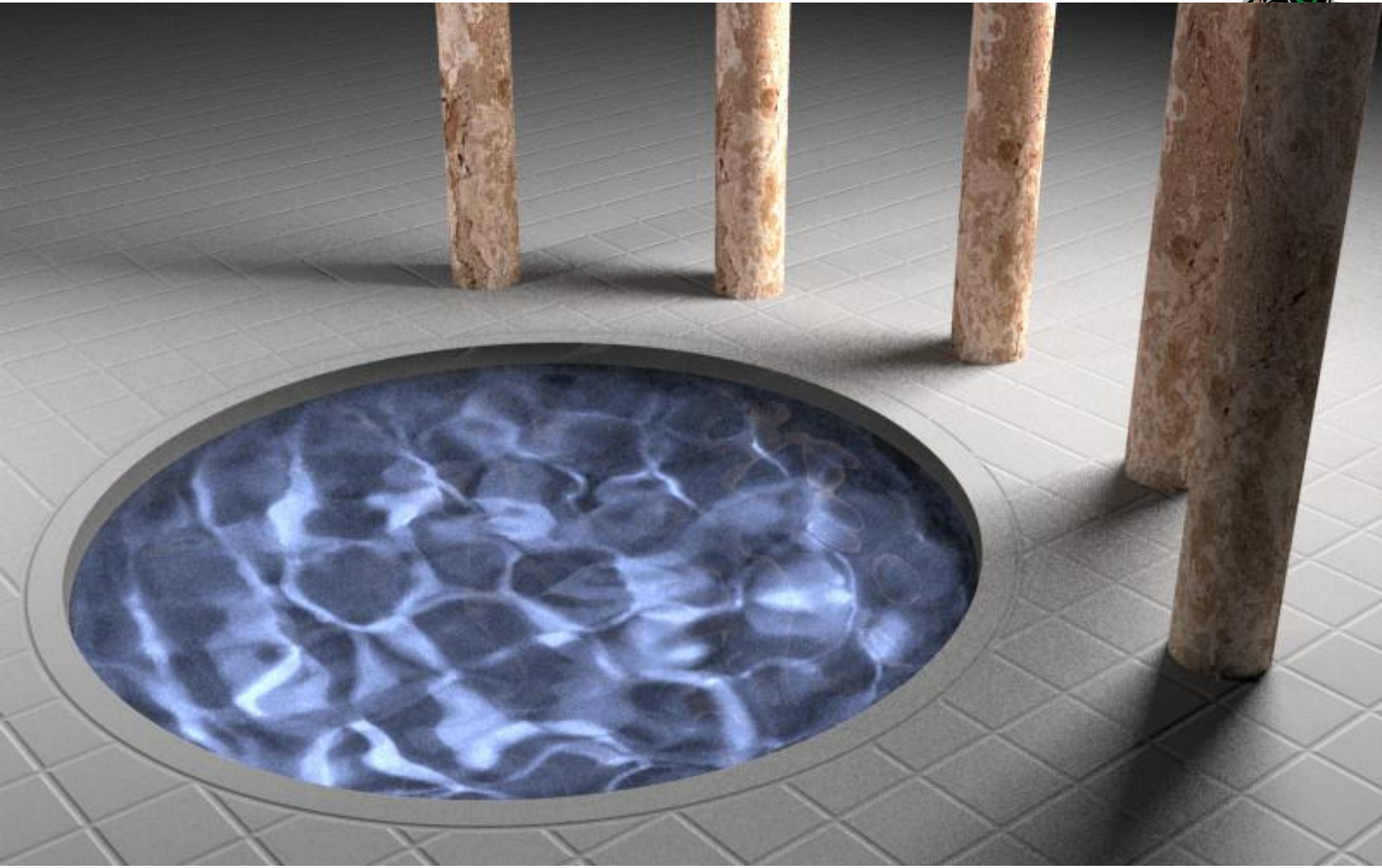
Path Tracing: Just a Quick View…

CS434

Daniel G. Aliaga
Department of Computer Science
Purdue University

Henrik Wann Jensen



Path Tracing

- Trace light transport paths to determine pixel intensities
- A path of length *k* is a sequence of vertices, $<x_0,\ldots,x_{k-1}>$ where every x_i and x_{i+1} is mutually visible, and x_0 is on a light
- Many such paths!
- We are most interested in “important” paths!

