NVIDIA

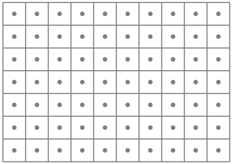
Ray-Traced Global Illumination for Games: Massively Parallel Path Space Filtering

Nikolaus Binder and Alexander Keller

Principles of Image Synthesis

Solving the visibility problem

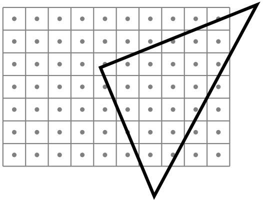
Rasterization



Principles of Image Synthesis

Solving the visibility problem

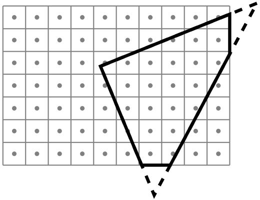
Rasterization



Principles of Image Synthesis

Solving the visibility problem

Rasterization

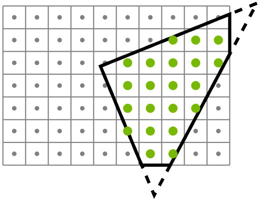


clipping

Principles of Image Synthesis

Solving the visibility problem

Rasterization



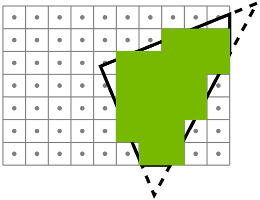
clipping

Z-buffer

Principles of Image Synthesis

Solving the visibility problem

Rasterization

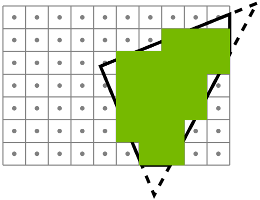
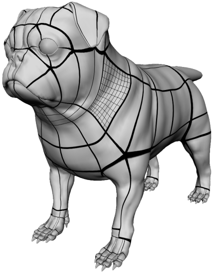


clipping

Z-buffer

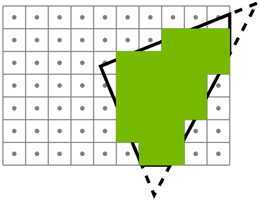
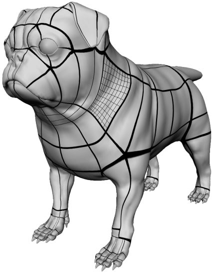
Principles of Image Synthesis

Solving the visibility problem

Rasterization	Reyes
	
clipping	dicing
Z-buffer	kind of Z-Buffer

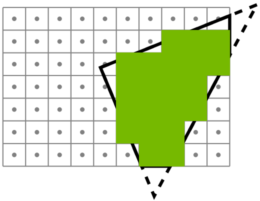
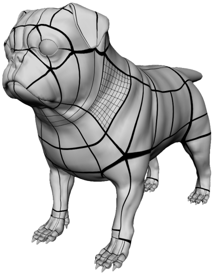
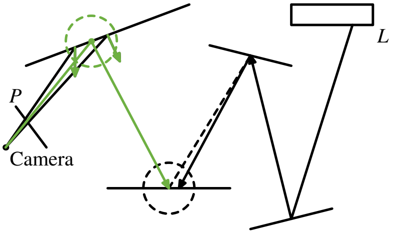
Principles of Image Synthesis

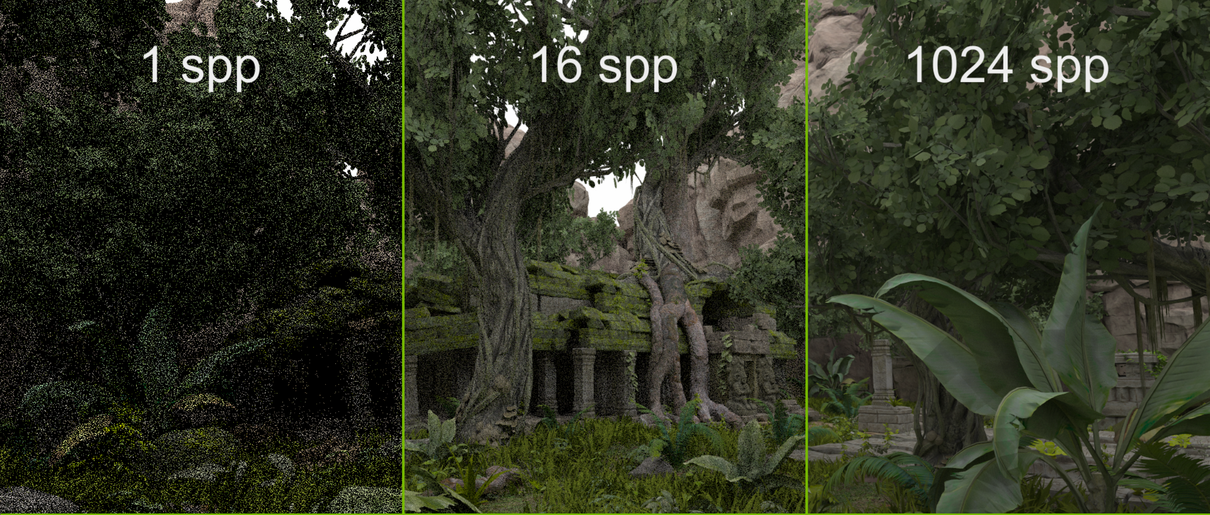
Solving the visibility problem

Rasterization	Reyes
	
clipping	dicing
Z-buffer	kind of Z-Buffer
shadow maps	shadow maps

Principles of Image Synthesis

Solving the visibility problem

Rasterization	Reyes	Ray Tracing
		
clipping	dicing	acceleration data structure
Z-buffer	kind of Z-Buffer	tracing rays with arbitrary origins
shadow maps	shadow maps	shadow rays



