NVIDIA

East Building, Ballroom BC

nvidia.com/siggraph2018

Machine Learning and Rendering

Alex Keller, Director of Research

Machine Learning and Rendering

Course web page at `https://sites.google.com/site/mlandrendering/`

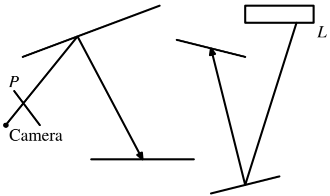
- 14:00 From Machine Learning to Graphics and back
  - Alexander Keller, NVIDIA
- 14:40 Robust & Efficient Light Transport by Machine Learning
  - Jaroslav Křivánek, Charles University, Prague
- 15:15 Deep Learning for Light Transport Simulation
  - Jan Novàk, Disney Research
- 16:05 Neural Realtime Rendering in Image Space
  - Anton Kaplanyan, Facebook Reality Labs
- 16:40 Deep Realtime Rendering
  - Marco Salvi, NVIDIA

Modern Path Tracing

Light transport simulation

- ways to formulate the radiance L_r reflected in a surface point x

$$L_r(x, \omega_r) = \int_{\mathscr{S}^2_-(x)} L_i(x, \omega) f_r(\omega_r, x, \omega) \cos\theta_x \, d\omega$$

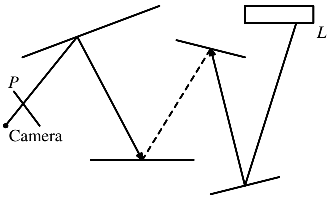


Modern Path Tracing

Light transport simulation

- ways to formulate the radiance L_r reflected in a surface point x

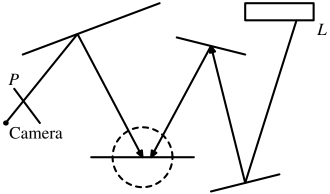
$$\begin{aligned} L_r(x, \omega_r) \\ &= \int_{\mathcal{S}_-^2(x)} L_i(x, \omega) f_r(\omega_r, x, \omega) \cos\theta_x \, d\omega \\ &= \int_{\partial V} V(x, y) L_i(x, \omega) f_r(\omega_r, x, \omega) \cos\theta_x \frac{\cos\theta_y}{|x-y|^2} dy \end{aligned}$$



Modern Path Tracing

Light transport simulation

- ways to formulate the radiance L_r reflected in a surface point x



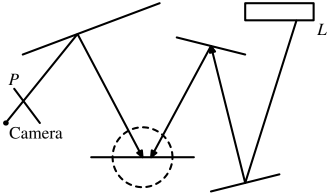
$$
\begin{aligned}
L_r(x, \omega_r) \\
&= \int_{\mathscr{S}^2_-(x)} L_i(x, \omega) f_r(\omega_r, x, \omega) \cos\theta_x \, d\omega \\
&= \int_{\partial V} V(x, y) L_i(x, \omega) f_r(\omega_r, x, \omega) \cos\theta_x \frac{\cos\theta_y}{|x - y|^2} dy \\
&= \int_{\partial V} \int_{\partial V} V(x', y) \delta_x(x') L_i(x', \omega) f_r(\omega_r, x', \omega) \cos\theta_{x'} \frac{\cos\theta_y}{|x' - y|^2} dx' dy
\end{aligned}
$$

NVIDIA.

Modern Path Tracing

Light transport simulation

- ways to formulate the radiance L_r reflected in a surface point x

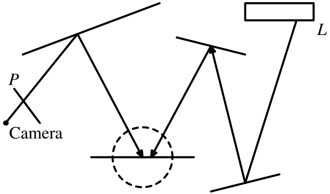


$$
\begin{aligned}
L_r(x, \omega_r) \\
&= \int_{\mathcal{S}^2_-(x)} L_i(x, \omega) f_r(\omega_r, x, \omega) \cos\theta_x \, d\omega \\
&= \int_{\partial V} V(x, y) L_i(x, \omega) f_r(\omega_r, x, \omega) \cos\theta_x \frac{\cos\theta_y}{|x - y|^2} dy \\
&= \int_{\partial V} \int_{\partial V} V(x', y) \left(\lim_{r(x) \to 0} \frac{\chi_B(x - x')}{\pi r(x)^2} \right) L_i(x', \omega) f_r(\omega_r, x', \omega) \cos\theta_{x'} \frac{\cos\theta_y}{|x' - y|^2} dx' dy
\end{aligned}
$$

Modern Path Tracing

Light transport simulation

- ways to formulate the radiance L_r reflected in a surface point x

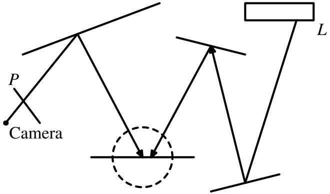


$$
\begin{aligned}
L_r(x, \omega_r) \\
&= \int_{\mathcal{S}_-^2(x)} L_i(x,\omega) f_r(\omega_r, x, \omega) \cos\theta_x \, d\omega \\
&= \int_{\partial V} V(x,y) L_i(x,\omega) f_r(\omega_r, x, \omega) \cos\theta_x \frac{\cos\theta_y}{|x-y|^2} dy \\
&= \lim_{r(x)\to 0} \int_{\partial V} \int_{\partial V} V(x',y) \frac{\chi_B(x-x')}{\pi r(x)^2} L_i(x',\omega) f_r(\omega_r, x', \omega) \cos\theta_y \frac{\cos\theta_{x'}}{|x'-y|^2} dx' \, dy
\end{aligned}
$$

Modern Path Tracing

Light transport simulation

- ways to formulate the radiance L_r reflected in a surface point x

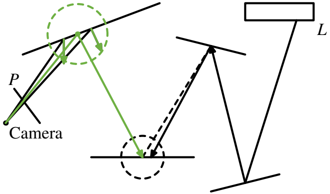


$$\begin{aligned} L_r(x,\omega_r) &= \int_{\mathscr{S}_-^2(x)} L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x \, d\omega \\ &= \int_{\partial V} V(x,y) L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x \frac{\cos\theta_y}{|x-y|^2} dy \\ &= \lim_{r(x)\to 0} \int_{\partial V} \int_{\mathscr{S}_-^2(y)} \frac{\chi_B(x-h(y,\omega))}{\pi r(x)^2} L_i(h(y,\omega),\omega) f_r(\omega_r,h(y,\omega),\omega)\cos\theta_y \, d\omega \, dy \end{aligned}$$

Modern Path Tracing

Light transport simulation

- ways to formulate the radiance L_r reflected in a surface point x

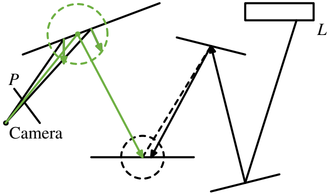


$$
\begin{aligned}
&L_r(x,\omega_r)\\
&= \int_{\mathscr{S}^2_-(x)} L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x\, d\omega\\
&= \int_{\partial V} V(x,y) L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x \frac{\cos\theta_y}{|x-y|^2}\, dy\\
&= \lim_{r(x)\to 0} \int_{\partial V}\int_{\mathscr{S}^2_-(y)} \frac{\chi_B(x-h(y,\omega))}{\pi r(x)^2} L_i(h(y,\omega),\omega) f_r(\omega_r,h(y,\omega),\omega)\cos\theta_y\, d\omega\, dy\\
&= \int_{\mathscr{S}^2_-(x)} L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x\, d\omega
\end{aligned}
$$

