Learning Light Transport the Reinforced Way

Ken Dahm and Alexander Keller

NVIDIA

Light Transport Simulation

How to do importance sampling

- compute functionals of a Fredholm integral equation of the 2nd kind

$$L(x,\omega) \quad = \quad L_e(x,\omega) + \int_{\mathscr{S}^2_+(x)} L(h(x,\omega_i),-\omega_i) f_r(\omega_i,x,\omega) \cos\theta_i d\omega_i$$

NVIDIA.

Light Transport Simulation

How to do importance sampling

- example: direct illumination

$$L(x,\omega) \quad = \quad L_e(x,\omega) + \int_{\mathscr{S}^2_+(x)} L_e(h(x,\omega_i),-\omega_i) f_r(\omega_i,x,\omega) \cos\theta_i d\omega_i$$

NVIDIA.

Light Transport Simulation

How to do importance sampling

- example: direct illumination

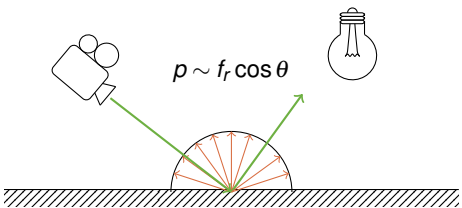
$$
\begin{aligned}
L(x,\omega) &= L_e(x,\omega) + \int_{\mathscr{S}^2_+(x)} L_e(h(x,\omega_i),-\omega_i) f_r(\omega_i,x,\omega)\cos\theta_i d\omega_i \\
&\approx L_e(x,\omega) + \frac{1}{N}\sum_{i=0}^{N-1} \frac{L_e(h(x,\omega_i),-\omega_i) f_r(\omega_i,x,\omega)\cos\theta_i}{p(\omega_i)}
\end{aligned}
$$

Light Transport Simulation

How to do importance sampling

- example: direct illumination

$$
\begin{aligned}
L(x,\omega) &= L_e(x,\omega) + \int_{\mathscr{S}^2_+(x)} L_e(h(x,\omega_i),-\omega_i) f_r(\omega_i,x,\omega)\cos\theta_i d\omega_i \\
&\approx L_e(x,\omega) + \frac{1}{N}\sum_{i=0}^{N-1} \frac{L_e(h(x,\omega_i),-\omega_i) f_r(\omega_i,x,\omega)\cos\theta_i}{p(\omega_i)}
\end{aligned}
$$

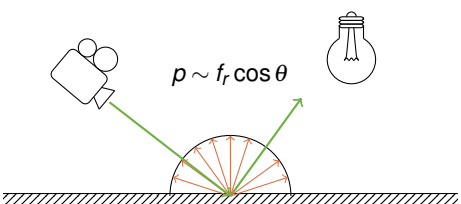


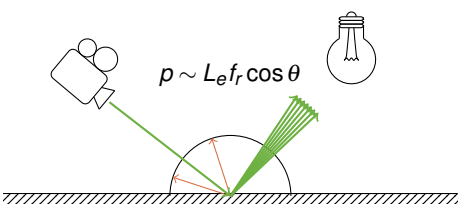
Light Transport Simulation

How to do importance sampling

- example: direct illumination

$$
\begin{aligned}
L(x,\omega) &= L_e(x,\omega) + \int_{\mathcal{S}^2_+(x)} L_e(h(x,\omega_i),-\omega_i) f_r(\omega_i,x,\omega)\cos\theta_i d\omega_i \\
&\approx L_e(x,\omega) + \frac{1}{N}\sum_{i=0}^{N-1} \frac{L_e(h(x,\omega_i),-\omega_i) f_r(\omega_i,x,\omega)\cos\theta_i}{p(\omega_i)}
\end{aligned}
$$





NVIDIA.

