Metropolis Light Transport

Bas du Pré
Marcel Penningnieuwland

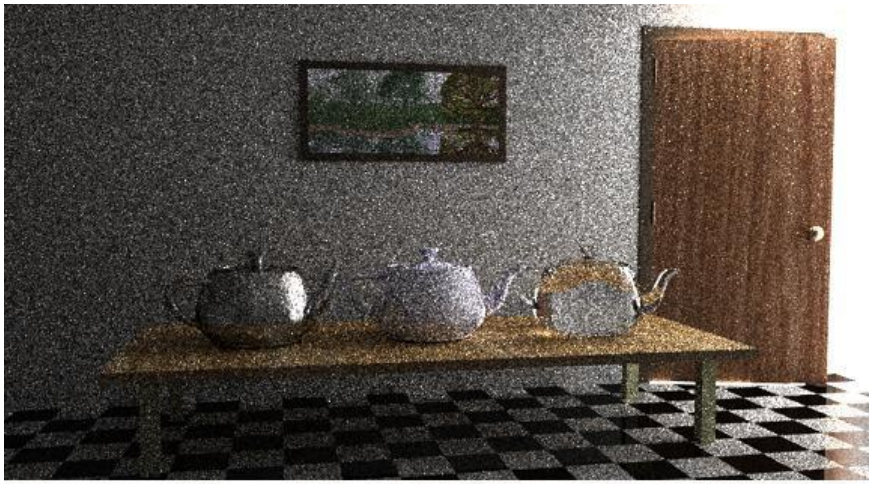
Contents

Introduction: What is MLT?

- Path tracing algorithm
- Randomly chosen paths. But:
  - We mostly want "good" paths
  - Re-use good subpaths

Introduction: Why MLT?

- MLT performs well in difficult scenes
  - Indirect lighting
  - Caustics
- How come?
  - Few bad paths
  - Few rays traced per new path



(a) Bidirectional path tracing with 40 samples per pixel.

(b) Metropolis light transport with an average of 250 mutations per pixel [the same computation time as (a)].

MLT algorithm overview

- Generate initial path
- For i = 1 to sampleCount:
  - Mutate the current path
  - Accept or reject the new path
  - Sample the path
- Return image

MLT algorithm overview

- Goal: to find good paths to sample
  - Not to find a single "best" path!
- Solution:
  - Mutation strategy
  - Acceptance probability

- Generate initial path
- For i = 1 to sampleCount:
  - Mutate the current path
  - Accept or reject the new path
  - Sample the path
- Return image

The Metropolis Sampling Algorithm

Goal

to generate a random walk X_0, X_1, ..., X_i such that X_i is eventually distributed proportionally to f.

$$X_i \in \Omega$$

$$f = \Omega \rightarrow \mathbb{R}^+$$

The Metropolis Sampling Algorithm

- **Given:** probability density K(x|y)

- **Initialization**
  choose arbitrary point X_0
- **For each iteration *t*:**
  - Generate random candidate X' from $K(X'|X_t)$
  - $a = f(X')/f(X_t)$
  - if a > random(0..1): $X_{t+1}=X'$
    else: $X_{t+1}=X_t$

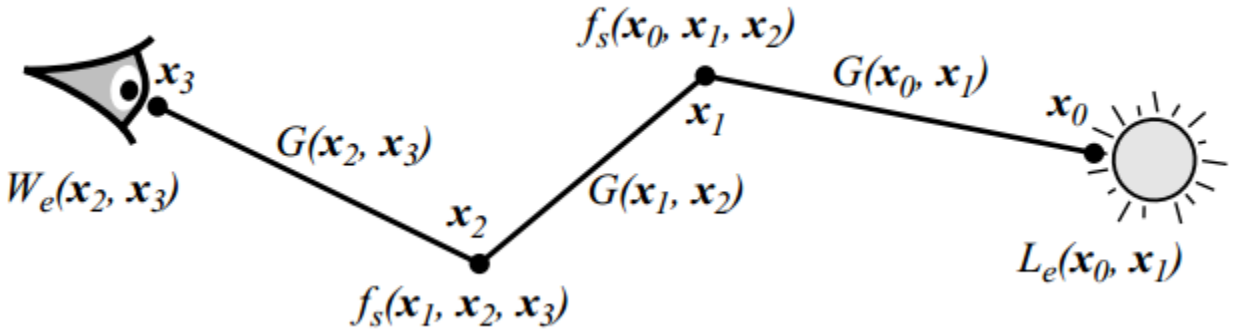
The Metropolis Sampling Algorithm

$$p_i(\bar{x}) = \int_{\Omega} K(\bar{x}\,|\,\bar{y})\; p_{i-1}(\bar{y})\; d\mu(\bar{y})$$