Modern Path Tracing

Light transport simulation

- ways to formulate the radiance L_r reflected in a surface point x

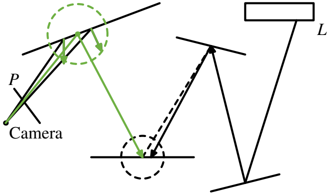


$$
\begin{aligned}
&L_r(x,\omega_r) \\
&= \int_{\mathscr{S}^2_-(x)} L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x \, d\omega \\
&= \int_{\partial V} V(x,y) L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x \frac{\cos\theta_y}{|x-y|^2} dy \\
&= \lim_{r(x)\to 0} \int_{\partial V} \int_{\mathscr{S}^2_-(y)} \frac{\chi_B(x-h(y,\omega))}{\pi r(x)^2} L_i(h(y,\omega),\omega) f_r(\omega_r,h(y,\omega),\omega)\cos\theta_y \, d\omega \, dy \\
&= \int_{\mathscr{S}^2_-(x)} \left(\lim_{r(x)\to 0} \frac{\int_{B(x)} w(x,x') L_i(x',\omega) dx'}{\int_{B(x)} w(x,x') dx'} \right) f_r(\omega_r,x,\omega)\cos\theta_x \, d\omega
\end{aligned}
$$

Modern Path Tracing

Light transport simulation

- ways to formulate the radiance L_r reflected in a surface point x

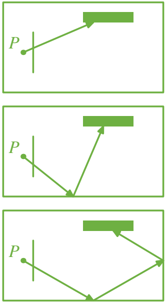


$$\begin{aligned}
L_r(x,\omega_r) &= \int_{\mathscr{S}^2_-(x)} L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x\,d\omega \\
&= \int_{\partial V} V(x,y) L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x \frac{\cos\theta_y}{|x-y|^2}\,dy \\
&= \lim_{r(x)\to 0} \int_{\partial V}\int_{\mathscr{S}^2_-(y)} \frac{\chi_B(x-h(y,\omega))}{\pi r(x)^2} L_i(h(y,\omega),\omega) f_r(\omega_r,h(y,\omega),\omega)\cos\theta_y\,d\omega\,dy \\
&= \lim_{r(x)\to 0} \int_{\mathscr{S}^2_-(x)} \frac{\int_{B(x)} w(x,x') L_i(x',\omega)\,dx'}{\int_{B(x)} w(x,x')\,dx'} f_r(\omega_r,x,\omega)\cos\theta_x\,d\omega
\end{aligned}$$

Modern Path Tracing

Light transport simulation

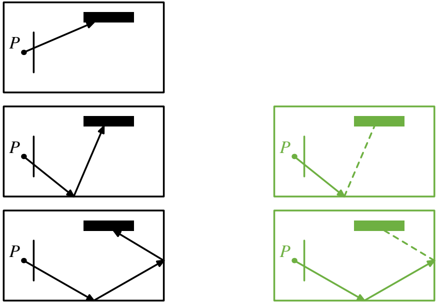
- path tracing: Starting paths from camera and iterating scattering and ray tracing
  - bad for small light sources, good for large light sources



Modern Path Tracing

Light transport simulation

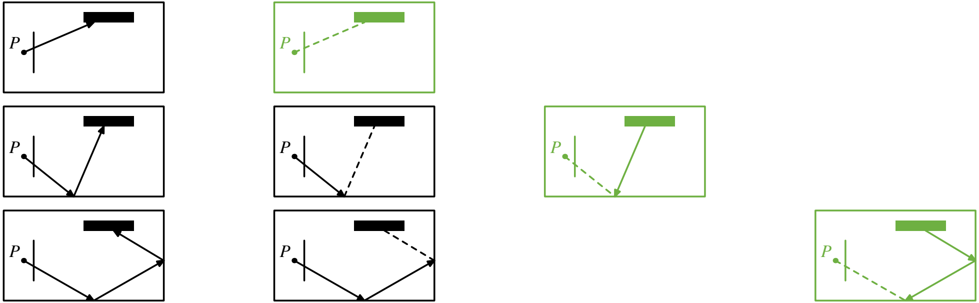
- path tracing with next event estimation by shadow rays (dashed lines)
  - good for small light sources, bad for close light sources



Modern Path Tracing

Light transport simulation

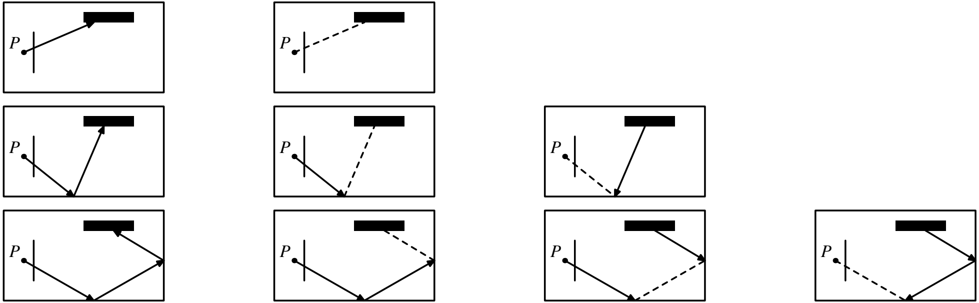
- light tracing, i.e. paths starting from the light source connected to the camera
  - can capture some caustics, where path tracing and next event estimation do not work



Modern Path Tracing

Light transport simulation

- all obvious ways to generate light transport paths
  - which ones are good ?

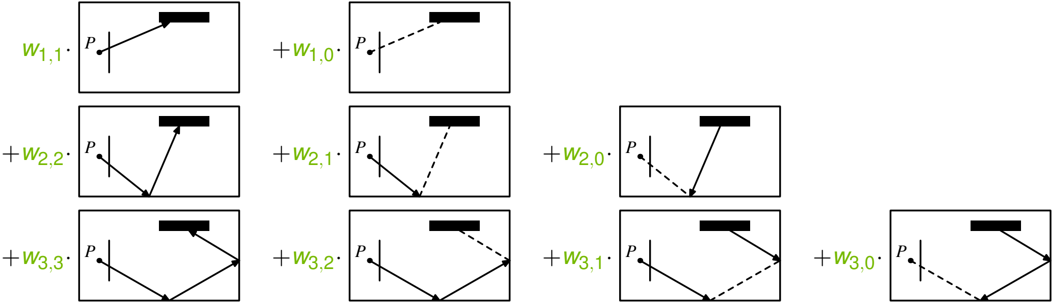


NVIDIA.