Machine Learning

Machine Learning

Taxonomy

supervised learning: learning from labeled data

goal: extrapolate/generalize response to unseen data

example: artificial neural networks

Machine Learning

Taxonomy

supervised learning: learning from labeled data

goal: extrapolate/generalize response to unseen data

example: artificial neural networks

unsupervised learning: learning from unlabeled data

goal: identify structure in data

example: autoencoder networks

Machine Learning

Taxonomy

supervised learning: learning from labeled data

goal: extrapolate/generalize response to unseen data

example: artificial neural networks

unsupervised learning: learning from unlabeled data

goal: identify structure in data

example: autoencoder networks

semi-supervised learning: reward-based supervision

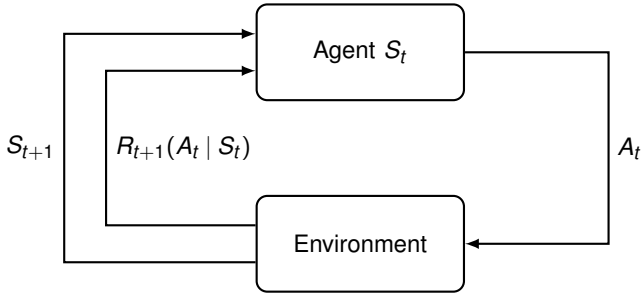
goal: maximize reward

example: reinforcement learning

The Reinforcement Learning Problem

Maximize reward

- policy $\pi_t : \mathbb{S} \to \mathbb{A}(S_t)$
  - to select an action $A_t \in \mathbb{A}(S_t)$
  - given current state $S_t \in \mathbb{S}$



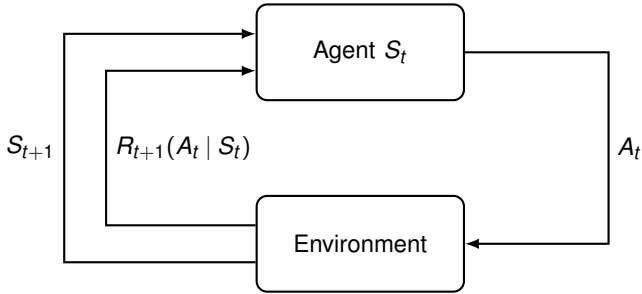
NVIDIA.

The Reinforcement Learning Problem

Maximize reward

- policy $\pi_t : \mathbb{S} \to \mathbb{A}(S_t)$
  - to select an action $A_t \in \mathbb{A}(S_t)$
  - given current state $S_t \in \mathbb{S}$
- state transition yields reward

$$R_{t+1}(A_t \mid S_t) \in \mathbb{R}$$



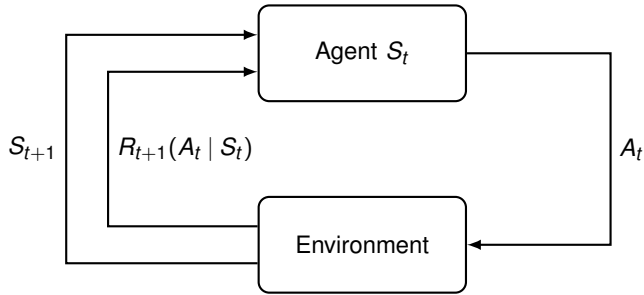
NVIDIA.

The Reinforcement Learning Problem

Maximize reward

- policy $\pi_t : \mathbb{S} \to \mathbb{A}(S_t)$
  - to select an action $A_t \in \mathbb{A}(S_t)$
  - given current state $S_t \in \mathbb{S}$
- state transition yields reward

$$R_{t+1}(A_t \mid S_t) \in \mathbb{R}$$



- classic goal: find policy to maximize discounted cumulative reward

$$V^{\pi}(S_t) \equiv \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}(A_{t+k} \mid S_{t+k}), \text{ where } 0 < \gamma < 1$$

NVIDIA.

