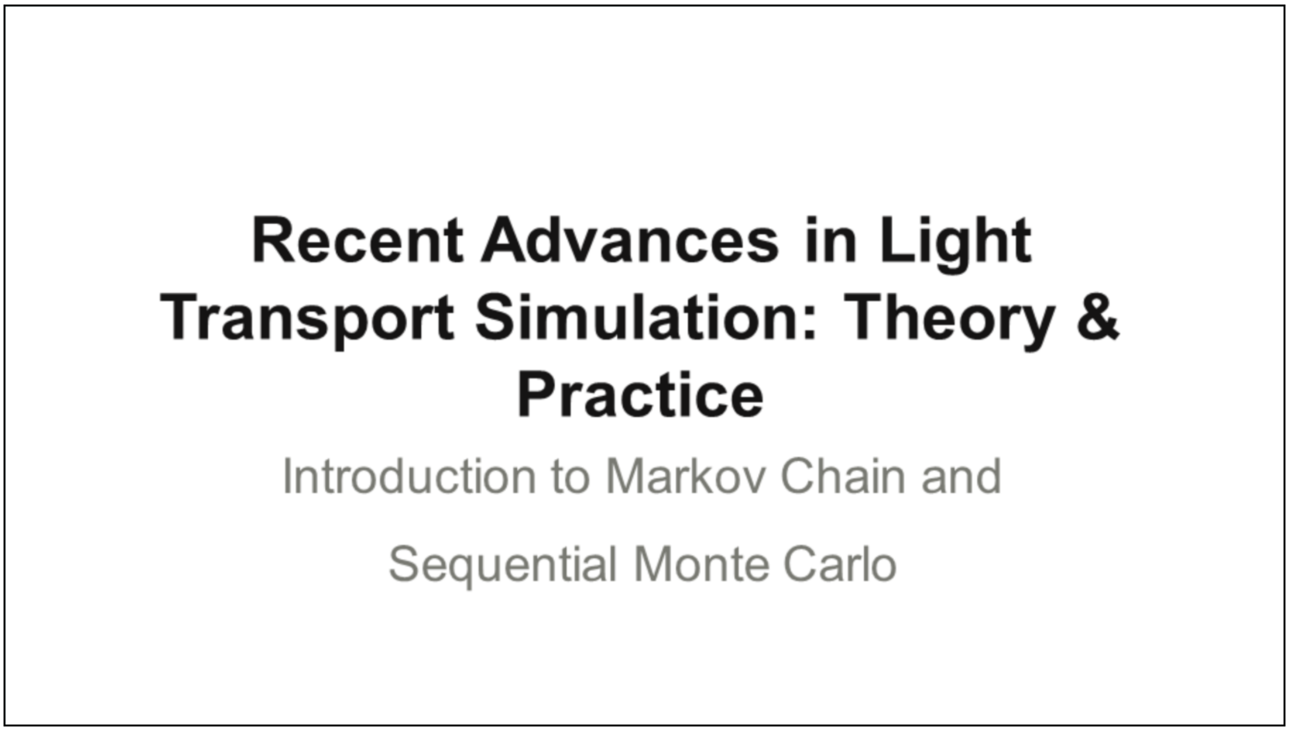
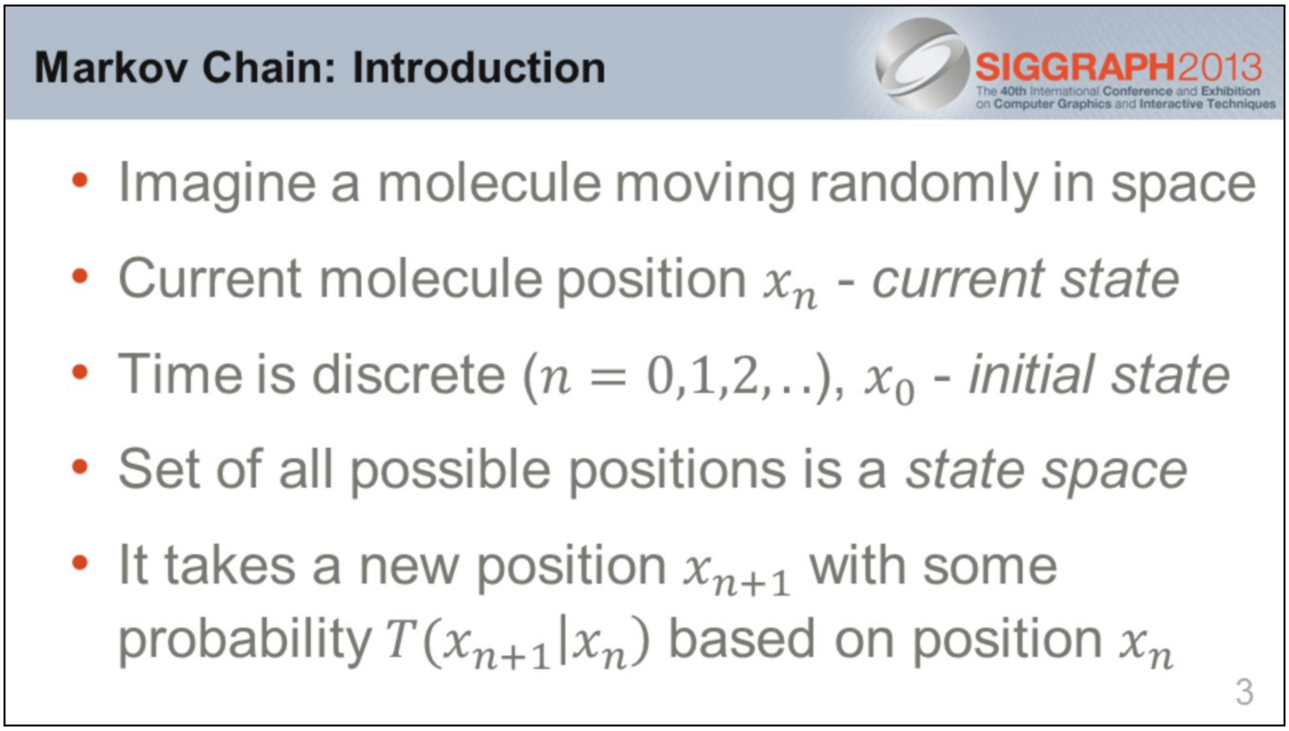