Modern Path Tracing

Light transport simulation

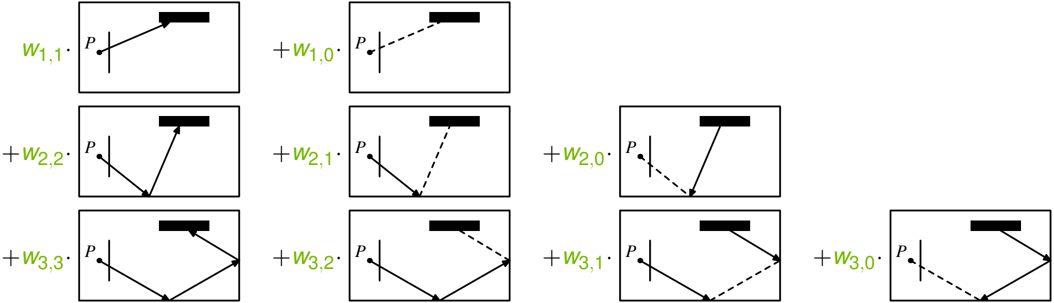
- bidirectional path tracing, optimally combining all techniques by weighting each contribution
  - $\sum_{i=0}^{l} w_{l,i} = 1$ for path length $l-1$, $l \in \mathbb{N}$



Modern Path Tracing

Light transport simulation

- bidirectional path tracing, optimally combining all techniques by weighting each contribution
  - $\sum_{i=0}^{l} w_{l,i} = 1$ for path length $l-1$, $l \in \mathbb{N}$



- problem of insufficient techniques, for example, if only one $w_{l,i} \neq 0$

Modern Path Tracing

Numerical integro-approximation

- Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)\,dx$$

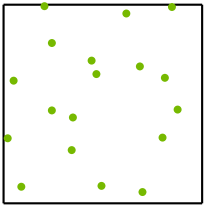
Modern Path Tracing

Numerical integro-approximation

- Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



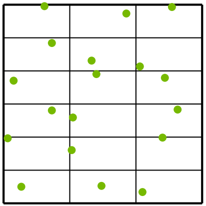
Modern Path Tracing

Numerical integro-approximation

- Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



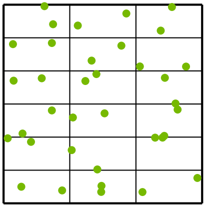
Modern Path Tracing

Numerical integro-approximation

- Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



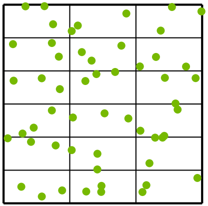
Modern Path Tracing

Numerical integro-approximation

- Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



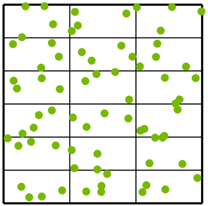
Modern Path Tracing

Numerical integro-approximation

- Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



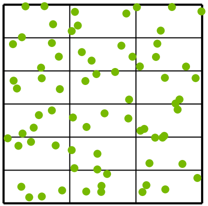
Modern Path Tracing

Numerical integro-approximation

- Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



- quasi-Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

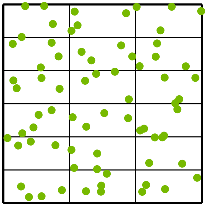
Modern Path Tracing

Numerical integro-approximation

- Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

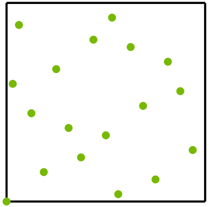
  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



- quasi-Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

  - much more uniform correlated samples x_i
  - realized by low-discrepancy sequences, which are progressive Latin-hypercube samples



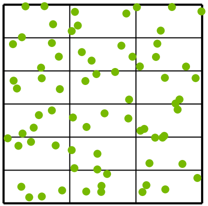
Modern Path Tracing

Numerical integro-approximation

- Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

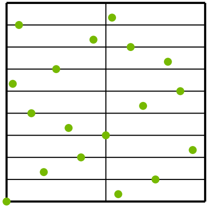
  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



- quasi-Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

  - much more uniform correlated samples x_i
  - realized by low-discrepancy sequences, which are progressive Latin-hypercube samples



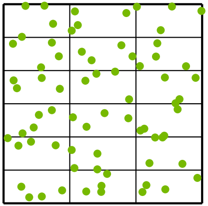
Modern Path Tracing

Numerical integro-approximation

- Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

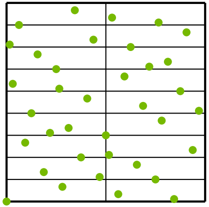
  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



- quasi-Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

  - much more uniform correlated samples x_i
  - realized by low-discrepancy sequences, which are progressive Latin-hypercube samples



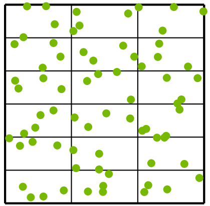
Modern Path Tracing

Numerical integro-approximation

- Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

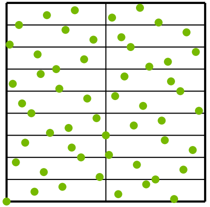
  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



- quasi-Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

  - much more uniform correlated samples x_i
  - realized by low-discrepancy sequences, which are progressive Latin-hypercube samples



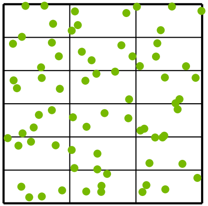
Modern Path Tracing

Numerical integro-approximation

- Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

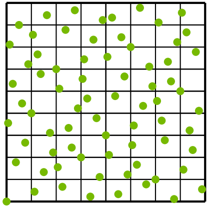
  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



- quasi-Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

  - much more uniform correlated samples x_i
  - realized by low-discrepancy sequences, which are progressive Latin-hypercube samples



Modern Path Tracing

Pushbutton paradigm

- deterministic
  - may improve speed of convergence
  - reproducible and simple to parallelize

Modern Path Tracing

Pushbutton paradigm

- deterministic
  - may improve speed of convergence
  - reproducible and simple to parallelize
- unbiased
  - zero difference between expectation and mathematical object
  - not sufficient for convergence

Modern Path Tracing

Pushbutton paradigm

- deterministic
  - may improve speed of convergence
  - reproducible and simple to parallelize
- biased
  - allows for ameliorating the problem of insufficient techniques
  - can tremendously increase efficiency

Modern Path Tracing

Pushbutton paradigm

- deterministic
  - may improve speed of convergence
  - reproducible and simple to parallelize
- biased
  - allows for ameliorating the problem of insufficient techniques
  - can tremendously increase efficiency
- consistent
  - error vanishes with increasing set of samples
  - no persistent artifacts introduced by algorithm

► Quasi-Monte Carlo image synthesis in a nutshell
► The Iray light transport simulation and rendering system

Modern Path Tracing

Pushbutton paradigm

- deterministic
  - may improve speed of convergence
  - reproducible and simple to parallelize
- biased
  - allows for ameliorating the problem of insufficient techniques
  - can tremendously increase efficiency
- consistent
  - error vanishes with increasing set of samples
  - no persistent artifacts introduced by algorithm

▶ Quasi-Monte Carlo image synthesis in a nutshell
▶ The Iray light transport simulation and rendering system

Reconstruction from noisy input: Massively parallel path space filtering (link)

From Machine Learning to Graphics

Machine Learning

Taxonomy

- unsupervised learning from unlabeled data
  - examples: clustering, auto-encoder networks

Machine Learning

Taxonomy

- unsupervised learning from unlabeled data
  - examples: clustering, auto-encoder networks

- semi-supervised learning by rewards
  - example: reinforcement learning

Machine Learning

Taxonomy

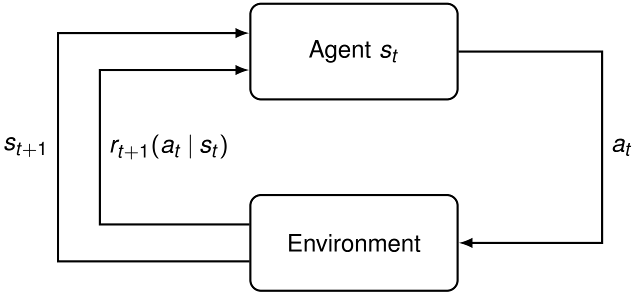
- unsupervised learning from unlabeled data
  - examples: clustering, auto-encoder networks
- semi-supervised learning by rewards
  - example: reinforcement learning
- supervised learning from labeled data
  - examples: support vector machines, decision trees, artificial neural networks