The Reinforcement Learning Problem

Q-Learning [Watkins 1989]

- finds optimal action-selection policy for any given Markov decision process

$$Q'(s,a) \quad = \quad (1-\alpha) \cdot Q(s,a) + \alpha \cdot \left(r(s,a) + \gamma V(s')\right) \text{ for a learning rate } \alpha \in [0,1]$$

The Reinforcement Learning Problem

Q-Learning [Watkins 1989]

- finds optimal action-selection policy for any given Markov decision process

$$Q'(s,a) \quad = \quad (1-\alpha) \cdot Q(s,a) + \alpha \cdot \left(r(s,a) + \gamma V(s')\right) \text{ for a learning rate } \alpha \in [0,1]$$

with the following options for the discounted cumulative reward

$$V(s') \quad \equiv \quad \begin{cases} \max_{a' \in \mathbb{A}} Q(s',a') & \text{consider best action} \\ \\ \\ \\ \end{cases}$$

NVIDIA.

The Reinforcement Learning Problem

Q-Learning [Watkins 1989]

- finds optimal action-selection policy for any given Markov decision process

$$Q'(s,a) \quad = \quad (1-\alpha) \cdot Q(s,a) + \alpha \cdot \left(r(s,a) + \gamma V(s')\right) \text{ for a learning rate } \alpha \in [0,1]$$

with the following options for the discounted cumulative reward

$$V(s') \quad \equiv \quad \begin{cases} \max_{a' \in \mathbb{A}} Q(s',a') & \text{consider best action} \\ \sum_{a' \in \mathbb{A}} \pi(s',a') Q(s',a') & \text{policy weighted average over discrete action space} \end{cases}$$

The Reinforcement Learning Problem

Q-Learning [Watkins 1989]

- finds optimal action-selection policy for any given Markov decision process

$$Q'(s,a) \quad = \quad (1-\alpha) \cdot Q(s,a) + \alpha \cdot \left(r(s,a) + \gamma V(s') \right) \text{ for a learning rate } \alpha \in [0,1]$$

with the following options for the discounted cumulative reward

$$V(s') \quad \equiv \quad \begin{cases} \max_{a' \in \mathbb{A}} Q(s',a') & \text{consider best action} \\ \sum_{a' \in \mathbb{A}} \pi(s',a') Q(s',a') & \text{policy weighted average over discrete action space} \\ \int_{\mathbb{A}} \pi(s',a') Q(s',a') da' & \text{policy weighted average over continuous action space} \end{cases}$$

Light Transport Simulation and Reinforcement Learning

Light Transport Simulation and Reinforcement Learning

Structural equivalence of the integral equations

- matching terms

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

Light Transport Simulation and Reinforcement Learning

Structural equivalence of the integral equations

- matching terms

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

Light Transport Simulation and Reinforcement Learning

Structural equivalence of the integral equations

- matching terms

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

Light Transport Simulation and Reinforcement Learning

Structural equivalence of the integral equations

- matching terms

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

Light Transport Simulation and Reinforcement Learning

Structural equivalence of the integral equations

- matching terms

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

Light Transport Simulation and Reinforcement Learning

Structural equivalence of the integral equations

- matching terms

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

hints at learning the incident radiance

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

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

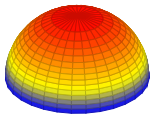
- the learning rate α is the only parameter left

Light Transport Simulation and Reinforcement Learning

Discretization of Q in analogy to irradiance representations

- action space: $a \in \mathscr{S}^2_+(y)$
  - equally sized patches

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \sqrt{1-u^2}\cos(2\pi v) \\ \sqrt{1-u^2}\sin(2\pi v) \\ u \end{pmatrix}$$



Light Transport Simulation and Reinforcement Learning

Discretization of Q in analogy to irradiance representations

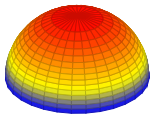
- action space: $a \in \mathscr{S}^2_+(y)$
  - equally sized patches

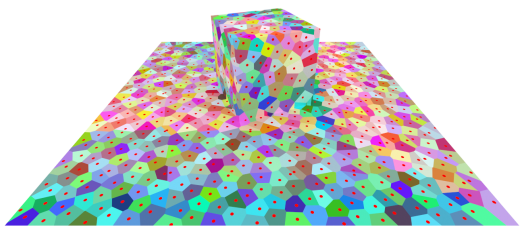
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \sqrt{1-u^2}\cos(2\pi v) \\ \sqrt{1-u^2}\sin(2\pi v) \\ u \end{pmatrix}$$