Recent Advances in Light Transport Simulation: Theory & Practice

Introduction to Markov Chain and

Sequential Monte Carlo

Markov Chains



Markov Chain: Introduction

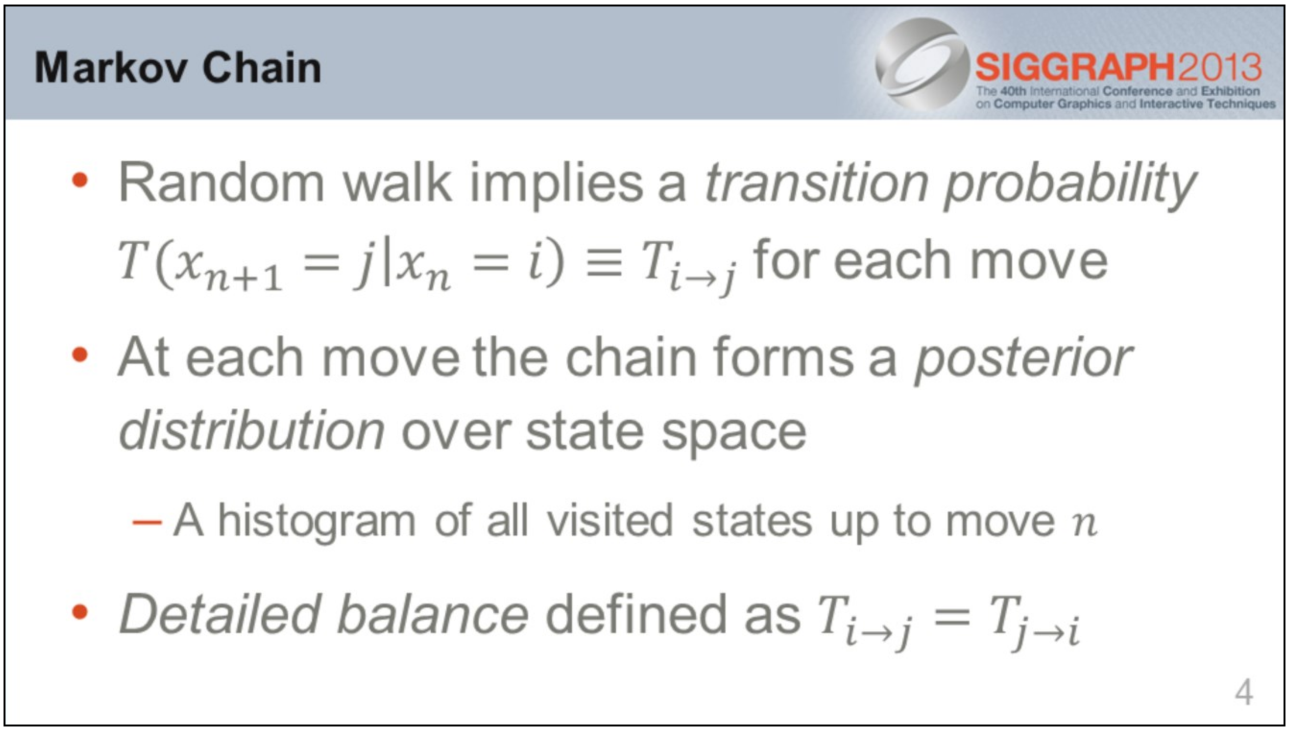
SIGGRAPH2013 The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- Imagine a molecule moving randomly in space
- Current molecule position x_n - *current state*
- Time is discrete ($n = 0,1,2,..$), x_0 - *initial state*
- Set of all possible positions is a *state space*
- It takes a new position x_{n+1} with some probability $T(x_{n+1}|x_n)$ based on position x_n