Important Paths

- Consider only paths that go from a light source to the eye
  - Other useful paths are sub-paths of these
  - Paths that miss the image plane contribute nothing, so are not important
- Paths that carry more energy are more important
- Why is that?

Sampling Important Paths

- Importance sampling
  - Sample paths of various lengths
  - Weight their contribution to pixel intensity by their importance
- How are these paths found?

Naïve Path Tracing (version 1)

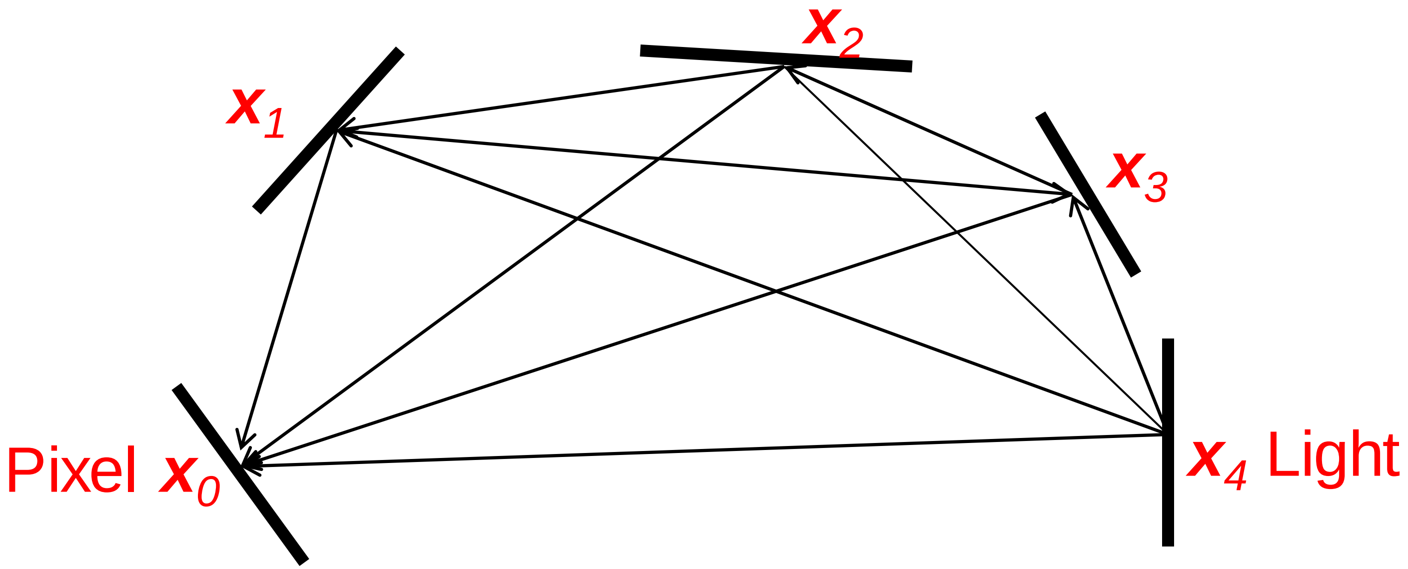
- Start at light
- Build a path by randomly choosing a direction at each bounce, and adding point hit by ray in that direction
- Join last point to eye
- What is the basic problem? What paths does it get?

Naïve Path Tracing (version 2)

- Start at eye
- Build a path by randomly choosing a direction at each bounce, and adding point hit by ray in that direction
- (optional) Join last point to light
- What is the basic problem? What paths does it get?