- state space: $s \in dV$
  - Voronoi diagram of low-discrepancy sequence

$$\mathbf{x}_i = \begin{pmatrix} \Phi_2(i) \\ i/N \end{pmatrix} \text{ for } i = 0, ..., N-1$$

  - nearest neighbor search



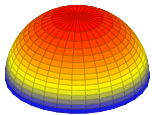


Light Transport Simulation and Reinforcement Learning

Discretization of Q in analogy to irradiance representations

- action space: $a \in \mathscr{S}^2_+(y)$
  - equally sized patches

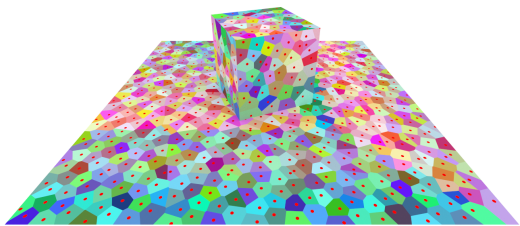
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \sqrt{1-u^2}\cos(2\pi v) \\ \sqrt{1-u^2}\sin(2\pi v) \\ u \end{pmatrix}$$

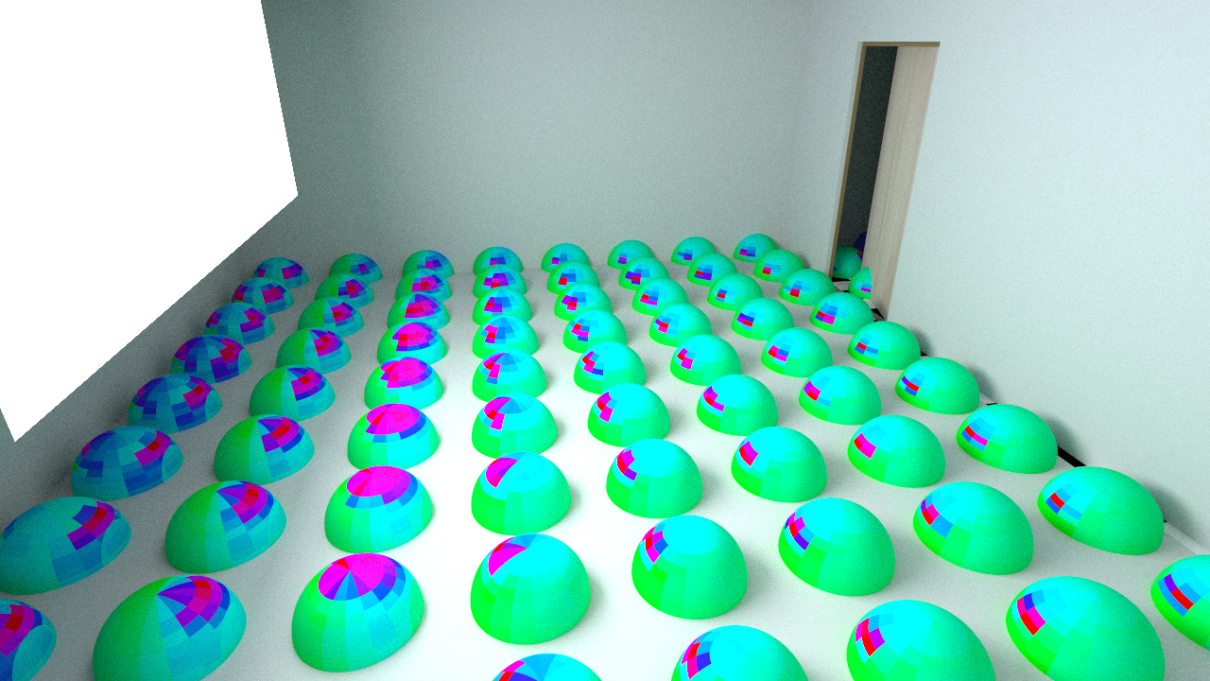


- state space: $s \in dV$
  - Voronoi diagram of low-discrepancy sequence

$$\mathbf{x}_i = \begin{pmatrix} \Phi_2(i) \\ i/N \end{pmatrix} \text{ for } i = 0, ..., N-1$$

