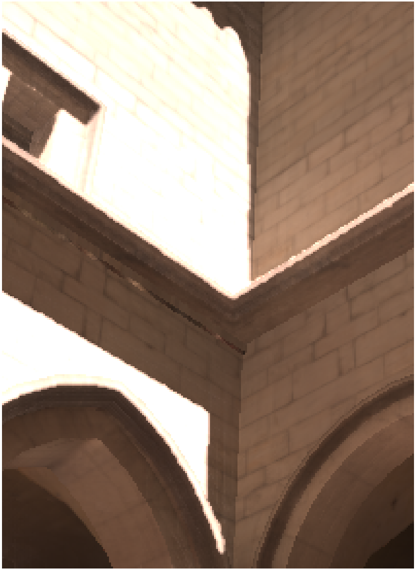
Edge-Avoiding À-Trous Wavelet Transform for fast Global Illumination Filtering

Holger Dammertz, Daniel Sewtz, Johannes Hanika, Hendrik P.A. Lensch

Filtering Noisy Monte Carlo Images



Filtering Noisy Monte Carlo Images



Interactive Global Illumination

Full Monte Carlo simulation

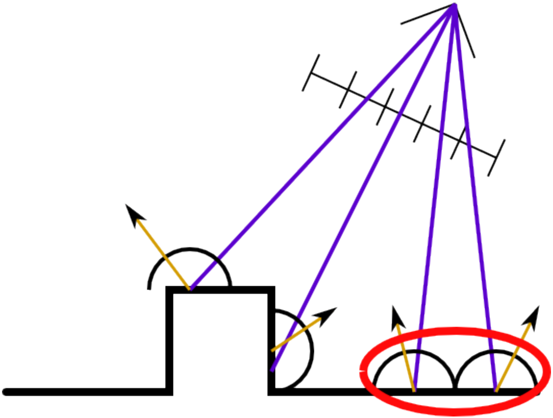
- can simulate all lighting simulations
- hours to converge
- few samples result in noisy images

Filtering of Monte Carlo Samples

- independent of sample generation
- large filter radius
- respect (visible) edges
- small overhead

Motivation for Filtering Monte Carlo Samples

$$L(x,\omega_o) = L_e(x,\omega_o) + \int_{\Omega_x} f_r(x,\omega_i,\omega_o) L_i(x,\omega_i) < n_x, \omega_i > d\omega_i$$



Related Work

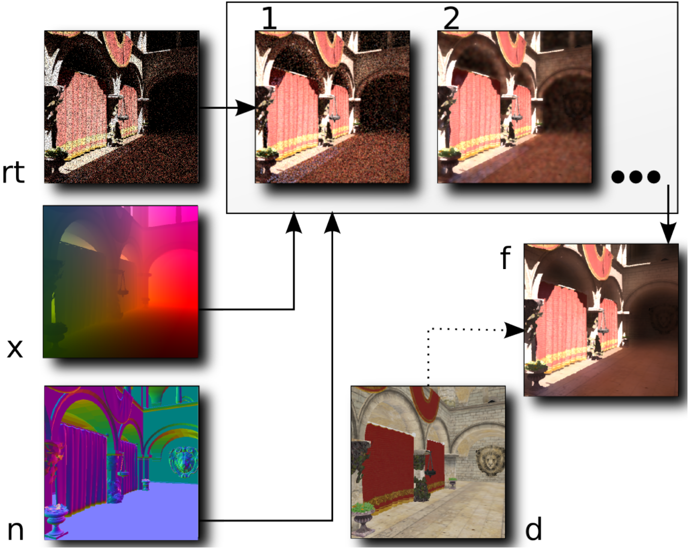
Edge-Avoiding Wavelets and the Bilateral Filter

- Edge Avoiding Wavelets: R. Fattal, 2009
- Discontinuity Buffer: A. Keller, 1998
- Multiscale shape and detail enhancement from multi-light image collections: R. Fattal, M. Agrawala, S. Rusinkiewicz, 2007