Pure Bi-Directional: Approach

- Build a path by working from the eye and the light and join in the middle
- Don't just look at overall path, also weigh contributions from all sub-paths:



Pure Bi-Directional: Analysis

- Advantages:
  - Each ray cast contributes to many paths
  - Building from both ends can catch difficult cases
    - All specular paths
    - Caustics
  - Extends to participating media (anisotropic, heterogeneous)
- Disadvantages:
  - Still using lots of effort to catch slow varying diffuse components
  - May not sample difficult to find paths

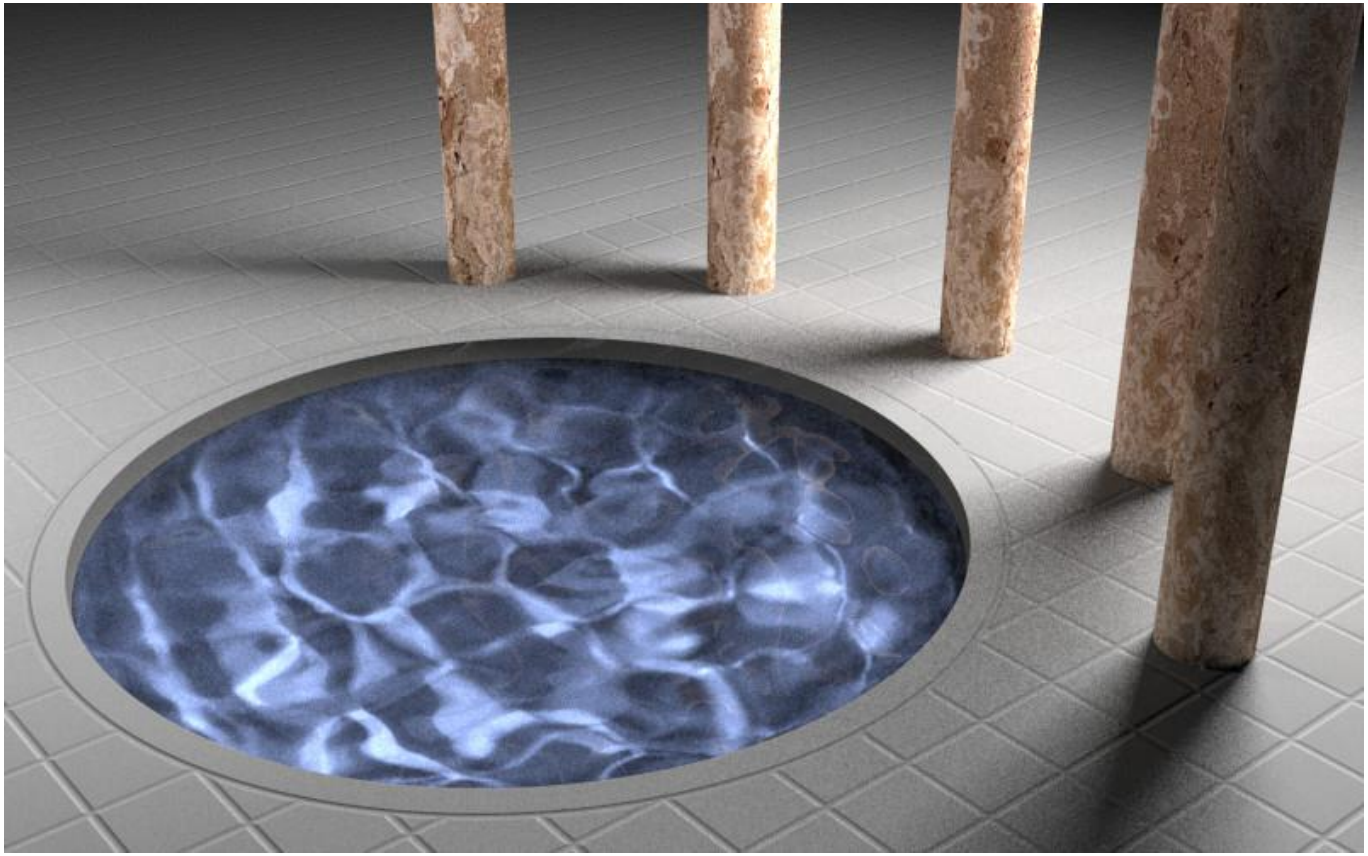
Metropolis Light Transport: Approach

- Other algorithms generate independent samples
  - Easy to control bias
- Metropolis algorithms generate a sequence of paths where each path can depend on the previous one
- For each sample:
  - Propose a new candidate depending on the previous sample
  - Choose to accept or reject according to a computed probability (if reject, re-use the old sample)
- Can prove the estimates for pixel intensities are correct