Reinforcement Learning

Goal: maximize reward

- state transition yields reward

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



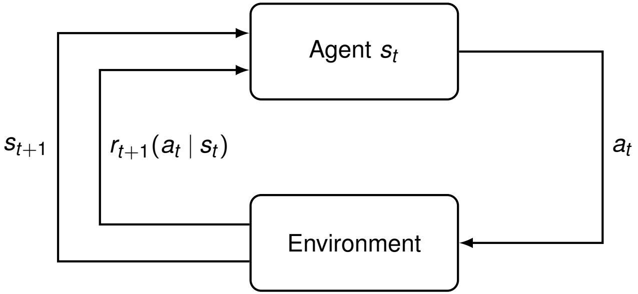
Reinforcement Learning

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 $s_t \in \mathbb{S}$



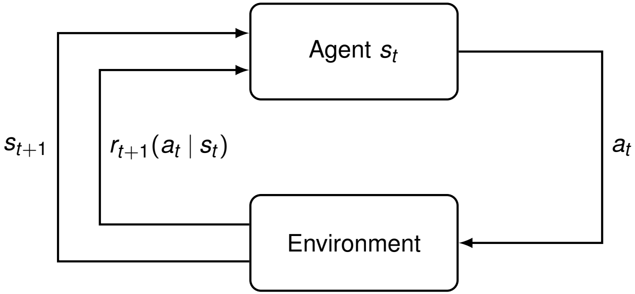
Reinforcement Learning

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 $s_t \in \mathbb{S}$



- maximizing the discounted cumulative reward

$$V(s_t) \equiv \sum_{k=0}^{\infty} \gamma^k \cdot r_{t+1+k}(a_{t+k} \mid s_{t+k}), \text{ where } 0 < \gamma < 1$$

Reinforcement Learning

Q-Learning [Watkins 1989]

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

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

Reinforcement Learning

Q-Learning [Watkins 1989]

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

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

with the following options for the discounted cumulative reward

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

Reinforcement Learning

Q-Learning [Watkins 1989]

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

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

with the following options for the discounted cumulative reward

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

Reinforcement Learning

Q-Learning [Watkins 1989]

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

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

with the following options for the discounted cumulative reward

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

Reinforcement Learning

Maximize reward by learning importance sampling online

- radiance integral equation

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

NVIDIA.

Reinforcement Learning

Maximize reward by learning importance sampling online

- 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 online

- 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 online

- structural equivalence of integral equation and Q-learning

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

NVIDIA.

Reinforcement Learning

Maximize reward by learning importance sampling online

- 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 online

- structural equivalence of integral equation and Q-learning

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

$$Q'(s,a) = (1-\alpha)Q(s,a) + \alpha\left(r(s,a) + \gamma\int_{\mathscr{A}} \pi(s',a') \, Q(s',a') \, da' \right)$$

Reinforcement Learning

Maximize reward by learning importance sampling online

- 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 online

- structural equivalence of integral equation and Q-learning

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

- 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

Reinforcement Learning

Online algorithm for guiding light transport paths

Function *pathTrace(camera, scene)*

```
throughput ← 1
ray ← setupPrimaryRay(camera)
for i ← 0 to ∞ do
    y, n ← intersect(scene, ray)
    if isEnvironment(y) then
        return throughput · getRadianceFromEnvironment(ray, y)
    else if isAreaLight(y)
        return throughput · getRadianceFromAreaLight(ray, y)
    ω, p_ω, f_s ← sampleBsdf(y, n)
    throughput ← throughput · f_s · cos(n, ω) / p_ω
    ray ← y, ω
```

Reinforcement Learning

Online algorithm for guiding light transport paths

Function *pathTrace(camera, scene)*

$throughput \leftarrow 1$

$ray \leftarrow \text{setupPrimaryRay}(camera)$

for $i \leftarrow 0$ **to** ∞ **do**

$y, n \leftarrow \text{intersect}(scene, ray)$

if $i > 0$ **then**

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

if *isEnvironment(y)* **then**

return $throughput \cdot \text{getRadianceFromEnvironment}(ray, y)$

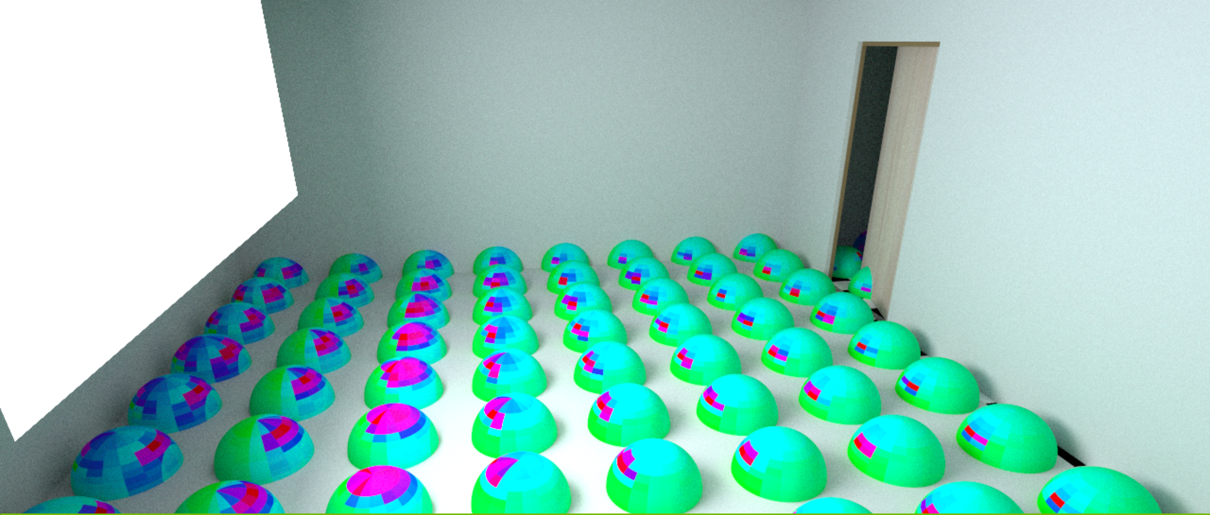
else if *isAreaLight(y)*

return $throughput \cdot \text{getRadianceFromAreaLight}(ray, y)$

$\omega, p_\omega, f_s \leftarrow$ **sampleScatteringDirectionProportionalToQ(y)**

$throughput \leftarrow throughput \cdot f_s \cdot \cos(n,\omega) \,/\, p_\omega$

$ray \leftarrow y,\ \omega$



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

Guiding paths to where the value Q comes from

- shorter expected path length
- dramatically reduced number of paths with zero contribution
- very efficient online learning by learning Q from Q

Reinforcement Learning

Guiding paths to where the value Q comes from

- shorter expected path length
- dramatically reduced number of paths with zero contribution
- very efficient online learning by learning Q from Q
- directions for research
  - representation of value Q: data structures from games
  - importance sampling proportional to the integrand, i.e. the product of policy $\gamma \cdot \pi$ times value Q