Interactive Global Illumination

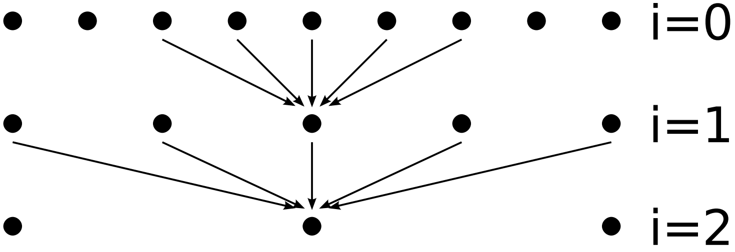
- Precomputed Radiance Transfer, Meshless Radiosity
- Photon Mapping
- Instant Radiosity

Overview



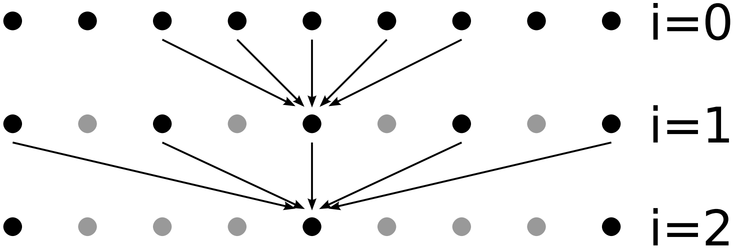
Decimated Wavelets

- Reduce coarse coefficients
- At each Iteration
  - Half the resolution
  - and average signal



À-Trous Wavelet

- With Holes
- At each Iteration
  - double the filter size
  - by skipping more values each time



- Benefits:
  - constant effort per iteration (in contrast to undecimated wavelets)
  - filtered information at each pixel (in contrast to decimated wavelets)

Algorithme À-Trous

À-Trous Wavelet Transform

1. At level $i = 0$ we start with the input signal $c_0(p)$
2. $c_{i+1}(p) = c_i(p) * h_i$, where $*$ is the discrete convolution. The distance between the entries in the filter h_i is 2^i.
3. $d_i(p) = c_i(p) - c_{i+1}(p)$,
   where d_i are the detail or wavelet coefficients of level i.
4. if $i < N$ (number of levels to compute):
   increment i, go to step 2
5. $\{d_0, d_1, ..., d_{N-1}, c_N\}$ is the wavelet transform of c.

Reconstruction:

$$c = c_N + \sum_{i=N-1}^{0} d_i$$

Algorithme À-Trous

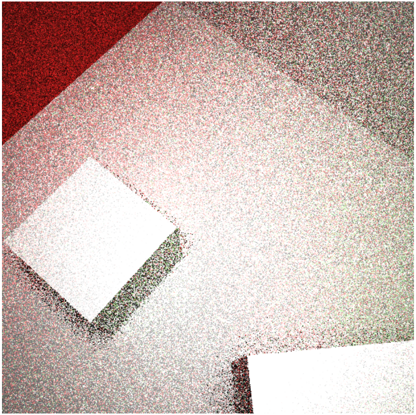
À-Trous Wavelet Filter

1. At level $i = 0$ we start with the input signal $c_0(p)$
2. $c_{i+1}(p) = c_i(p) * h_i$, where $*$ is the discrete convolution. The distance between the entries in the filter h_i is 2^i.
3. $d_i(p) = c_i(p) - c_{i+1}(p)$,
   where d_i are the detail or wavelet coefficients of level i.
4. if $i < N$ (number of levels to compute):
   increment i, go to step 2
5. $\{d_0, d_1, ..., d_{N-1}, c_N\}$ is the wavelet transform of c.