Metropolis Proposal Strategies

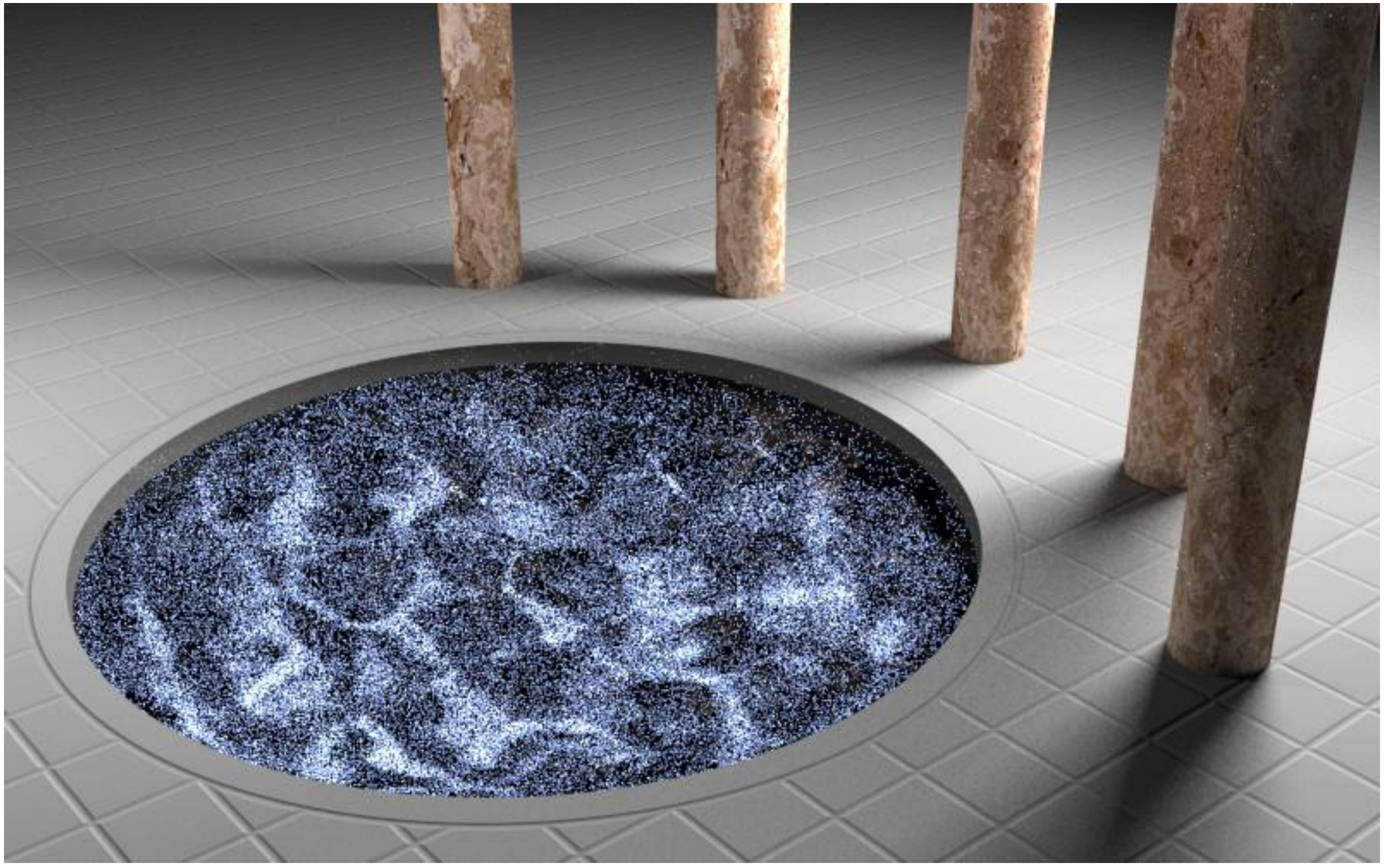
- Task: Given the previous sample, generate a new one
  - Should be very different, but should also be good
- Methods:
  - Randomly chop out some part of the path and replace it with a new piece
  - Randomly perturb a vertex on the path
  - Less randomly change the pixel that is affected
  - Other choices possible