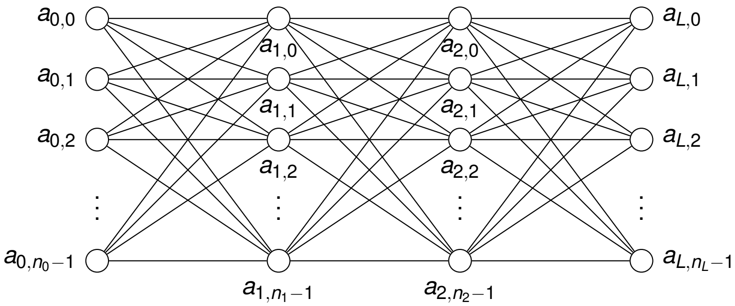
▶ On-line learning of parametric mixture models for light transport simulation
▶ Product importance sampling for light transport path guiding
▶ Fast product importance sampling of environment maps
▶ Learning light transport the reinforced way
▶ Practical path guiding for efficient light-transport simulation

From Graphics back to Machine Learning

Artificial Neural Networks in a Nutshell

Supervised learning of high dimensional function approximation

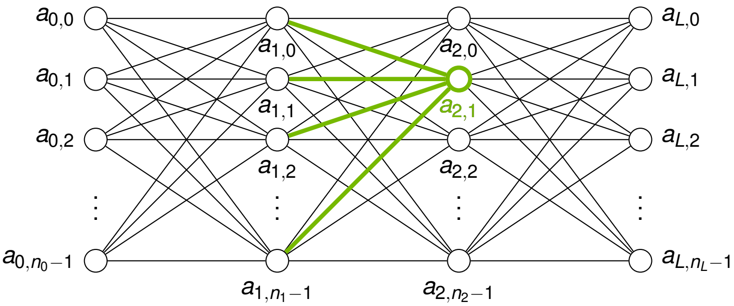
- input layer a_0, $L-1$ hidden layers, and output layer a_L



Artificial Neural Networks in a Nutshell

Supervised learning of high dimensional function approximation

- input layer a_0, $L-1$ hidden layers, and output layer a_L

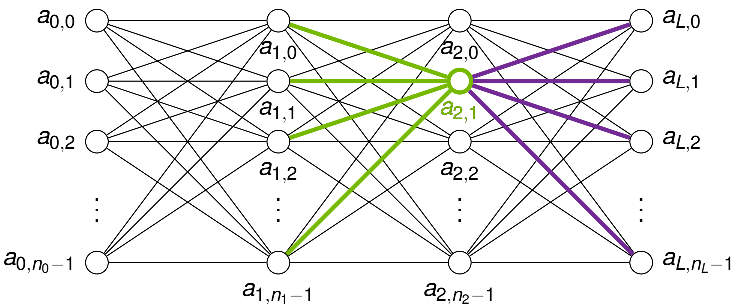


  - n_l rectified linear units (ReLU) $a_{l,i} = \max\{0, \sum w_{l,j,i} a_{l-1,j}\}$ in layer l

Artificial Neural Networks in a Nutshell

Supervised learning of high dimensional function approximation

- input layer a_0, $L-1$ hidden layers, and output layer a_L

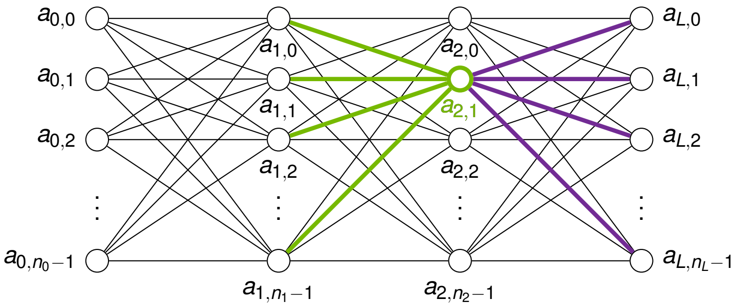


  - n_l rectified linear units (ReLU) $a_{l,i} = \max\{0, \sum w_{l,j,i} a_{l-1,j}\}$ in layer l
  - backpropagating the error $\delta_{l-1,i} = \sum_{a_{l,j}>0} \delta_{l,j} w_{l,j,i}$

Artificial Neural Networks in a Nutshell

Supervised learning of high dimensional function approximation

- input layer a_0, $L-1$ hidden layers, and output layer a_L

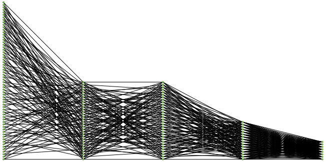


  - n_l rectified linear units (ReLU) $a_{l,i} = \max\{0, \sum w_{l,j,i} a_{l-1,j}\}$ in layer l
  - backpropagating the error $\delta_{l-1,i} = \sum_{a_{l,j}>0} \delta_{l,j} w_{l,j,i}$, update weights $w'_{l,j,i} = w_{l,j,i} - \lambda \delta_{l,j} a_{l-1,i}$ if $a_{l,j} > 0$

Artificial Neural Networks in a Nutshell

Supervised learning of high dimensional function approximation

- example architectures



classifier

▶ Multilayer feedforward networks are universal approximators

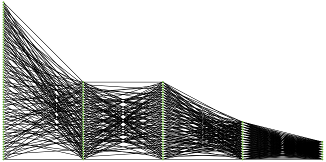
▶ Approximation capabilities of multilayer feedforward networks

▶ Universal approximation bounds for superpositions of a sigmoidal function

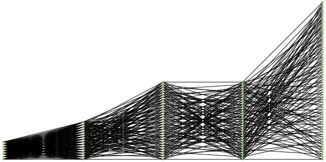
Artificial Neural Networks in a Nutshell

Supervised learning of high dimensional function approximation

- example architectures



classifier



generator

▸ Multilayer feedforward networks are universal approximators

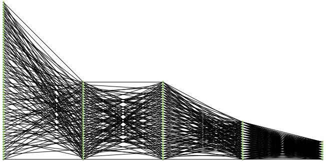
▸ Approximation capabilities of multilayer feedforward networks

▸ Universal approximation bounds for superpositions of a sigmoidal function

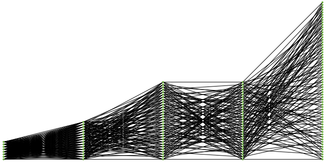
Artificial Neural Networks in a Nutshell

Supervised learning of high dimensional function approximation

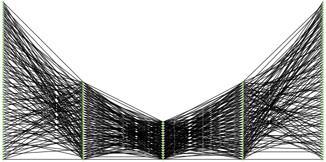
- example architectures



classifier



generator



auto-encoder

- ▶ Multilayer feedforward networks are universal approximators
- ▶ Approximation capabilities of multilayer feedforward networks
- ▶ Universal approximation bounds for superpositions of a sigmoidal function

Efficient Training of Artificial Neural Networks

Using an integral equation for supervised learning

- Q-learning

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

Efficient Training of Artificial Neural Networks

Using an integral equation for supervised learning

- Q-learning

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

for $\alpha = 1$ yields the residual, i.e. loss

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

Efficient Training of Artificial Neural Networks

Using an integral equation for supervised learning

- Q-learning

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

  for $\alpha = 1$ yields the residual, i.e. loss

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

- supervised learning algorithm
  - light transport paths generated by a low discrepancy sequence for online training