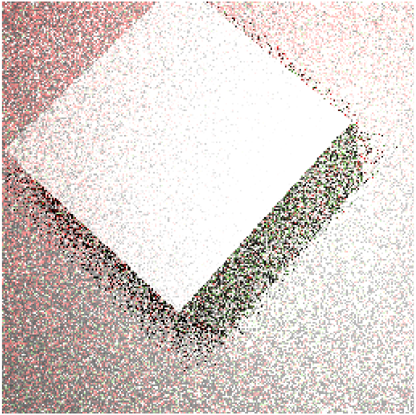
Filtered Result:

$$f = c_N$$

À-Trous Filter



À-Trous Filter



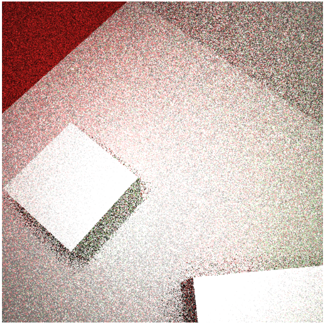
Edge Stopping Function

Compute weighted convolution

$$c_{i+1}(p) = \frac{\sum_{q \in \Omega} h_i(q) \cdot w(p,q) \cdot c_i(p)}{\sum_{q \in \Omega} h_i(q) \cdot w(p,q)}$$

with weights

$$w(p,q) = w_n \cdot w_x \cdot w_{rt}$$



rt

orig. À-Trous

Edge Stopping Function

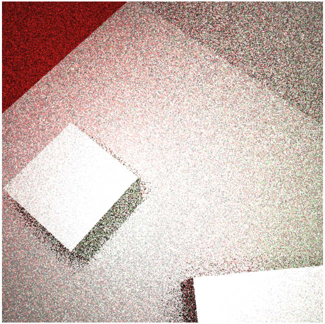
Compute weighted convolution

$$c_{i+1}(p) = \frac{\sum_{q \in \Omega} h_i(q) \cdot w(p,q) \cdot c_i(p)}{\sum_{q \in \Omega} h_i(q) \cdot w(p,q)}$$

with weights

$w(p,q) = w_n \cdot w_x \cdot w_{rt}$

$w_n(p,q) = e^{\left(-\frac{||n_p - n_q||}{\sigma_n^2}\right)}$



rt

w_n

Edge Stopping Function

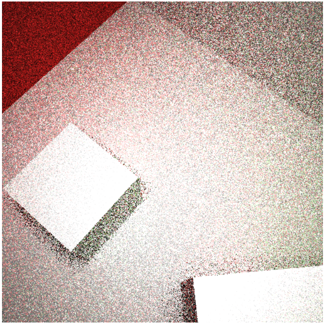
Compute weighted convolution

$$c_{i+1}(p) = \frac{\sum_{q\in\Omega} h_i(q)\cdot w(p,q)\cdot c_i(p)}{\sum_{q\in\Omega} h_i(q)\cdot w(p,q)}$$

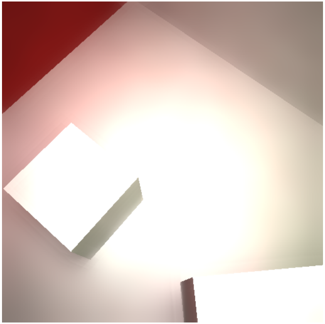
with weights

$w(p,q) = w_n \cdot w_x \cdot w_{rt}$

$w_x(p,q) = e^{\left(-\frac{||x_p - x_q||}{\sigma_x^2}\right)}$



rt



$w_n \cdot w_x$

Edge Stopping Function

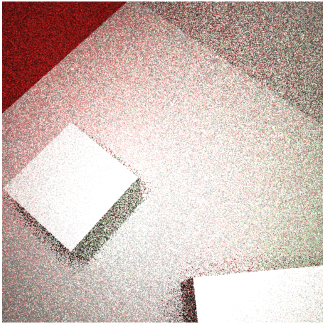
Compute weighted convolution

$$c_{i+1}(p) = \frac{\sum_{q\in\Omega} h_i(q) \cdot w(p,q) \cdot c_i(p)}{\sum_{q\in\Omega} h_i(q) \cdot w(p,q)}$$