Eventually the algorithm converges to a stable state, i.e.: $p(\bar{x}) = \sum_{\bar{y}} K(\bar{x}\,|\,\bar{y})\; p(\bar{y})$

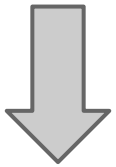
The light transport equation

$$L(\mathbf{x}' \to \mathbf{x}'') = L_e(\mathbf{x}' \to \mathbf{x}'') + \int_{\mathcal{M}} L(\mathbf{x} \to \mathbf{x}')\, f_s(\mathbf{x} \to \mathbf{x}' \to \mathbf{x}'')\, G(\mathbf{x} \leftrightarrow \mathbf{x}')\, dA(\mathbf{x}).$$



The path integral formulation

$$
\begin{aligned}
m_j = & \int_{\mathcal{M}^2} L_{\mathrm{e}}(\mathbf{x}_0 \to \mathbf{x}_1)\, G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)\, W_{\mathrm{e}}^{(j)}(\mathbf{x}_0 \to \mathbf{x}_1)\, dA(\mathbf{x}_0)\, dA(\mathbf{x}_1) \\
& + \int_{\mathcal{M}^3} L_{\mathrm{e}}(\mathbf{x}_0 \to \mathbf{x}_1)\, G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)\, f_{\mathrm{s}}(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1 \leftrightarrow \mathbf{x}_2) \\
& \qquad G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2)\, W_{\mathrm{e}}^{(j)}(\mathbf{x}_1 \to \mathbf{x}_2)\, dA(\mathbf{x}_0)\, dA(\mathbf{x}_1)\, dA(\mathbf{x}_2) \\
& + \cdots .
\end{aligned}
$$



$$
m_j = \int_{\Omega} f_j(\bar{x})\, d\mu(\bar{x})
$$

The path integral formulation

Why do we want this form?

$$m_j = \int_\Omega f_j(\bar{x})\, d\mu(\bar{x})$$

Recall from Metropolis Sampling:

$$p_i(\bar{x}) = \int_\Omega K(\bar{x}\,|\,\bar{y})\, p_{i-1}(\bar{y})\, d\mu(\bar{y})$$

The path integral formulation

$$m_j = \int_\Omega f_j(\bar{x})\, d\mu(\bar{x})$$

f_j is of the form

$$f_j(\bar{x}) = w_j(\bar{x})\, f(\bar{x})$$

where $f(x)$ does not depend on j

Calculating the image

To calculate the image:

$$m_j = E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{w_j(\bar{X}_i)f(\bar{X}_i)}{p(\bar{X}_i)}\right]$$

Calculating the image

To calculate the image:

$$m_j = E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{w_j(\bar{X}_i)f(\bar{X}_i)}{p(\bar{X}_i)}\right]$$

- m_j = the value of pixel j

Calculating the image

To calculate the image:

$$m_j = E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{w_j(\bar{X}_i)f(\bar{X}_i)}{p(\bar{X}_i)}\right]$$

- mj = the value of pixel j
- N = the total number of samples

(we're taking an average of samples)

Calculating the image

To calculate the image:

$$m_j = E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{w_j(\bar{X}_i)f(\bar{X}_i)}{p(\bar{X}_i)}\right]$$

- mj = the value of pixel j
- N = the total number of samples
- Xi = path #i
- wj(Xi) f(Xi) = The amount of light through Xi that hits pixel j

Calculating the image

To calculate the image:

$$m_j = E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{w_j(\bar{X}_i)f(\bar{X}_i)}{p(\bar{X}_i)}\right]$$

- mj = the value of pixel j
- N = the total number of samples
- Xi = path #i
- wj(Xi) f(Xi) = The amount of light through Xi that hits the pixel
- p = The sampling distribution
  (the chance we sample Xi on a single iteration)

Generating the initial path

- Idea: Avoid bias by adding weights to paths
- Take a large random set of paths
  - using BDPT
  - we call paths X_0^1, ..., X_0^n
  - assign weights W_0^1, ..., W_0^n
- Then sample those paths
  - relative to their weight
- Run metropolis for each sample