  - nearest neighbor search including surface normal





Learning Light Transport the Reinforced Way

Online reinforcement learning algorithm for guiding light transport paths

Function *trace(camera, pixel, scene)*

- *throughput* $\leftarrow 1$
- *color* $\leftarrow 0$
- *ray* $\leftarrow$ setupPrimaryRay(*camera*, *pixel*)
- **for** $i \leftarrow 0$ **to** ∞ **do**
  - $h, n \leftarrow$ intersect(*scene*, *ray*)
  - **if** *isAreaLight(h)* **then**
    - *color* $\leftarrow$ *throughput* * getEmission(h)
    - break
  - f_s, ω, $p_\omega \leftarrow$ sampleBsdf(h, n)
  - *throughput* $\leftarrow$ *throughput* * f_s * $\cos(n, \omega)$ / p_ω
  - *ray* $\leftarrow h, \omega$
- write(*pixel*, *color*)

NVIDIA.

Learning Light Transport the Reinforced Way

Online reinforcement learning algorithm for guiding light transport paths

```
Function trace(camera, pixel, scene)
    throughput ← 1
    color ← 0
    ray ← setupPrimaryRay(camera, pixel)
    for i ← 0 to ∞ do
        h, n ← intersect(scene, ray)
        if i > 0 then
            updateQtable(ray.o, h)
        if isAreaLight(h) then
            color ← throughput * getEmission(h)
            break
        f_s, ω, p_ω ← sampleScatteringDir(h)
        throughput ← throughput * f_s * cos(n, ω) / p_ω
        ray ← h, ω
    write(pixel, color)
```

Learning Light Transport the Reinforced Way

Online reinforcement learning algorithm for guiding light transport paths

```
Function trace(camera, pixel, scene)
    throughput ← 1
    color ← 0
    ray ← setupPrimaryRay(camera, pixel)
    for i ← 0 to ∞ do
        h, n ← intersect(scene, ray)
        if i > 0 then
            updateQtable(ray.o, h)
        if isAreaLight(h) then
            color ← throughput * getEmission(h)
            break
        f_s, ω, p_ω ← sampleScatteringDir(h)
        throughput ← throughput * f_s * cos(n, ω) / p_ω
        ray ← h, ω
    write(pixel, color)
```

```
Function updateQtable(ray, h)
    idxPrev ← findCell(ray.o)
    idxCurr ← findCell(h)
    update ← 0
    if isAreaLight(h) then
        update ← max(getEmission(h))
    else
        update ← max(getLastAttenuation(ray) * qmax[idxCurr])
    idxQ ← findIndex(idxPrev, ray)
    qtable[idxQ] ← (1 − α) ∗ qtable[idxQ] + α ∗ update
```

Learning Light Transport the Reinforced Way

Online reinforcement learning algorithm for guiding light transport paths

Function *trace(camera, pixel, scene)*

- *throughput* ← 1
- *color* ← 0
- *ray* ← setupPrimaryRay(*camera*, *pixel*)
- **for** $i \leftarrow 0$ **to** ∞ **do**
  - $h, n \leftarrow$ intersect(*scene*, *ray*)
  - **if** $i > 0$ **then**
    - **updateQtable**(*ray.o*, *h*)
  - **if** *isAreaLight(h)* **then**
    - *color* ← *throughput* * getEmission(*h*)
    - break
  - f_s, ω, p_ω ← **sampleScatteringDir**(*h*)
  - *throughput* ← *throughput* * f_s * cos(n, ω) / p_ω
  - *ray* ← *h*, ω
- write(*pixel*, *color*)