Efficient Training of Artificial Neural Networks

Using an integral equation for supervised learning

- Q-learning

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

  for $\alpha = 1$ yields the residual, i.e. loss

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

- supervised learning algorithm
  - light transport paths generated by a low discrepancy sequence for online training
  - learn weights of an artificial neural network for $Q(x,n)$ by back-propagating loss of each path

▶ A machine learning driven sky model

▶ Global illumination with radiance regression Functions

▶ Machine learning and integral equations

▶ Neural importance sampling

Efficient Training of Artificial Neural Networks

Learning from noisy/sampled labeled data

- find set of weights θ of an artificial neural network f to minimize summed loss L
  - using clean targets y_i and data $\hat{x}_i$ distributed according to $\hat{x} \sim p(\hat{x}|y_i)$

$$\text{argmin}_\theta \sum_i L(f_\theta(\hat{x}_i), y_i)$$

Efficient Training of Artificial Neural Networks

Learning from noisy/sampled labeled data

- find set of weights θ of an artificial neural network f to minimize summed loss L
  - using clean targets y_i and data $\hat{x}_i$ distributed according to $\hat{x} \sim p(\hat{x}|y_i)$

$$\operatorname{argmin}_\theta \sum_i L(f_\theta(\hat{x}_i), y_i)$$

  - using targets $\hat{y}_i$ distributed according to $\hat{y} \sim p(\hat{y})$ instead

$$\operatorname{argmin}_\theta \sum_i L(f_\theta(\hat{x}_i), \hat{y}_i)$$

Efficient Training of Artificial Neural Networks

Learning from noisy/sampled labeled data

- find set of weights θ of an artificial neural network f to minimize summed loss L
  - using clean targets y_i and data $\hat{x}_i$ distributed according to $\hat{x} \sim p(\hat{x}|y_i)$

$$\text{argmin}_\theta \sum_i L(f_\theta(\hat{x}_i), y_i)$$

  - using targets $\hat{y}_i$ distributed according to $\hat{y} \sim p(\hat{y})$ instead

$$\text{argmin}_\theta \sum_i L(f_\theta(\hat{x}_i), \hat{y}_i)$$

    - allows for much faster training of artificial neural networks used in simulations

- amounts to learning integration and integro-approximation

▶ Noise2Noise: Learning image restoration without clean data

Example Applications of Artificial Neural Networks in Rendering

Learning from noisy/sampled labeled data

- denoising quasi-Monte Carlo rendered images

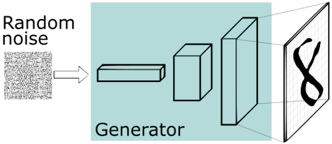
(a) Input (64 spp), 23.93 dB (b) Noisy targets, 32.42 dB (c) Clean targets, 32.95 dB (d) Reference (131k spp)

  - noisy targets computed 2000× faster than clean targets

Example Applications of Artificial Neural Networks in Rendering

Sampling according to a distribution given by observed data

- generative adversarial network (GAN)



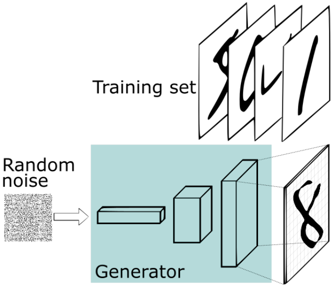
▶ image source

▶ Tutorial on GANs

Example Applications of Artificial Neural Networks in Rendering

Sampling according to a distribution given by observed data

- generative adversarial network (GAN)

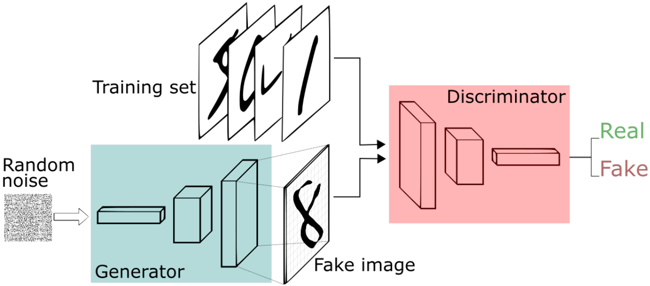


▶ image source
▶ Tutorial on GANs

Example Applications of Artificial Neural Networks in Rendering

Sampling according to a distribution given by observed data

- generative adversarial network (GAN)



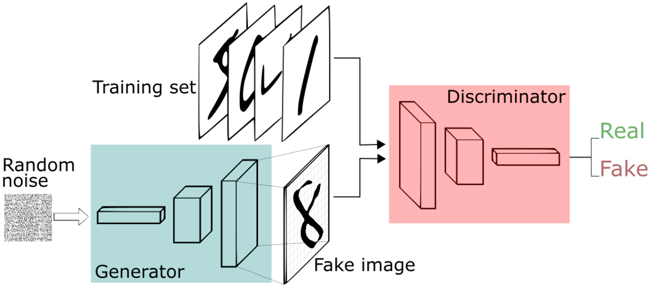
▶ image source
▶ Tutorial on GANs

Example Applications of Artificial Neural Networks in Rendering

Sampling according to a distribution given by observed data

- generative adversarial network (GAN)

  - update generator G using $\nabla_{\theta_g} \sum_{i=1}^{m} \log(1 - D(G(\xi_i)))$



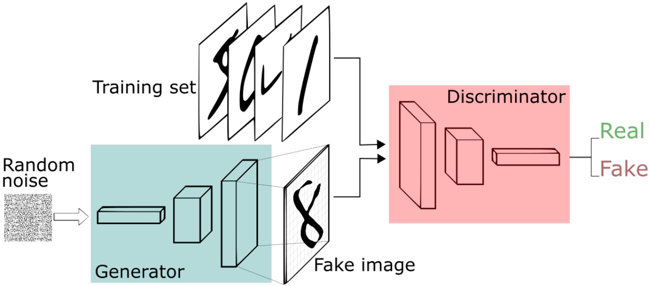
▶ image source

▶ Tutorial on GANs

Example Applications of Artificial Neural Networks in Rendering

Sampling according to a distribution given by observed data

- generative adversarial network (GAN)
  - update discriminator D (k times) using $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x_i) + \log(1 - D(G(\xi_i)))]$
  - update generator G using $\nabla_{\theta_g} \sum_{i=1}^{m} \log(1 - D(G(\xi_i)))$