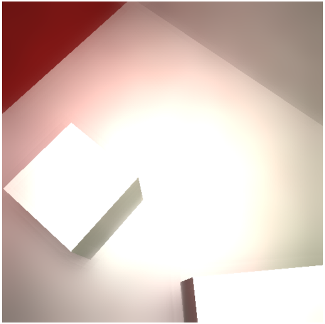
with weights

$w(p,q) = w_n \cdot w_x \cdot w_{rt}$

$w_{rt}(p,q) = e^{\left(-\frac{||I_p - I_q||}{\sigma_{rt}^2}\right)}$



rt



$w_n \cdot w_x \cdot w_{rt}$

Hard edges in the lighting

rt input

full edge stopping

without rt

- To work correctly needs low variance direct illumination
- Well chosen σ_{rt}

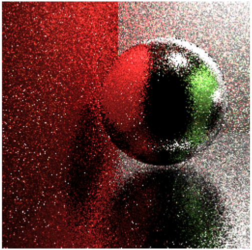
Performance

Timings (GTX285)

Scene	RT (ms)	GL (ms)	Upload (ms)	À-Trous (ms)	FPS
Box	307.6	2.9	5.8	5.6	3.2
Sponza	835.2	35.5	4.4	5.6	1.13

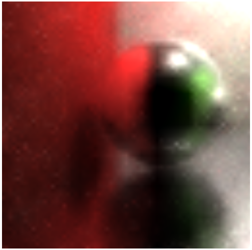
Res.	512×512			1024×1024			1920×1080		
# Iter	1	5	10	1	5	10	1	5	10
ms	1.2	5.6	11.0	4.2	20.9	41.6	8.2	42.6	86.0

Comparison to CDF(2,2) Wavelet

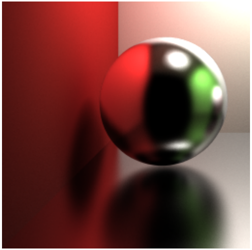


PT Input

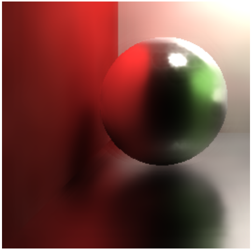
À-Trous



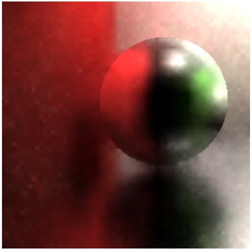
CDF(2,2)



PT Reference

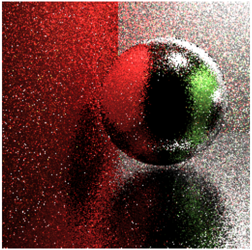


Ours



EAW CDF(2,2)

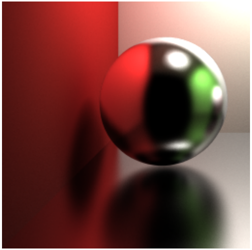
Comparison to RedBlack Wavelet



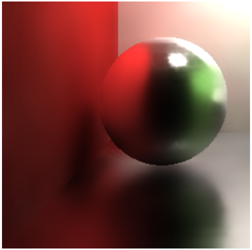
PT Input

À-Trous

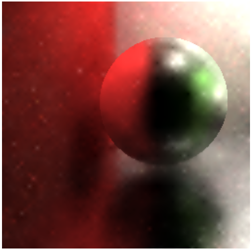
RedBlack



PT Reference

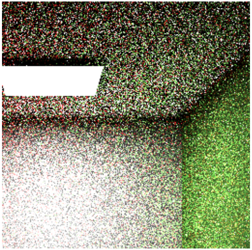


Ours



EAW RedBlack

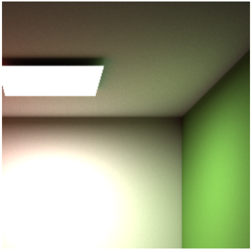
Decimated vs. À-Trous



PT Input

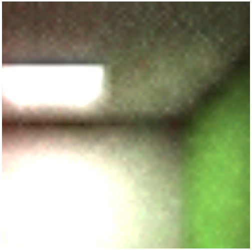
À-Trous

CDF(2,2)