Mutation strategy

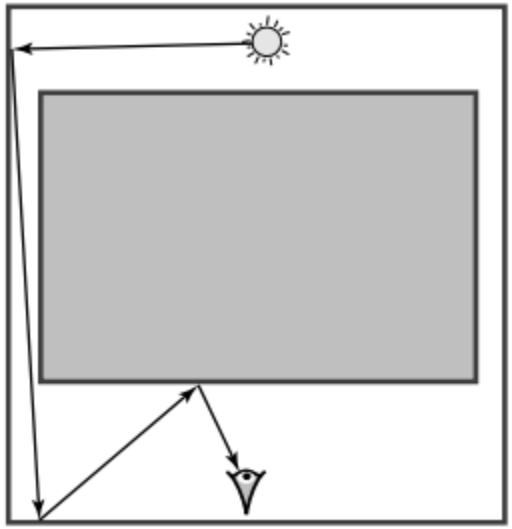
- Problem: Finding good alternative paths
- We must consider:
  - significant changes to the path
    - avoids noise
    - alter path length
    - different surfaces

Mutation strategy

- Problem: Finding good alternative paths
- We must consider:
  - significant changes to the path
  - paths through different pixels

Mutation strategy

- Problem: Finding good alternative paths
- We must consider:
  - significant changes to the path
  - paths through different pixels
  - ergodicity
    - entire path space is reachable



Mutation strategy

- Problem: Finding good alternative paths
- We must consider:
  - significant changes to the path
  - ergodicity
  - stratification
    - many samples for one pixel
    - few samples for another

Mutation strategy

- Problem: Finding good alternative paths
- We must consider:
  - significant changes to the path
  - ergodicity
  - stratification
  - speed
    - trace few rays per path

Mutation strategy

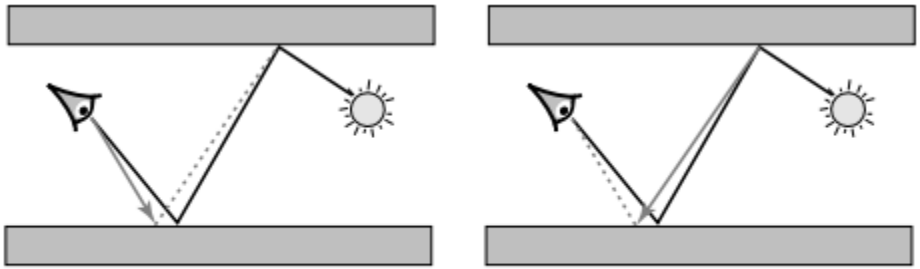
- Problem: Finding good alternative paths
- We must consider:
  - significant changes to the path
  - ergodicity
  - stratification
  - speed
  - high acceptance probability
    - otherwise: sample the same path often

Mutation strategy

- Bidirectional mutations
  - procedure:
    - remove a subpath
    - create a new subpath
  - benefits:
    - significant changes
      - new subpath has random length
      - bounce off other surfaces
    - ergodicity
      - chance to remove entire path
    - speed
      - only need to trace new subpath