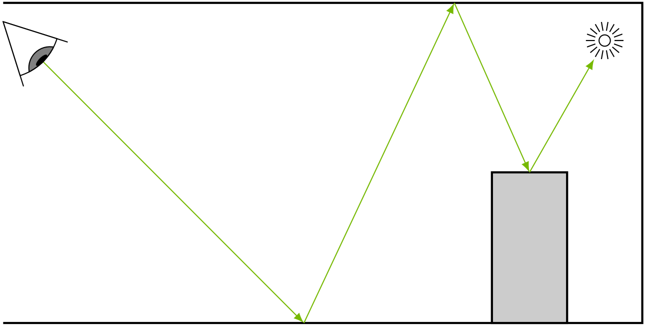
Path tracing on a budget

Massively Parallel Path Space Filtering

Massively Parallel Path Space Filtering

Sharing instead of splitting

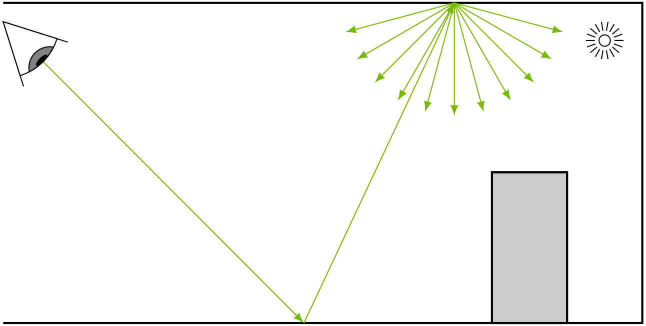
- filtering beyond screen space



Massively Parallel Path Space Filtering

Sharing instead of splitting

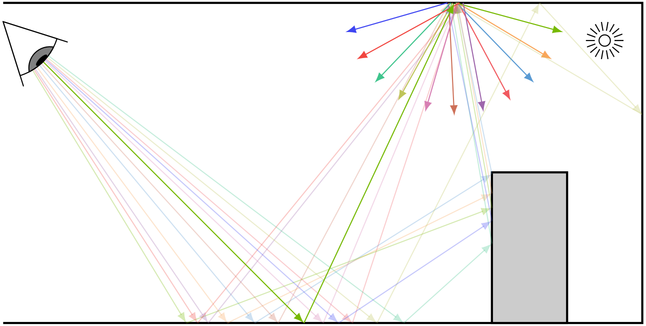
- filtering beyond screen space
- algorithm



Massively Parallel Path Space Filtering

Sharing instead of splitting

- filtering beyond screen space
- algorithm
  1. generate paths,
     select and store vertices

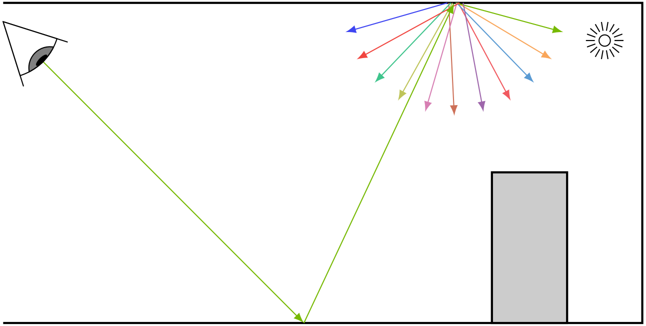


Massively Parallel Path Space Filtering

Sharing instead of splitting

- filtering beyond screen space

- algorithm
  1. generate paths,
     select and store vertices
  2. average contributions
     with similar vertex descriptors

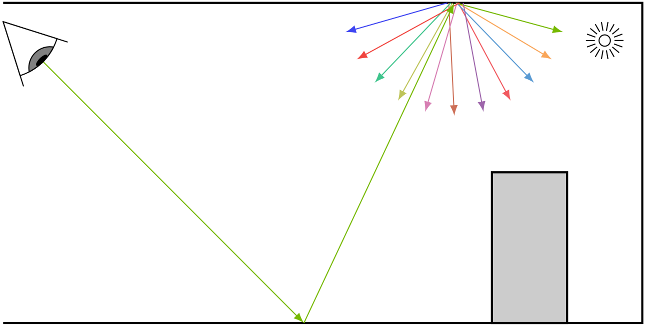


Massively Parallel Path Space Filtering

Sharing instead of splitting

- filtering beyond screen space

- algorithm
  1. generate paths,
     select and store vertices
  2. average contributions
     with similar vertex descriptors
  3. use averaged contributions

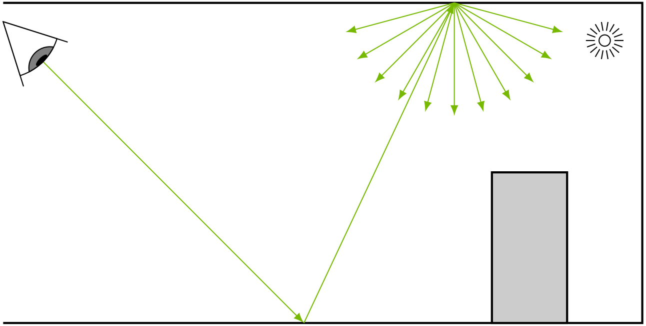


Massively Parallel Path Space Filtering

Sharing instead of splitting

- filtering beyond screen space

- algorithm
  1. generate paths,
     select and store vertices
  2. average contributions
     with similar vertex descriptors
  3. use averaged contributions