PT Reference

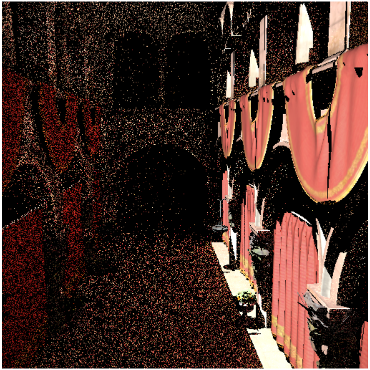
Ours



EAW CDF(2,2)

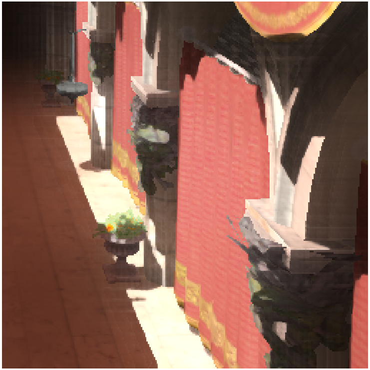
Optimizations

- Decouple Direct/Indirect Illumination
  - Rasterize primary rays and compute direct lighting with other method
  - Ray Trace only indirect illumination, filter it and add it to final image
- Subsample the input image



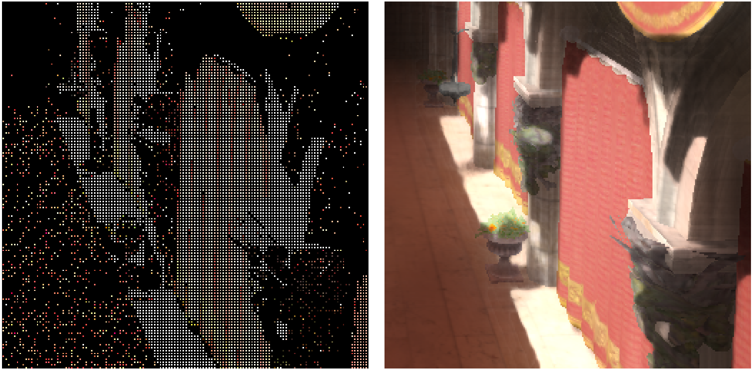
Optimizations

- Decouple Direct/Indirect Illumination
  - Rasterize primary rays and compute direct lighting with other method
  - Ray Trace only indirect illumination, filter it and add it to final image
- Subsample the input image



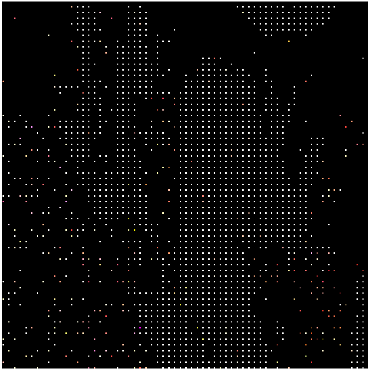
Optimizations

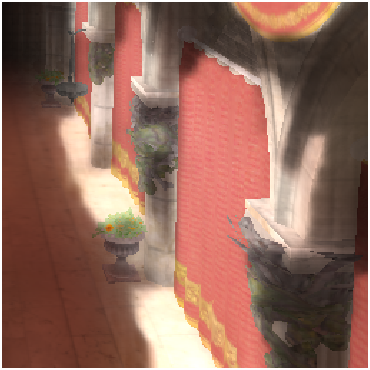
- Decouple Direct/Indirect Illumination
  - Rasterize primary rays and compute direct lighting with other method
  - Ray Trace only indirect illumination, filter it and add it to final image
- Subsample the input image