Mutation strategy

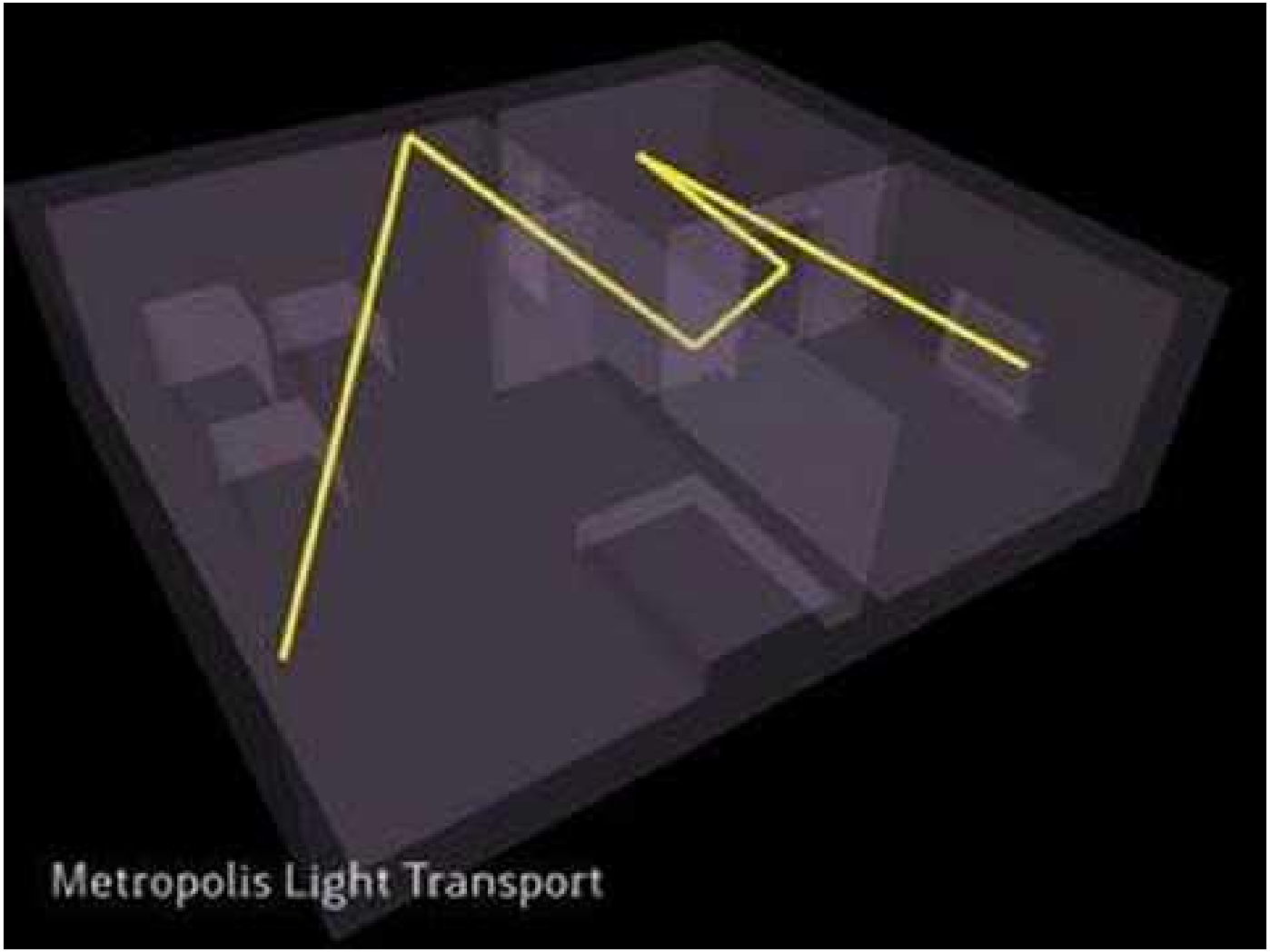
- Bidirectional mutations
- Perturbations
  - Small changes to a (sub)path
  - Lens perturbation
    - Change the lens edge by some angle
  - Caustic perturbation
    - Change the angle of the subpath's second edge