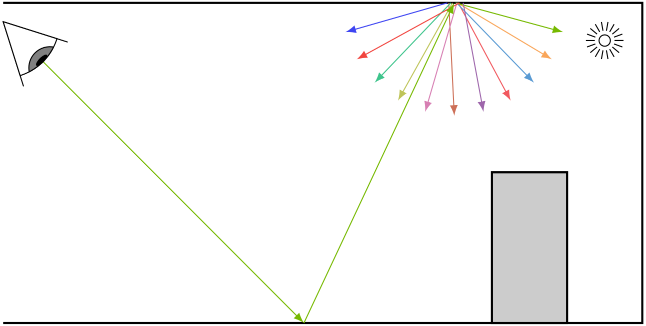


Massively Parallel Path Space Filtering

Sharing instead of splitting

- filtering beyond screen space

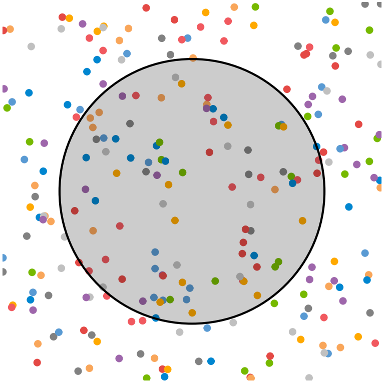
- algorithm
  1. generate paths,
     select and store vertices
  2. average contributions
     with similar vertex descriptors
  3. use averaged contributions



Massively Parallel Path Space Filtering

Bottleneck: Calculating averages

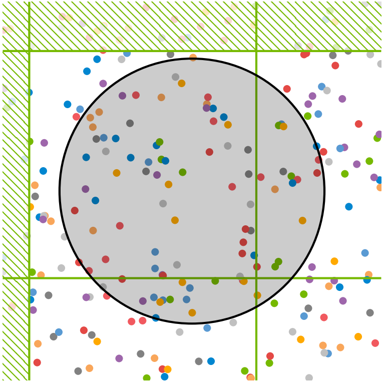
- include many “close by” contributions in average



Massively Parallel Path Space Filtering

Bottleneck: Calculating averages

- include many “close by” contributions in average
  - efficient culling by range search

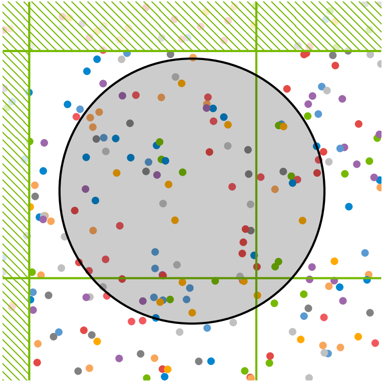


NVIDIA.

Massively Parallel Path Space Filtering

Bottleneck: Calculating averages

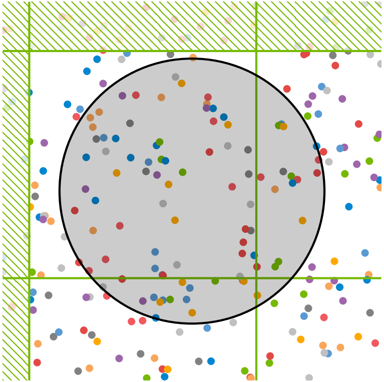
- include many “close by” contributions in average
  - efficient culling by range search
  - but still have to iterate over all of them



Massively Parallel Path Space Filtering

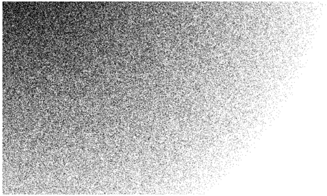
Bottleneck: Calculating averages

- include many “close by” contributions in average
  - efficient culling by range search
  - but still have to iterate over all of them
  - and every vertex needs to do this individually



Massively Parallel Path Space Filtering

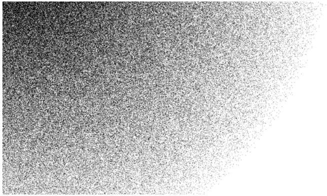
Principle



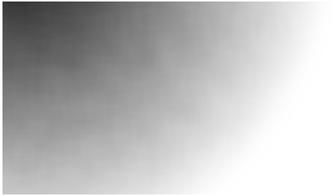
input

Massively Parallel Path Space Filtering

Principle



input

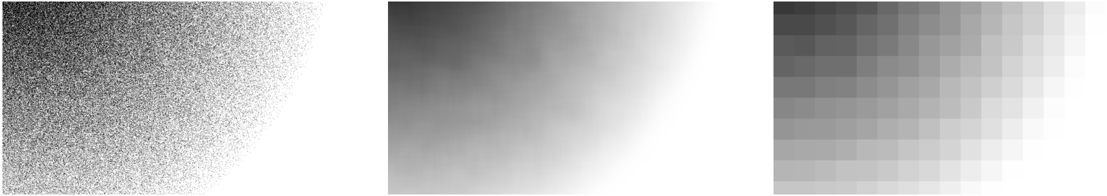


local averaging

NVIDIA.

Massively Parallel Path Space Filtering

Principle

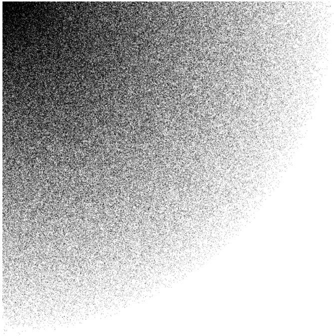


input | local averaging | average per cell

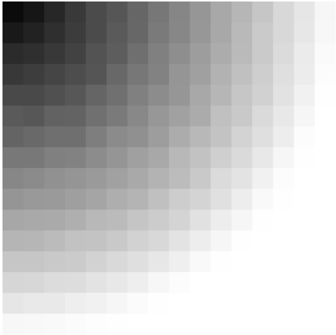
- instead of calculating one average **per vertex**, calculate one average **per cell**
  - cell identified by quantizing a descriptor $(x_i, \ldots)$
  - proximity defined by equality after quantization instead of distance
  - worst case complexity $\mathcal{O}(N)$ instead of $\mathcal{O}(N^2)$