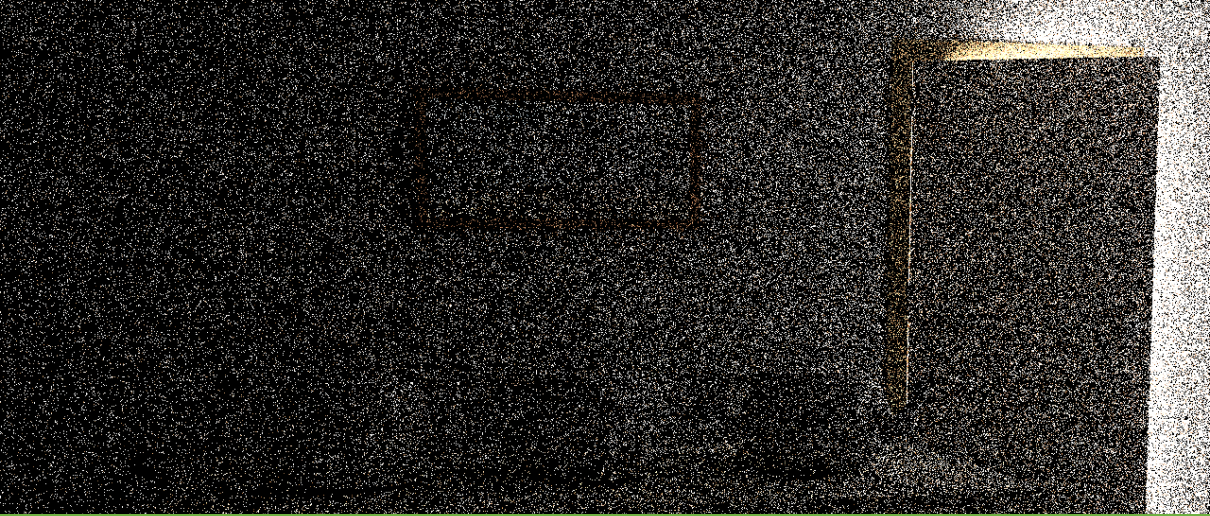
Function *updateQtable(ray, h)*

- *idxPrev* ← findCell(*ray.o*)
- *idxCurr* ← findCell(*h*)
- *update* ← 0
- **if** *isAreaLight(h)* **then**
  - *update* ← max(getEmission(*h*))
- **else**
  - *update* ← max(getLastAttenuation(*ray*) * *qmax*[*idxCurr*])
- *idxQ* ← findIndex(*idxPrev*, *ray*)
- $qtable[idxQ] \leftarrow (1-\alpha) * qtable[idxQ] + \alpha * update$

Function *sampleScatteringDir(h)*

- *idx* ← findCell(*h*)
- *idxPatch*, p_s ← sampleCdf(*idx*)
- $\omega \leftarrow$ *uniformSamplePatch*(*idxPatch*)
- return $f_s, \omega, p_s * numpatches/(2*\pi)$