Optimizations

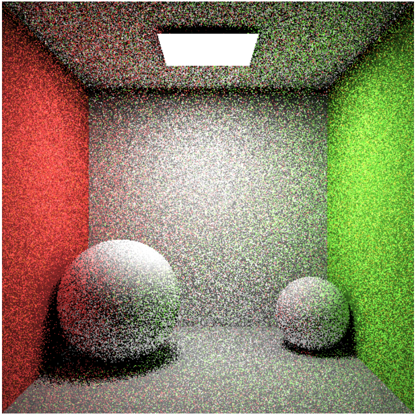
- Decouple Direct/Indirect Illumination
  - Rasterize primary rays and compute direct lighting with other method
  - Ray Trace only indirect illumination, filter it and add it to final image
- Subsample the input image



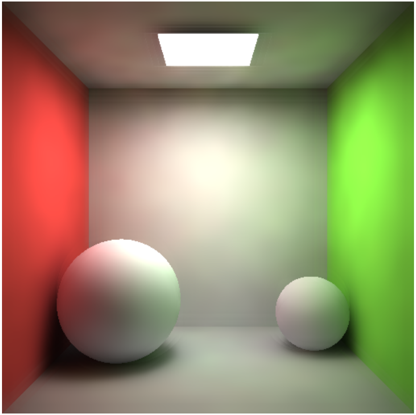


Results

Input

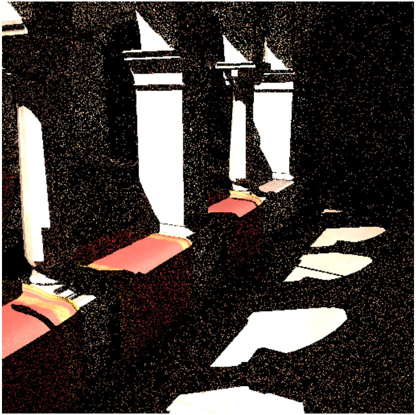


Output

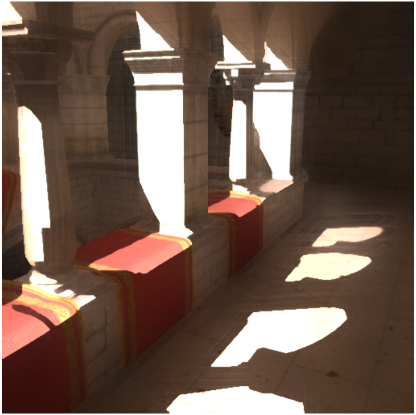


Results

Input

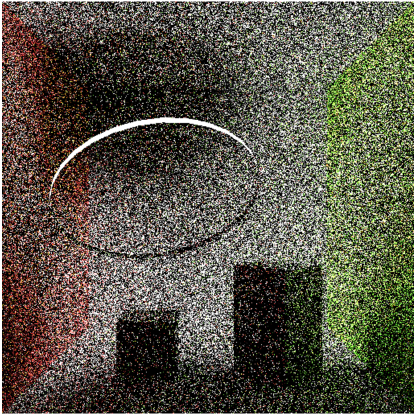


Output

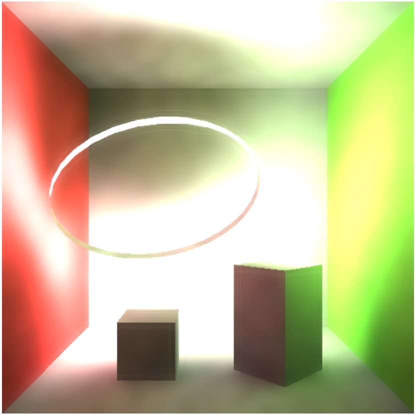


Results

Input

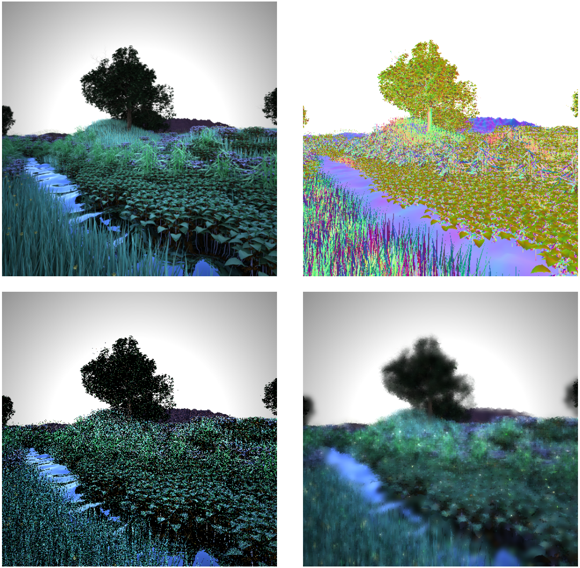


Output



Failure Case

highly undersampled input



Summary

Conclusion

- Filter for noisy monte carlo images
- preserving many sharp details
- at interactive rates
- when not undersampled and correct sigma

Future Work

- Extend to time dimension
- Adaptive Edge Stopping Function

Summary

Conclusion

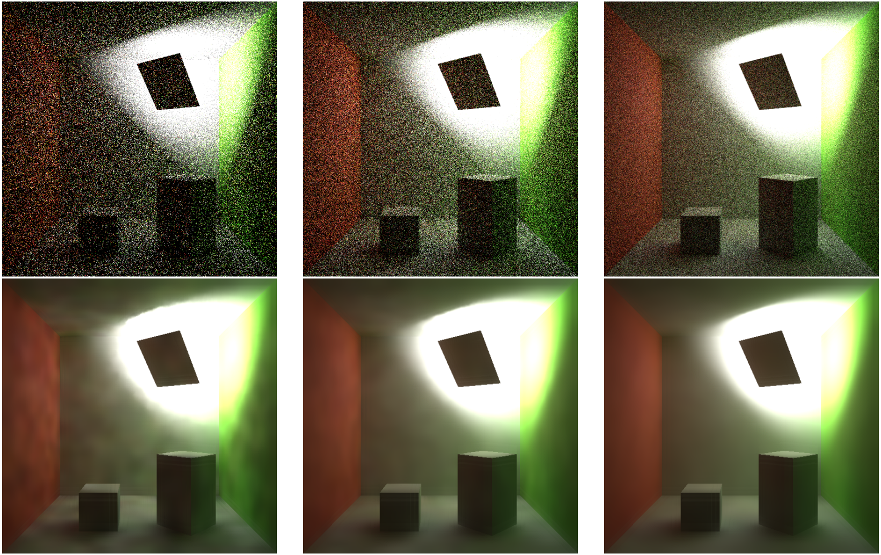
- Filter for noisy monte carlo images
- preserving many sharp details
- at interactive rates
- when not undersampled and correct sigma

Future Work

- Extend to time dimension
- Adaptive Edge Stopping Function