Massively Parallel Path Space Filtering

Resolving quantization artifacts



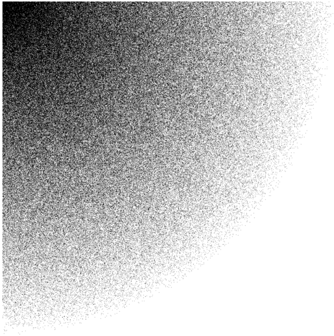
input



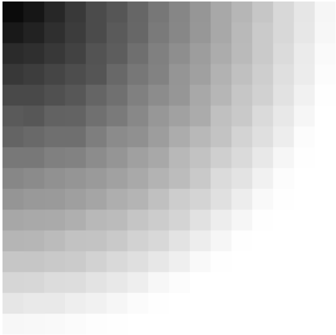
average per cell

Massively Parallel Path Space Filtering

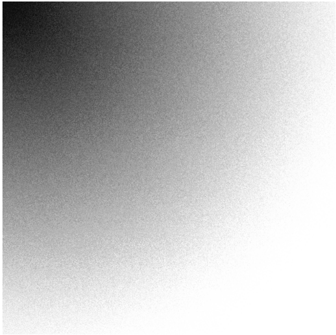
Resolving quantization artifacts



input



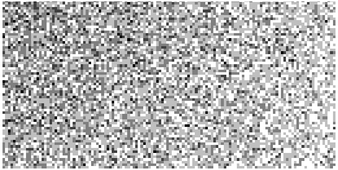
average per cell



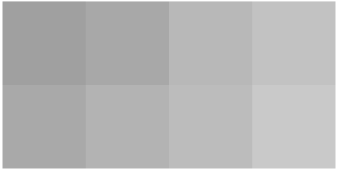
with jittering

Massively Parallel Path Space Filtering

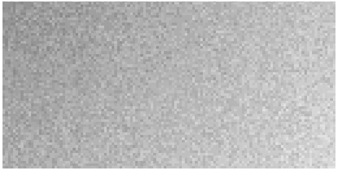
Resolving quantization artifacts



input



average per cell

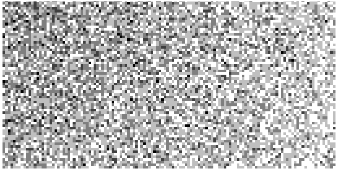


with jittering

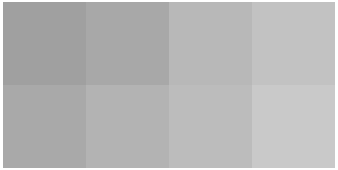
- jittering descriptor (x_i, ...) on store and look up

Massively Parallel Path Space Filtering

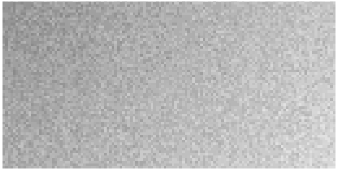
Resolving quantization artifacts



input



average per cell



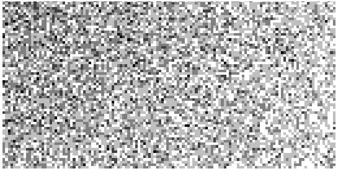
with jittering

- jittering descriptor (x_i, ...) on store and look up
  - hides quantization artifacts

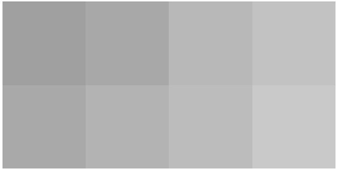
NVIDIA.

Massively Parallel Path Space Filtering

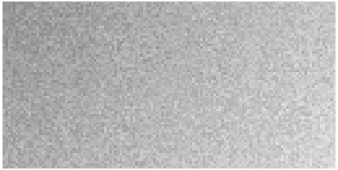
Resolving quantization artifacts



input



average per cell

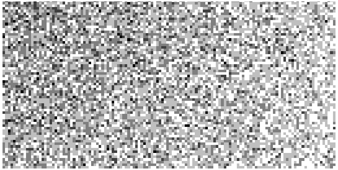


with jittering

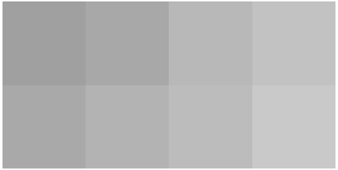
- jittering descriptor (x_i, ...) on store and look up
  - hides quantization artifacts
  - resulting uniform noise amenable to (existing) post filtering

Massively Parallel Path Space Filtering

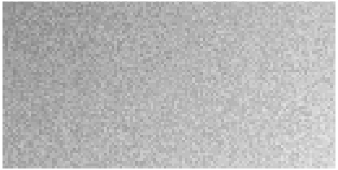
Resolving quantization artifacts



input



average per cell



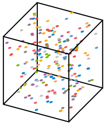
with jittering

- jittering descriptor (x_i, ...) on store and look up
  - hides quantization artifacts
  - resulting uniform noise amenable to (existing) post filtering
- amounts to stochastic evaluation of interpolation

Massively Parallel Path Space Filtering

Hashing instead of searching

- descriptors for selected vertices include

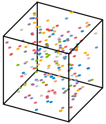


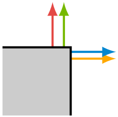
world space location *x*

Massively Parallel Path Space Filtering

Hashing instead of searching

- descriptors for selected vertices include



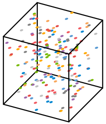


world space location x and optionally normal n,

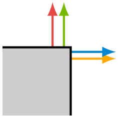
Massively Parallel Path Space Filtering

Hashing instead of searching

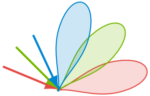
- descriptors for selected vertices include



world space location x



and optionally normal n,

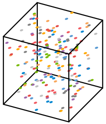


incident angle ω,

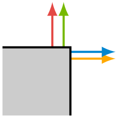
Massively Parallel Path Space Filtering

Hashing instead of searching

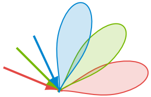
- descriptors for selected vertices include



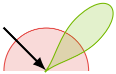
world space location x



and optionally normal n,



incident angle ω,



and BRDF layer

Massively Parallel Path Space Filtering

Storing and looking up data with quantized descriptors

- fast updates, no pre-processing

Massively Parallel Path Space Filtering

Storing and looking up data with quantized descriptors

- fast updates, no pre-processing
- access in constant time

Massively Parallel Path Space Filtering

Storing and looking up data with quantized descriptors

- fast updates, no pre-processing
- access in constant time
  - requires injective mapping $(x, n, ...) \mapsto [0, M)$