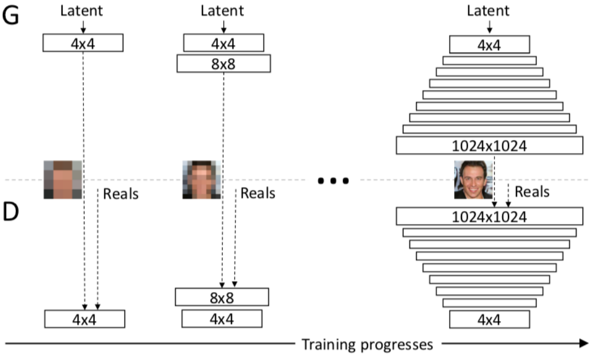


► image source
► Tutorial on GANs

Example Applications of Artificial Neural Networks in Rendering

Sampling according to a distribution given by observed data

- Celebrity GAN

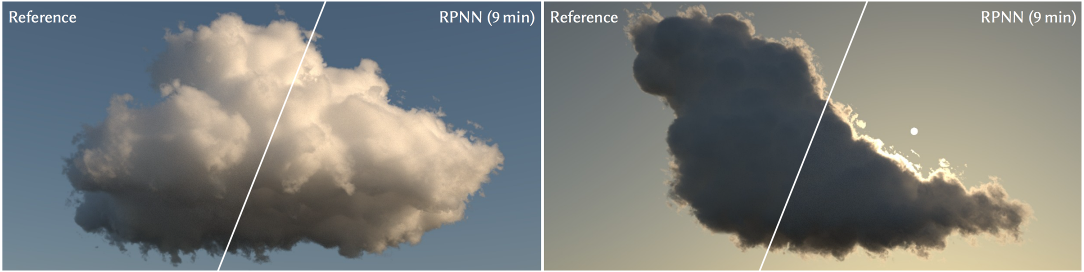


▶ Progressive growing of GANs for improved quality, stability, and variation

Example Applications of Artificial Neural Networks in Rendering

Replacing simulations by learned predictions for more efficiency

- much faster simulation of participating media
  - hierarchical stencil of volume densities as input to the neural network



► Deep scattering: Rendering atmospheric clouds with radiance-predicting neural networks

► Learning particle physics by example: Accelerating science with generative adversarial networks

Neural Networks linear in Time and Space

Neural Networks linear in Time and Space

Complexity

- the brain
  - about 10^{11} nerve cells with to up to 10^4 connections to others

Neural Networks linear in Time and Space

Complexity

- the brain
  - about 10^{11} nerve cells with to up to 10^4 connections to others

- artificial neural networks
  - number of neural units

$$n = \sum_{l=1}^{L} n_l \qquad \text{where } n_l \text{ is the number of neurons in layer } l$$

Neural Networks linear in Time and Space

Complexity

- the brain
  - about 10^{11} nerve cells with to up to 10^4 connections to others

- artificial neural networks
  - number of neural units

$$n = \sum_{l=1}^{L} n_l \qquad \text{where } n_l \text{ is the number of neurons in layer } l$$

  - number of weights

$$n_w = \sum_{l=1}^{L} n_{l-1} \cdot n_l$$

Neural Networks linear in Time and Space

Complexity

- the brain
  - about 10^{11} nerve cells with to up to 10^4 connections to others

- artificial neural networks
  - number of neural units

$$n = \sum_{l=1}^{L} n_l \qquad \text{where } n_l \text{ is the number of neurons in layer } l$$

  - number of weights

$$n_w = \sum_{l=1}^{L} c \cdot n_l$$

  - constrain to constant number c of weights per neuron

Neural Networks linear in Time and Space

Complexity

- the brain
  - about 10^{11} nerve cells with to up to 10^4 connections to others

- artificial neural networks
  - number of neural units

$$n = \sum_{l=1}^{L} n_l \qquad \text{where } n_l \text{ is the number of neurons in layer } l$$

  - number of weights

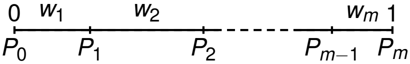
$$n_w = \sum_{l=1}^{L} c \cdot n_l = c \cdot n$$

  - constrain to constant number c of weights per neuron to reach complexity linear in n

Neural Networks linear in Time and Space

Sampling proportional to the weights of the trained neural units

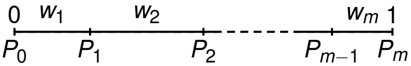
- partition of unit interval by sums $P_k := \sum_{j=1}^{k} |w_j|$ of normalized absolute weights



Neural Networks linear in Time and Space

Sampling proportional to the weights of the trained neural units

- partition of unit interval by sums $P_k := \sum_{j=1}^{k} |w_j|$ of normalized absolute weights



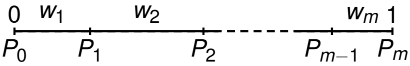
  - using a uniform random variable $\xi \in [0,1)$ to

    select input $i \Leftrightarrow P_{i-1} \leq \xi < P_i$ satisfying $\text{Prob}(\{P_{i-1} \leq \xi < P_i\}) = |w_i|$

Neural Networks linear in Time and Space

Sampling proportional to the weights of the trained neural units

- partition of unit interval by sums $P_k := \sum_{j=1}^{k} |w_j|$ of normalized absolute weights



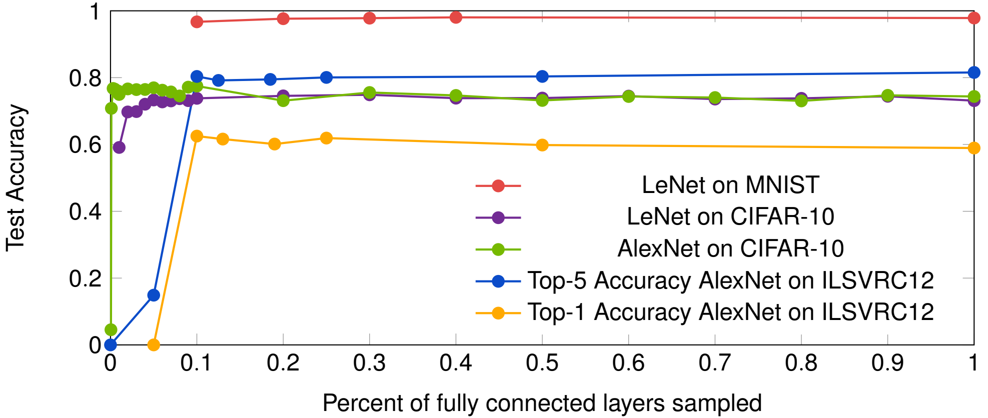
  - using a uniform random variable $\xi \in [0,1)$ to

    select input $i \Leftrightarrow P_{i-1} \leq \xi < P_i$ satisfying $\text{Prob}(\{P_{i-1} \leq \xi < P_i\}) = |w_i|$

- in fact derivation of quantization to ternary weights in $\{-1, 0, +1\}$
  - integer weights result from neurons referenced more than once
  - relation to drop connect and drop out

Neural Networks linear in Time and Space

Sampling proportional to the weights of the trained neural units



Neural Networks linear in Time and Space

Sampling paths through networks

- complexity bounded by number of paths times depth L of network

Neural Networks linear in Time and Space

Sampling paths through networks

- complexity bounded by number of paths times depth L of network

- application after training
  - backwards random walks using sampling proportional to the weights of a neuron
  - compression and quantization by importance sampling

Neural Networks linear in Time and Space

Sampling paths through networks

- complexity bounded by number of paths times depth L of network

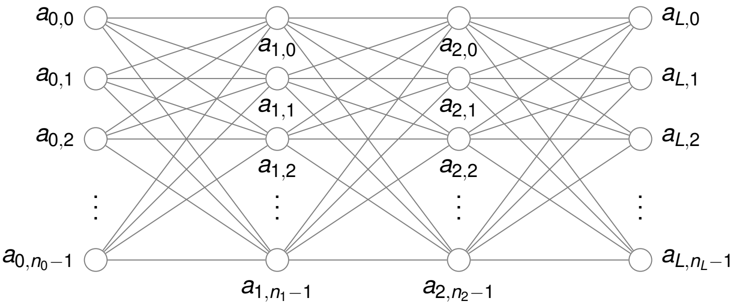
- application after training
  - backwards random walks using sampling proportional to the weights of a neuron
  - compression and quantization by importance sampling