Questions?

More than one bounce per pixel



GLSL Implementation

```
uniform sampler2D colorMap, normalMap, posMap;
uniform float c_phi, n_phi, p_phi, stepwidth;
uniform float kernel[25];
uniform vec2 offset[25];

void main(void) {
 vec4 sum = vec4(0.0);
 vec2 step = vec2(1./512., 1./512.); // resolution
 vec4 cval = texture2D(colorMap, gl_TexCoord[0].st);
 vec4 nval = texture2D(normalMap, gl_TexCoord[0].st);
 vec4 pval = texture2D(posMap, gl_TexCoord[0].st);

 float cum_w = 0.0;
 for(int i = 0; i < 25; i++) {
  vec2 uv = gl_TexCoord[0].st + offset[i]*step*stepwidth;

  vec4 ctmp = texture2D(colorMap, uv);
  vec4 t = cval - ctmp;
  float dist2 = dot(t,t);
  float c_w = min(exp(-(dist2)/c_phi), 1.0);

  vec4 ntmp = texture2D(normalMap, uv);
  t = nval - ntmp;
  dist2 = max(dot(t,t)/(stepwidth*stepwidth),0.0);
  float n_w = min(exp(-(dist2)/n_phi), 1.0);

  vec4 ptmp = texture2D(posMap, uv);
  t = pval - ptmp;
  dist2 = dot(t,t);
  float p_w = min(exp(-(dist2)/p_phi),1.0);

  float weight = c_w * n_w * p_w;
  sum += ctmp * weight * kernel[i];
  cum_w += weight*kernel[i];
 }
 gl_FragData[0] = sum/cum_w;
}
```