Massively Parallel Path Space Filtering

Storing and looking up data with quantized descriptors

- fast updates, no pre-processing
- access in constant time
  - requires injective mapping $(x, n, \ldots) \mapsto [0, M)$

$\Rightarrow$ **hash map**

Massively Parallel Path Space Filtering

Fast hash map

- trade a larger table size for faster access

Massively Parallel Path Space Filtering

Fast hash map

- trade a larger table size for faster access
- simple, fast hash functions

Massively Parallel Path Space Filtering

Fast hash map

- trade a larger table size for faster access
- simple, fast hash functions
- linear probing for collision resolution

Massively Parallel Path Space Filtering

Fast hash map

- trade a larger table size for faster access
- simple, fast hash functions
- linear probing for collision resolution
- use a second hash of the descriptor instead of storing full keys
  - may fail, but is very very unlikely

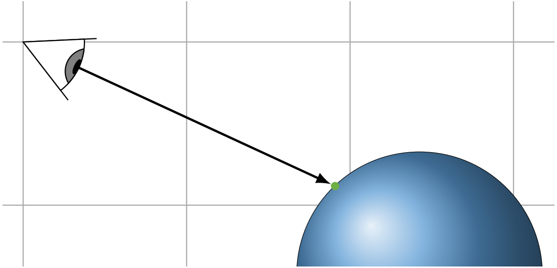
Massively Parallel Path Space Filtering

Linear instead of quadratic

- finding the hash table location i

$$i \leftarrow \text{hash}(\tilde{x}, \ldots) \ \% \ \text{table\_size}$$

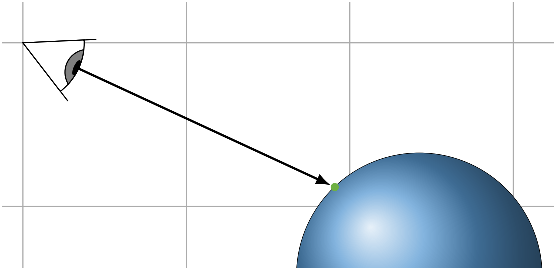
for both averaging and querying



Massively Parallel Path Space Filtering

Linear instead of quadratic

- finding the hash table location i



$$i \leftarrow \text{hash}(\tilde{x}, \ldots) \ \% \ \text{table\_size}$$

$$v \leftarrow \text{hash2}(\tilde{x}, n, \ldots)$$

for both averaging and querying

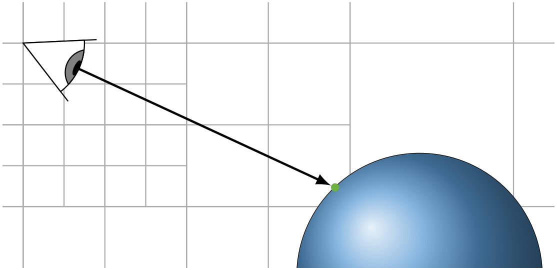
Massively Parallel Path Space Filtering

Linear instead of quadratic

- finding the hash table location i

$$
\begin{aligned}
l' &\leftarrow \text{level\_of\_detail}(|p_{\text{cam}} - x'|) \\
\tilde{x} &\leftarrow \left\lfloor \frac{x'}{scale \cdot 2^{l'}} \right\rfloor \\
i &\leftarrow \text{hash}(\tilde{x}, \ldots) \ \% \ \text{table\_size} \\
v &\leftarrow \text{hash2}(\tilde{x}, n, \ldots)
\end{aligned}
$$

for both averaging and querying



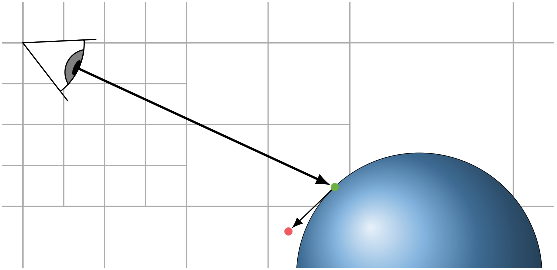
Massively Parallel Path Space Filtering

Linear instead of quadratic

- finding the hash table location i

$$l \leftarrow \text{level\_of\_detail}(|p_{\text{cam}} - x|)$$
$$x' \leftarrow x + \text{jitter}(n) \cdot \mathit{scale} \cdot 2^{l}$$
$$l' \leftarrow \text{level\_of\_detail}(|p_{\text{cam}} - x'|)$$
$$\tilde{x} \leftarrow \left\lfloor \frac{x'}{\mathit{scale} \cdot 2^{l'}} \right\rfloor$$
$$i \leftarrow \text{hash}(\tilde{x}, \ldots) \;\; \% \text{ table\_size}$$
$$v \leftarrow \text{hash2}(\tilde{x}, n, \ldots)$$

for both averaging and querying



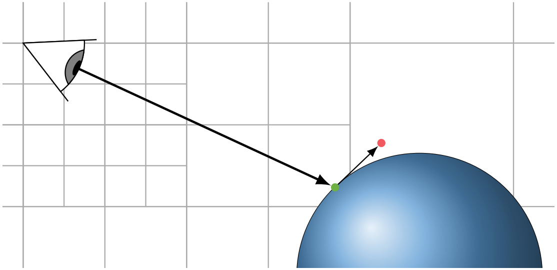
Massively Parallel Path Space Filtering

Linear instead of quadratic

- finding the hash table location i

$$l \leftarrow \text{level\_of\_detail}(|p_{\mathrm{cam}} - x|)$$
$$x' \leftarrow x + \text{jitter}(n) \cdot scale \cdot 2^l$$
$$l' \leftarrow \text{level\_of\_detail}(|p_{\mathrm{cam}} - x'|)$$
$$\tilde{x} \leftarrow \left\lfloor \frac{x'}{scale \cdot 2^{l'}} \right\rfloor$$
$$i \leftarrow \text{hash}(\tilde{x}, \ldots) \ \ \% \text{ table\_size}$$
$$v \leftarrow \text{hash2}(\tilde{x}, n, \ldots)$$