- application before training
  - uniform (bidirectional) random walks to connect inputs and outputs
  - sparse from scratch

Neural Networks linear in Time and Space

Sampling paths through networks

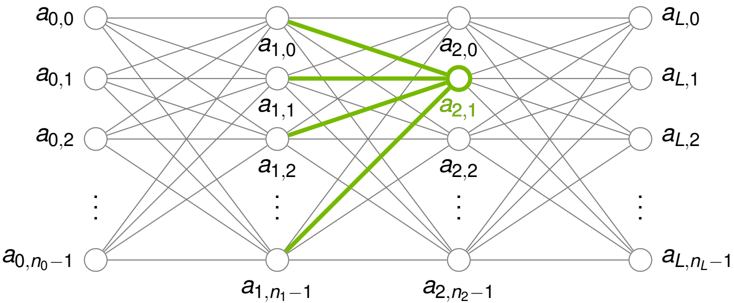
- sparse from scratch



Neural Networks linear in Time and Space

Sampling paths through networks

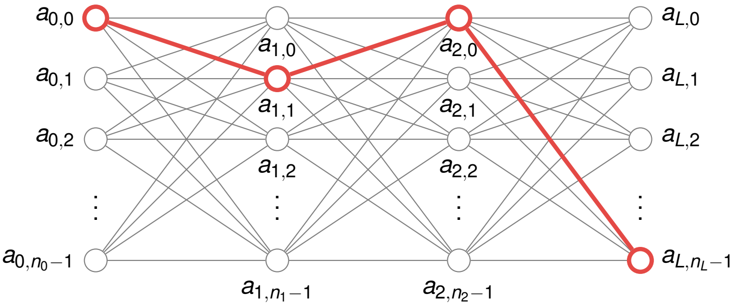
- sparse from scratch



Neural Networks linear in Time and Space

Sampling paths through networks

- sparse from scratch



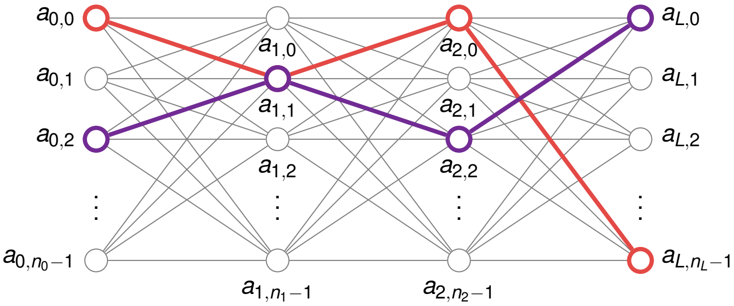
  - guaranteed connectivity

▶ Monte Carlo methods and neural networks

Neural Networks linear in Time and Space

Sampling paths through networks

- sparse from scratch



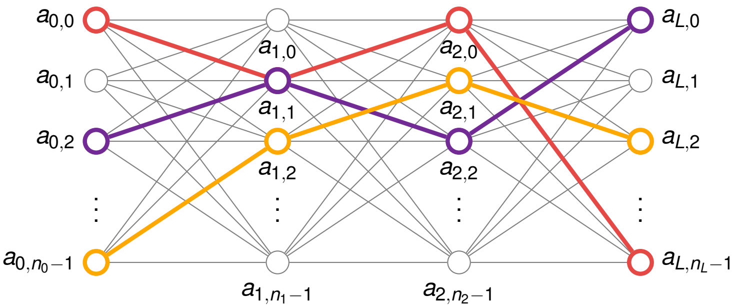
  - guaranteed connectivity

▶ Monte Carlo methods and neural networks

Neural Networks linear in Time and Space

Sampling paths through networks

- sparse from scratch



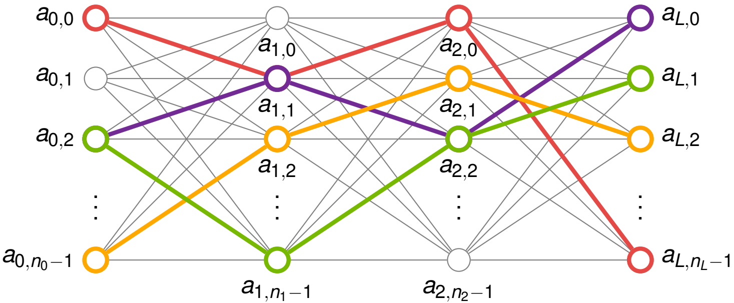
  - guaranteed connectivity

▶ Monte Carlo methods and neural networks

Neural Networks linear in Time and Space

Sampling paths through networks

- sparse from scratch



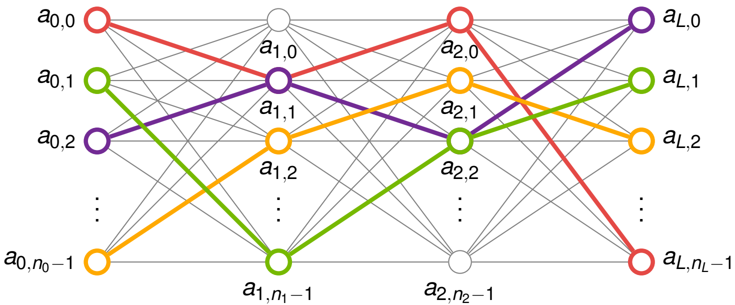
  – guaranteed connectivity

▸ Monte Carlo methods and neural networks

Neural Networks linear in Time and Space

Sampling paths through networks

- sparse from scratch

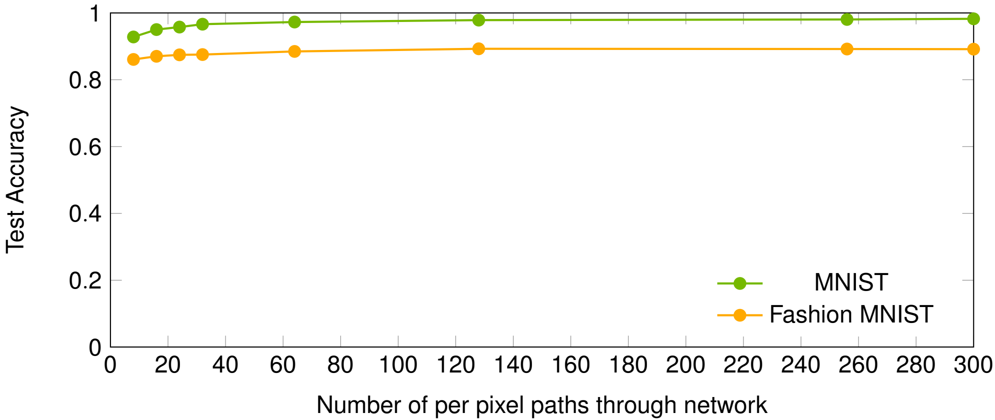


  - guaranteed connectivity and coverage

▶ Monte Carlo methods and neural networks

Neural Networks linear in Time and Space

Test accuracy for 4 layer feedforward network (784/300/300/10) trained sparse from scratch



From Machine Learning to Graphics and back

Summary

- light transport and reinforcement learning described by same integral equation
  - learn where radiance comes from
- neural networks results of linear complexity by path tracing
  - ternarization and quantization of trained artificial neural networks
  - sparse from scratch training