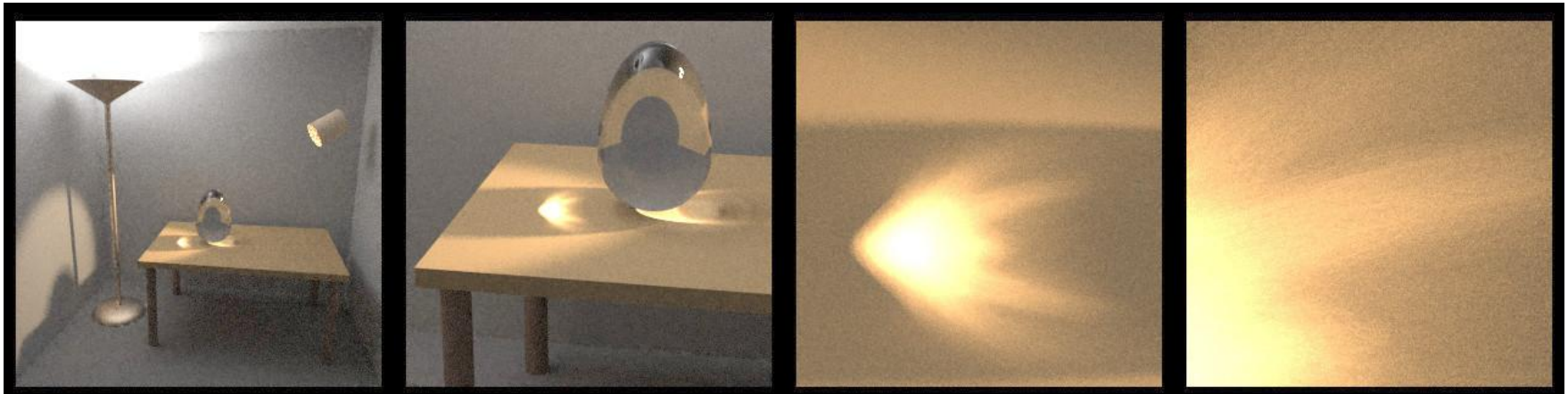
Lens perturbation

Caustic perturbation

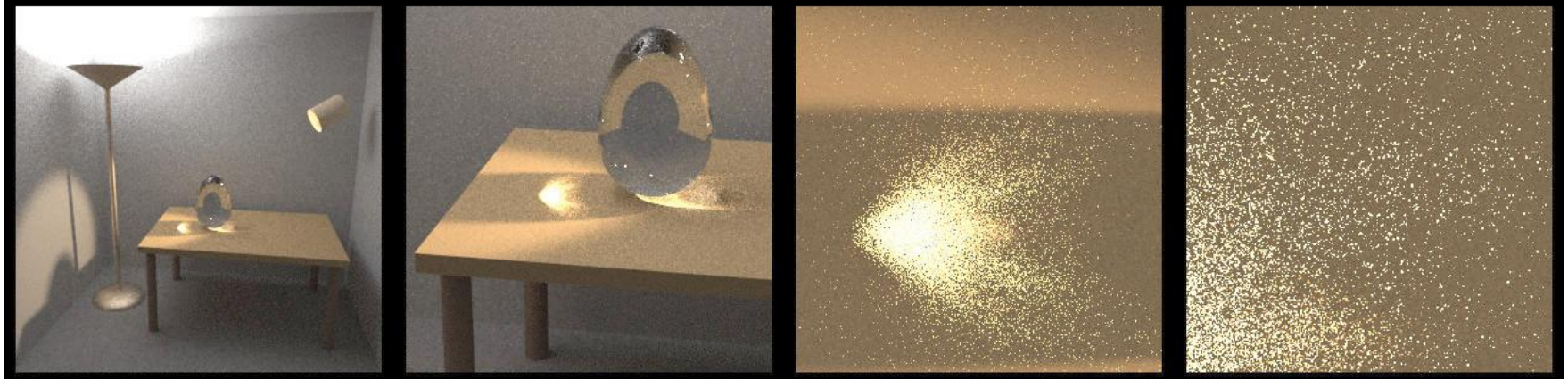
Results



Results



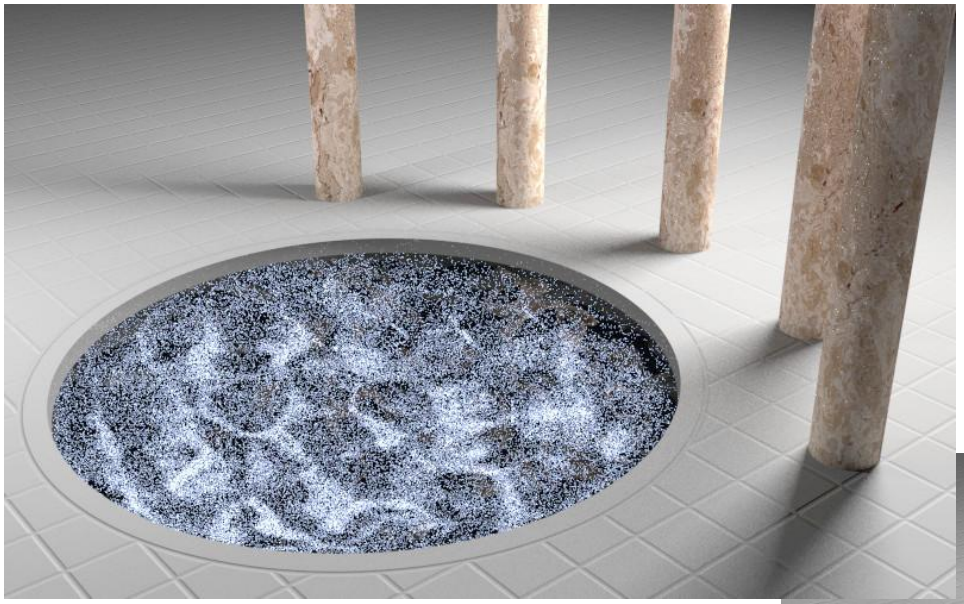
Metropolis light transport

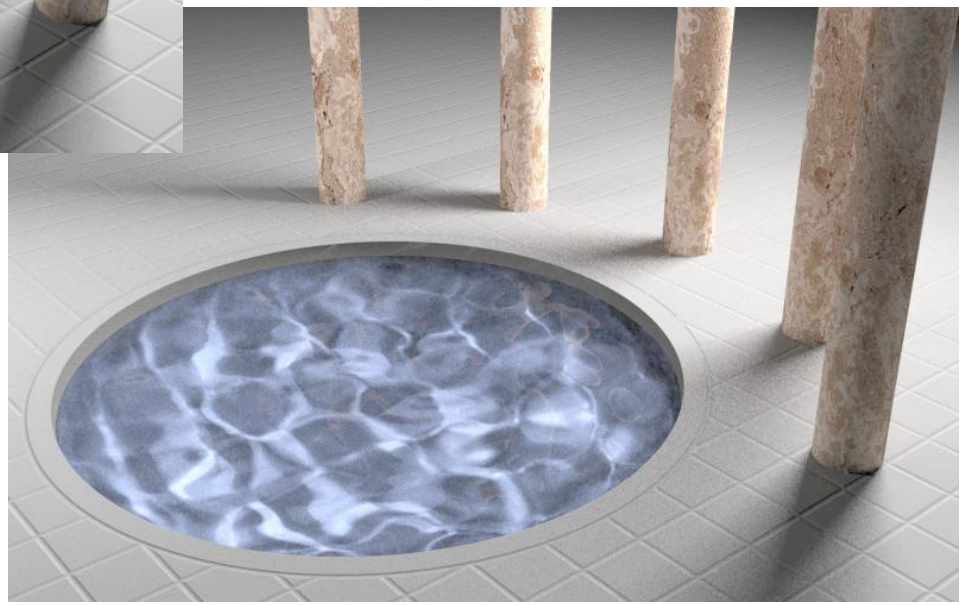