for both averaging and querying



Massively Parallel Path Space Filtering

Linear instead of quadratic

- finding the hash table location i

$$
\begin{aligned}
l' &\leftarrow \text{level\_of\_detail}(|p_{\mathrm{cam}} - x'|) \\
\tilde{x} &\leftarrow \left\lfloor \frac{x'}{scale \cdot 2^{l'}} \right\rfloor \\
i &\leftarrow \text{hash}(\tilde{x}, \ldots) \ \% \ \text{table\_size} \\
v &\leftarrow \text{hash2}(\tilde{x}, n, \ldots)
\end{aligned}
$$

for both averaging and querying

Massively Parallel Path Space Filtering

Linear instead of quadratic

- finding the hash table location i

$$l \leftarrow \text{level\_of\_detail}(|p_{\text{cam}} - x|)$$

$$x' \leftarrow x + \text{jitter}(n) \cdot \textit{scale} \cdot 2^{l}$$

$$l' \leftarrow \text{level\_of\_detail}(|p_{\text{cam}} - x'|)$$

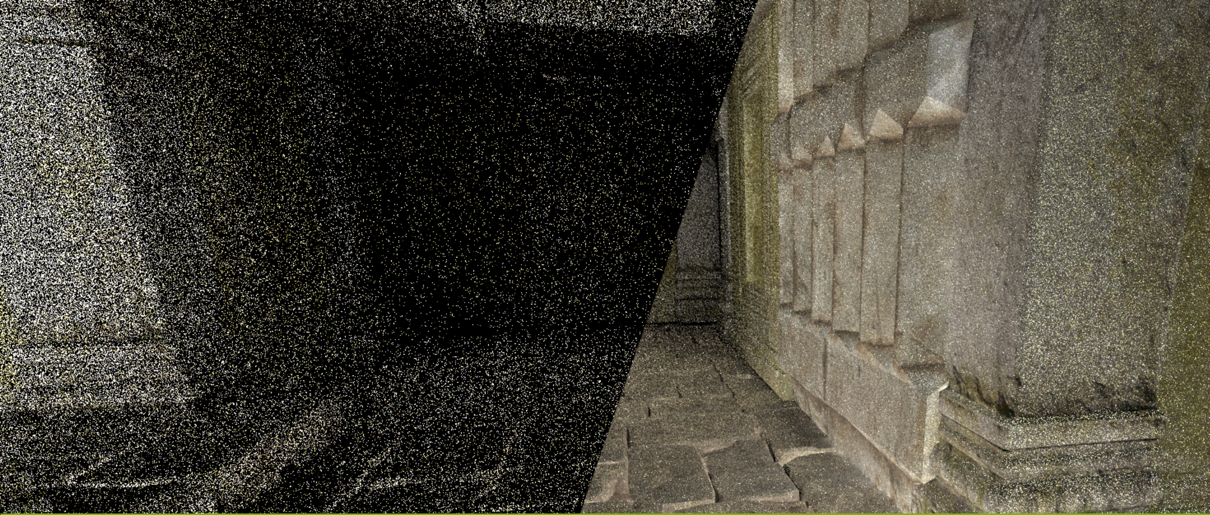
$$\tilde{x} \leftarrow \left\lfloor \frac{x'}{\textit{scale} \cdot 2^{l'}} \right\rfloor$$

$$i \leftarrow \text{hash}(\tilde{x}, \ldots) \ \% \ \text{table\_size}$$

$$v \leftarrow \text{hash2}(\tilde{x}, n, \ldots)$$

  for both averaging and querying

- jittering before quantization hides discretization artifacts in uniform noise



Massively parallel path space filtering at second bounce (2ms@HD)

Massively Parallel Path Space Filtering

Temporal filtering vs. temporal accumulation

- exponential moving average $\tilde{f} = (1-\alpha) f_{old} + \alpha f_{new}$ with constant α
  - will not converge over time
  - lags and blurs over time
  - in fact low pass filter

Massively Parallel Path Space Filtering

Temporal filtering vs. temporal accumulation

- cumulative moving average $\tilde{f} = (1 - \frac{1}{N}) f_{old} + \frac{1}{N} f_{new}$
  - converges over time
  - vanishing adaptivity with increasing number of samples
  - equivalent to exponential average with $\alpha = \frac{1}{N}$

Massively Parallel Path Space Filtering

Temporal filtering vs. temporal accumulation

- exponential moving average $\tilde{f} = (1-\alpha) f_{old} + \alpha f_{new}$ with adaptive α
  - temporal gradients to determine
    - $\alpha = \frac{1}{N}$ if no temporal changes are detected
    - $\alpha \in (\frac{1}{N}, 1]$ depending on the amount of change

Massively parallel path space filtering at first bounce (1ms@HD)

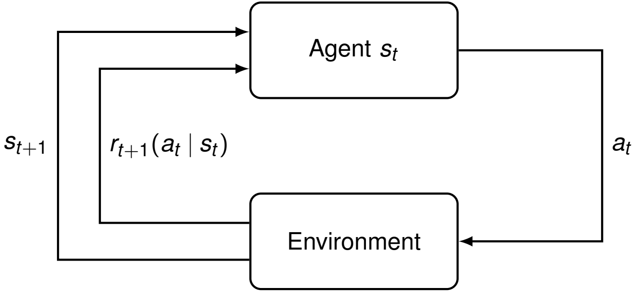
Reinforcement Learning

Reinforcement Learning

Goal: maximize reward

- state transition yields reward

$$r_{t+1}(a_t \mid s_t) \in \mathbb{R}$$



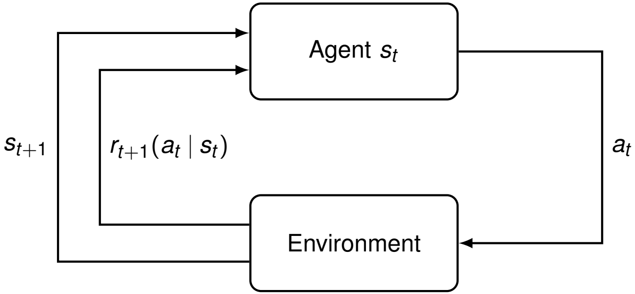
Reinforcement Learning

Goal: maximize reward

- state transition yields reward

$$r_{t+1}(a_t \mid s_t) \in \mathbb{R}$$

- learn a policy π_t
  - to select an action $a_t \in \mathbb{A}(s_t)$
  - given the current state $s_t \in \mathbb{S}$