Implementation Details

CPU

- global illumination ray tracing with 1 sample per pixel
  - upload rt buffer to GPU

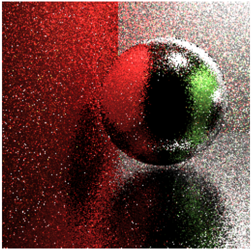
OpenGL

- render normal and position buffer
- wrap rt, normal and position buffer into textures

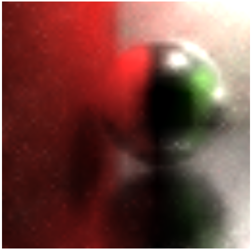
Shader

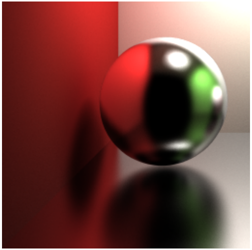
- Apply filter by rendering to texture using input buffers
- Iterate with growing filter size
- Performance depends only on resolution and filter size

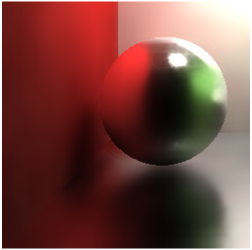
Comparison to CDF(2,2) Wavelet

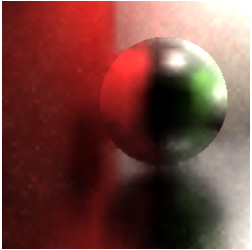

PT Input


À-Trous

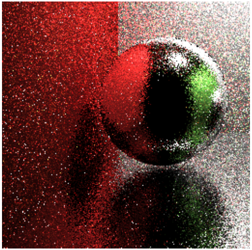

CDF(2,2)


PT Reference


Ours


EAW CDF(2,2)

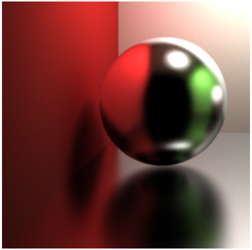
Comparison to RedBlack Wavelet



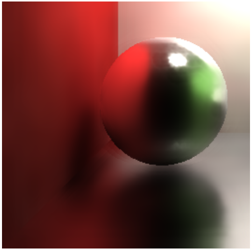
PT Input

À-Trous

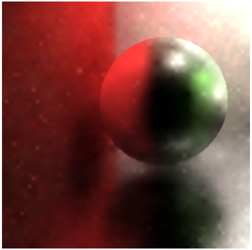
RedBlack



PT Reference

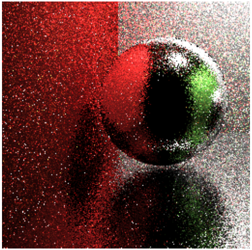


Ours



EAW RedBlack

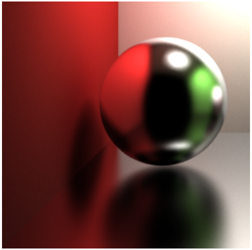
Comparison to Bilateral Filter



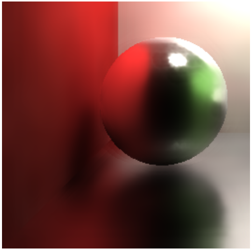
PT Input

À-Trous

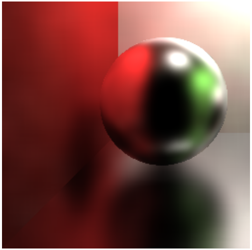
Bilateral



PT Reference



Ours



Multilateral