Bidirectional path tracing

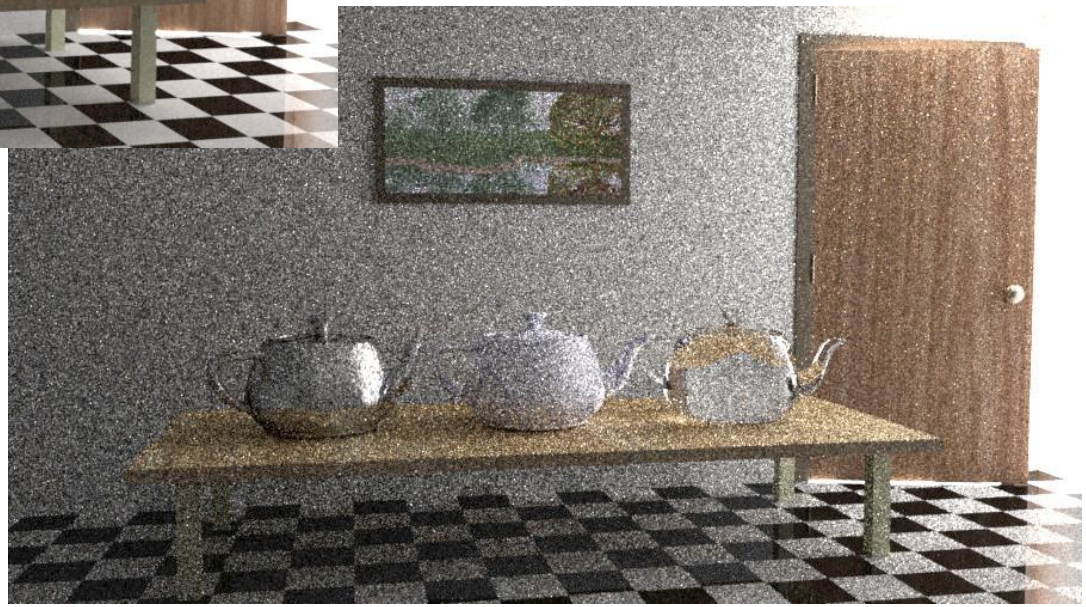
Results

Results





Results



Conclusion

Metropolis Light Transport

- path tracing
  - but optimized
- uses metropolis sampling
- good results
  - depending on the mutation strategy
- unbiased