Reinforcement Learning

Maximize reward by learning importance sampling

- structural equivalence of integral equation and Q-learning

$$
\begin{array}{llllll}
L(x,\omega) & = & L_e(x,\omega) & + \int_{\mathcal{S}_+^2(x)} & f_s(\omega_i,x,\omega)\cos\theta_i & L(h(x,\omega_i),-\omega_i) \quad d\omega_i \\
Q'(s,a) & = (1-\alpha)Q(s,a)+\alpha\Big(& r(s,a) & + \gamma\int_{\mathcal{A}} & \pi(s',a') & Q(s',a') \quad da' \Big)
\end{array}
$$

Reinforcement Learning

Maximize reward by learning importance sampling

- structural equivalence of integral equation and Q-learning

$$
\begin{aligned}
L(x,\omega) &= L_e(x,\omega) + \int_{\mathcal{S}_+^2(x)} f_s(\omega_i, x, \omega)\cos\theta_i \, L(h(x,\omega_i), -\omega_i) \, d\omega_i \\
Q'(s,a) &= (1-\alpha)Q(s,a) + \alpha\left(r(s,a) + \gamma \int_{\mathcal{A}} \pi(s',a') \, Q(s',a') \, da' \right)
\end{aligned}
$$

Reinforcement Learning

Maximize reward by learning importance sampling

- structural equivalence of integral equation and Q-learning

$$
\begin{aligned}
L(x,\omega) &= && L_e(x,\omega) + \int_{\mathcal{S}^2_+(x)} f_s(\omega_i,x,\omega)\cos\theta_i \, L(h(x,\omega_i),-\omega_i) \, d\omega_i \\
Q'(s,a) &= (1-\alpha)Q(s,a)+\alpha\Big(&& r(s,a) + \gamma\int_{\mathcal{A}} \pi(s',a') \, Q(s',a') \, da' \Big)
\end{aligned}
$$

Reinforcement Learning

Maximize reward by learning importance sampling

- structural equivalence of integral equation and Q-learning

$$
\begin{aligned}
L(x,\omega) &= && L_e(x,\omega) &&+ \int_{\mathcal{S}_+^2(x)} && f_s(\omega_i,x,\omega)\cos\theta_i && L(h(x,\omega_i),-\omega_i) && d\omega_i \\
Q'(s,a) &= (1-\alpha)Q(s,a)+\alpha\Big(&& r(s,a) &&+ \gamma\int_{\mathcal{A}} && \pi(s',a') && Q(s',a') && da' \Big)
\end{aligned}
$$

Reinforcement Learning

Maximize reward by learning importance sampling

- structural equivalence of integral equation and Q-learning

$$
\begin{aligned}
L(x,\omega) &= && L_e(x,\omega) + \int_{\mathscr{S}^2_+(x)} f_s(\omega_i,x,\omega)\cos\theta_i \; L(h(x,\omega_i),-\omega_i)\, d\omega_i \\
Q'(s,a) &= (1-\alpha)Q(s,a)+\alpha\Big(&& r(s,a) + \gamma\int_{\mathscr{A}} \pi(s',a') \; Q(s',a')\, da' \Big)
\end{aligned}
$$

Reinforcement Learning

Maximize reward by learning importance sampling

- structural equivalence of integral equation and Q-learning

$$
\begin{aligned}
L(x,\omega) &= && L_e(x,\omega) && + \int_{\mathscr{S}_+^2(x)} && f_s(\omega_i,x,\omega)\cos\theta_i && L(h(x,\omega_i),-\omega_i) && d\omega_i \\
Q'(s,a) &= (1-\alpha)Q(s,a)+\alpha\Big(&& r(s,a) && +\gamma\int_{\mathscr{A}} && \pi(s',a') && Q(s',a') && da'\Big)
\end{aligned}
$$

- graphics example: learning the incident radiance

$$
Q'(x,\omega) = (1-\alpha)Q(x,\omega) + \alpha\left(L_e(y,-\omega) + \int_{\mathscr{S}_+^2(y)} f_s(\omega_i,y,-\omega)\cos\theta_i Q(y,\omega_i)d\omega_i\right)
$$

Reinforcement Learning

Maximize reward by learning importance sampling

- structural equivalence of integral equation and Q-learning

$$
\begin{array}{lllllll}
L(x,\omega) & = & L_e(x,\omega) & + \int_{\mathcal{S}_+^2(x)} & f_s(\omega_i,x,\omega)\cos\theta_i & L(h(x,\omega_i),-\omega_i) & d\omega_i \\
Q'(s,a) & = (1-\alpha)Q(s,a)+\alpha\Big(& r(s,a) & + \gamma\int_{\mathcal{A}} & \pi(s',a') & Q(s',a') & da'\Big)
\end{array}
$$