Results



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

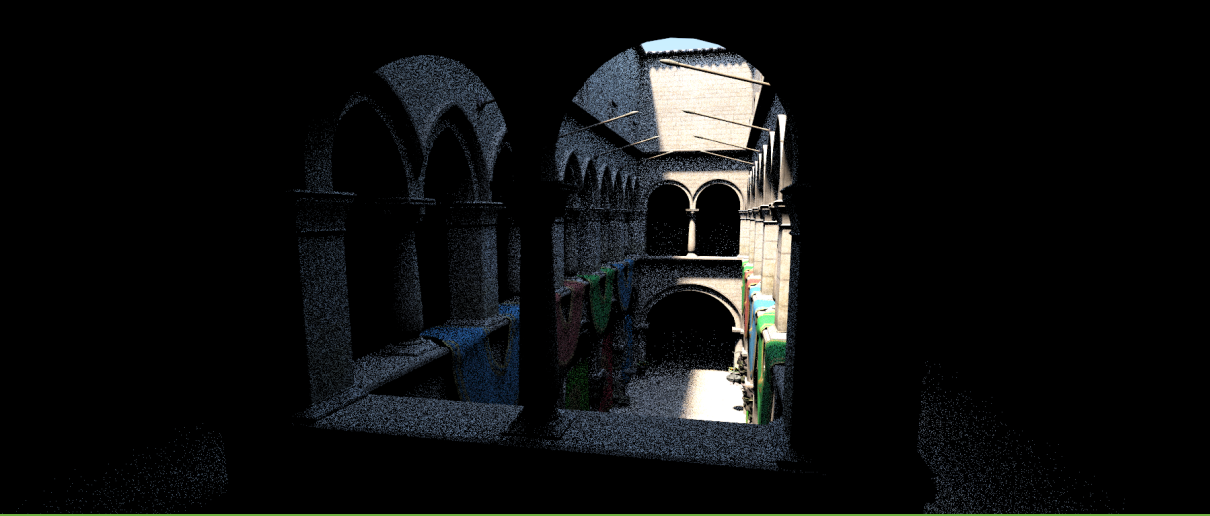
Path tracing with online reinforcement learning at the same number of paths

Metropolis light transport at the same number of paths

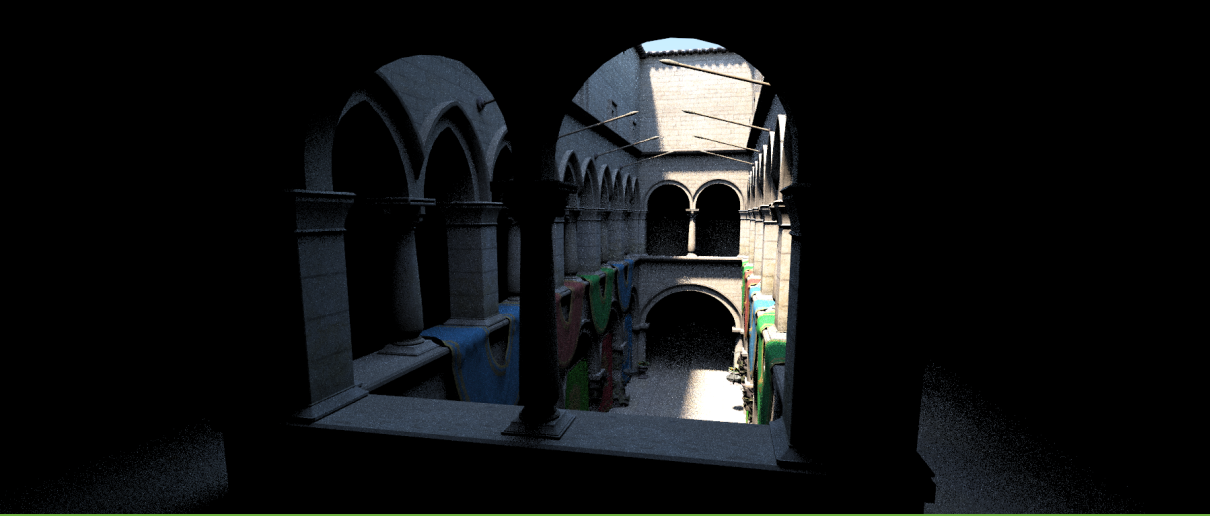
2048 paths traced with BRDF importance sampling in a scene with challenging visibility

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

Metropolis light transport at the same number of paths



Paths tracing with next event estimation and BRDF importance sampling



Path tracing with next event estimation and online reinforcement learning
at the same number of samples

Learning Light Transport the Reinforced Way

Summary

- structural identity of reinforcement learning and a Fredholm integral equation of the 2nd kind
- dramatic variance reduction
  - simple to implement using data structures common in real-time games
  - shorter path length
  - more coherent path generation/traversal
- investigating interaction with Russian roulette absorption and splitting
  - as average path length is shortened already
- investigating high dimensional function approximation via neural networks
- https://arxiv.org/abs/1701.07403