Light Through Ripples



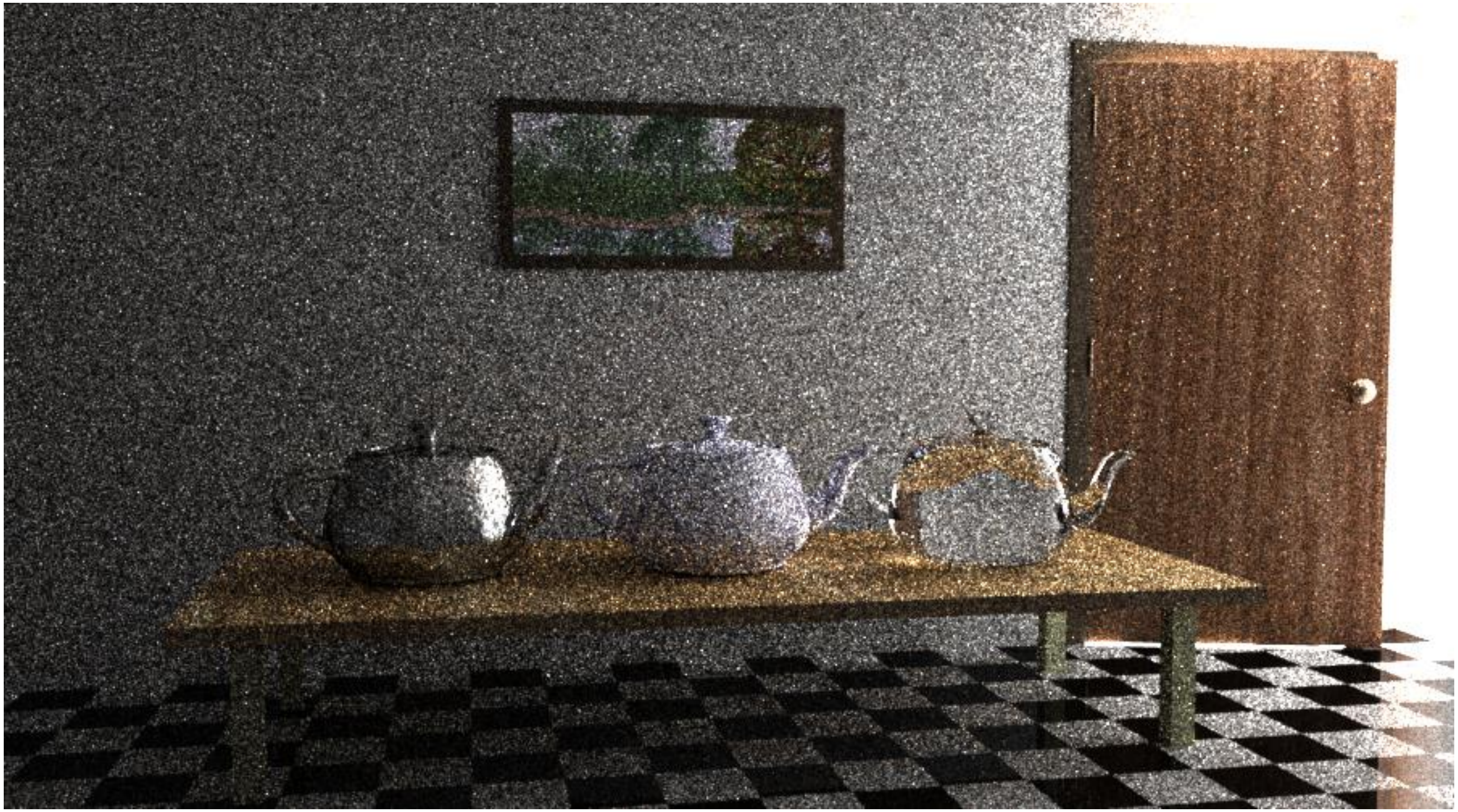
http://graphics.stanford.edu/papers/metro/

Light Through Ripples (Path tracing)



http://graphics.stanford.edu/papers/metro/

Bidirectional Path Tracing



Metropolis Light Transport

Metropolis: Analysis

- Easy to implement basic algorithm
  - Some of the details for good results are difficult
  - Easy to parallelize
- Can do difficult scenarios:
  - Light through a crack, almost impossible any other way
  - Caustics from light reflecting off the bottom of a wavy pool
- But, still computes diffuse illumination on a per point basis