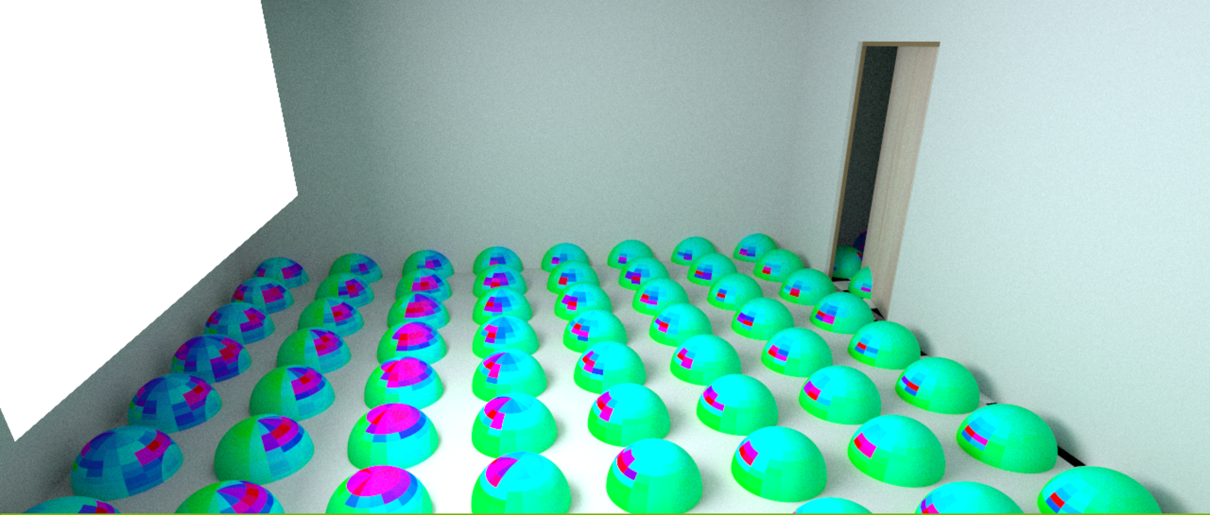
- graphics example: learning the incident radiance

$$
Q'(x,\omega) = (1-\alpha)Q(x,\omega) + \alpha\left(L_e(y,-\omega) + \int_{\mathcal{S}_+^2(y)} f_s(\omega_i,y,-\omega)\cos\theta_i Q(y,\omega_i)d\omega_i\right)
$$

to be used as a policy for selecting an action ω in state x to reach the next state $y := h(x,\omega)$

- the learning rate α is the only parameter left

▶ Technical Note: Q-Learning



approximate solution Q stored on discretized hemispheres across scene surface

2048 paths traced with BRDF importance sampling in a scene with challenging visibility

Path tracing with online reinforcement learning at the same number of paths

Metropolis light transport at the same number of paths

Reinforcement Learning

Principle

- radiance L is light sources L_e plus transported radiance $T_f L$

$$L = L_e + T_f L$$

Reinforcement Learning

Principle

- radiance L is light sources L_e plus transported radiance $T_f L$

$$L = L_e + T_f L$$

- use an approximation, e.g. discretized hemispheres at selected locations in scene to store

$$L_c = L_e + T_f L_c$$

Reinforcement Learning

Principle

- radiance L is light sources L_e plus transported radiance $T_f L$

$$L = L_e + T_f L$$

- use an approximation, e.g. discretized hemispheres at selected locations in scene to store

$$L_c = L_e + T_f L_c$$

  for guiding importance sampling towards where the radiance comes from

Reinforcement Learning

Principle

- radiance L is light sources L_e plus transported radiance $T_f L$

$$L = L_e + T_f L$$

- use an approximation, e.g. discretized hemispheres at selected locations in scene to store

$$L_c = L_e + T_f L_c$$

  for guiding importance sampling towards where the radiance comes from

- learn the approximation by

$$L'_c = (1-\alpha)L_c + \alpha(L_e + T_f L_c)$$

Reinforcement Learning
Principle

- radiance L is light sources L_e plus transported radiance $T_f L$

$$L = L_e + T_f L$$

- use an approximation, e.g. discretized hemispheres at selected locations in scene to store

$$L_c = L_e + T_f L_c$$

  for guiding importance sampling towards where the radiance comes from

- learn the approximation by

$$L'_c = (1-\alpha)L_c + \alpha(L_e + T_f L_c) = (1-\alpha)L_c + \alpha\left(L_e + \sum f_{r,i} L_{c,i}\right)$$

  using the **current approximation instead of tracing single paths at higher variance**

Reinforcement Learning

Principle

- shorter expected path length
- dramatically reduced number of paths with zero contribution

Reinforcement Learning

Principle

- shorter expected path length
- dramatically reduced number of paths with zero contribution
- challenges
  - product importance sampling proportional to the integrand, i.e. policy $\gamma \cdot \pi$ times value Q
  - efficient representation of value Q

▶ S9900 - Irradiance Fields: RTX Diffuse Global Illumination for Local and Cloud Graphics

▶ Learning light transport the reinforced way

▶ Machine learning and integral equations

▶ Neural importance sampling

Photon-Guided Shadow Rays

Photon-Guided Shadow Rays

Photon maps similar to massively parallel path space filtering

- incorporate knowledge about visibility to improve efficiency
- control the number of shadow rays
- look up photons around a shading point and trace shadow rays towards their origin
  - photon origins used as virtual point light sources (VPL)

Light hierarchy

Photon-guided shadow rays (PGSR)

PGSR + single pass screen space PSF

Point Clouds

Stochastically hashed particle maps

- on hash collision keep particle with the smallest random number and increment cell counter
- issue of wide bit-width memory access on GPU
- hash table size vs. quantization vs. uniformity of hash function

  large hash table size: less collisions, totally divergent memory access

  small hash table size: more collisions, automatically thinning particles in dense regions

Generic material model to reduce divergence

one BSDF model fed by parameter point cloud

Ray Traced Global Illumination for Games

Building blocks

- massively parallel path space filtering
- efficient light transport simulation by reinforcement learning
- photon-guided shadow rays

▶ S9900 - Irradiance Fields: RTX Diffuse Global Illumination for Local and Cloud Graphics

▶ Gradient estimation for real-time adaptive temporal filtering

▶ Massively parallel path space filtering