3

One intuitive way of thinking about a Markov chain would be to imagine it as a molecule performing random movements in a gas or fluid. We imply an important assumption that the molecule has no memory, i.e. its next move depends solely on its current position and does not depend on any of its previous positions. In this case, such memory-less process is called a *Markov chain*. Current position x_n of the molecule at time *n* is called a *current state* of the Markov chain. If time *n* is continuous, the process is called time-continuous Markov Process. Hereafter we will consider only Markov chains with discrete time. The position x_0 of the molecule at time *n*=0 is called an *initial state* of the Markov chain. The space of all possible positions of the molecule is called a *state space*, on which a Markov chain is defined. The molecule moves probabilistically in this medium, such that for each current position of the molecule (current state) x_n and any desired position x_{n+1} (proposal state) for the next move there is defined a conditional transition probability $T(x_{n+1}|x_n)$ of performing such move.

Now we will move away of the molecule example. That means that the state x of the Markov chain can belong to any abstract state space Ω. For example, as we will see later a state x can correspond to a complete light path from a light source to a camera sensor.



Markov Chain

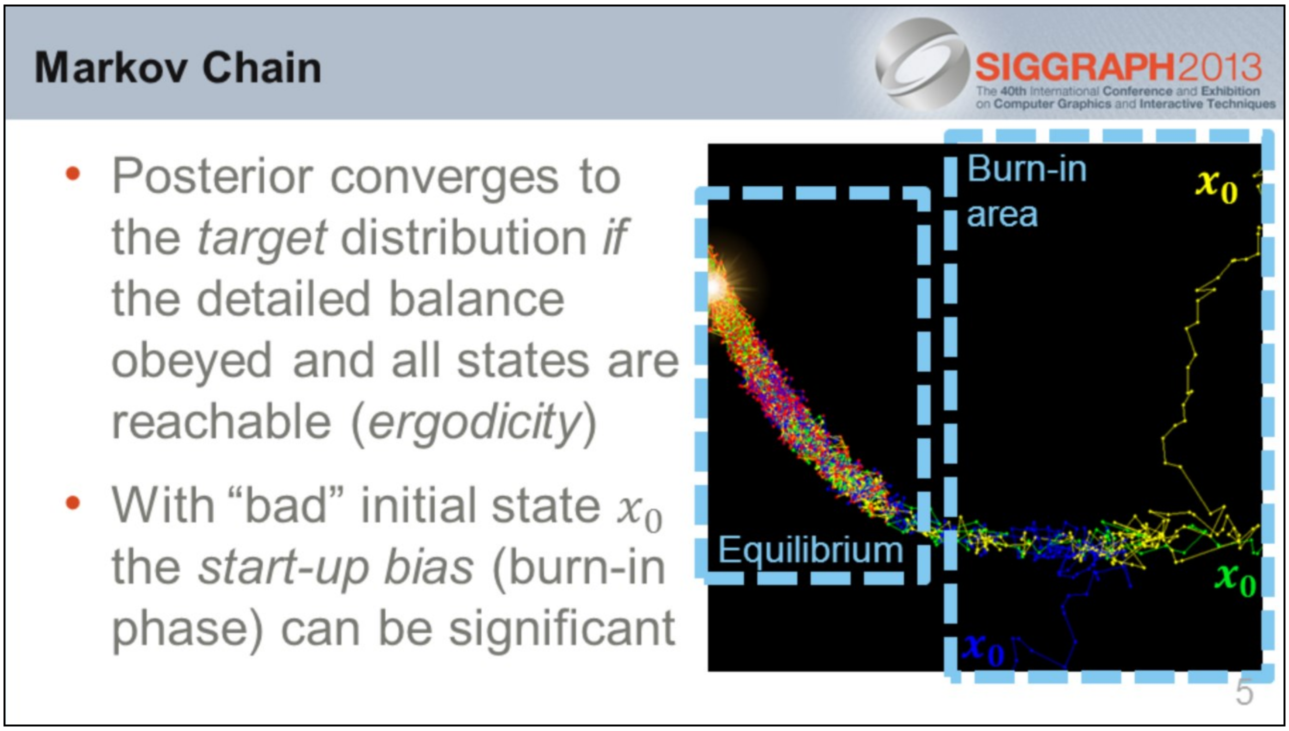
SIGGRAPH2013 The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- Random walk implies a *transition probability* $T(x_{n+1} = j | x_n = i) \equiv T_{i \to j}$ for each move
- At each move the chain forms a *posterior distribution* over state space
  - A histogram of all visited states up to move n
- *Detailed balance* defined as $T_{i \to j} = T_{j \to i}$

4

The random walk implicitly defines the conditional transition probability T for every move. For every current state i and a proposal state j at move n a transition probability $T(x_{n+1} = j | x_n = i)$ is defined. Hereafter we will denote this probability simply as $T_{i \to j}$ for brevity. After n moves we can build a histogram of all visited states up to the move n. This histogram forms a *posterior* probability distribution which evolves with every move n. This posterior distribution is induced by the constructed Markov chain.

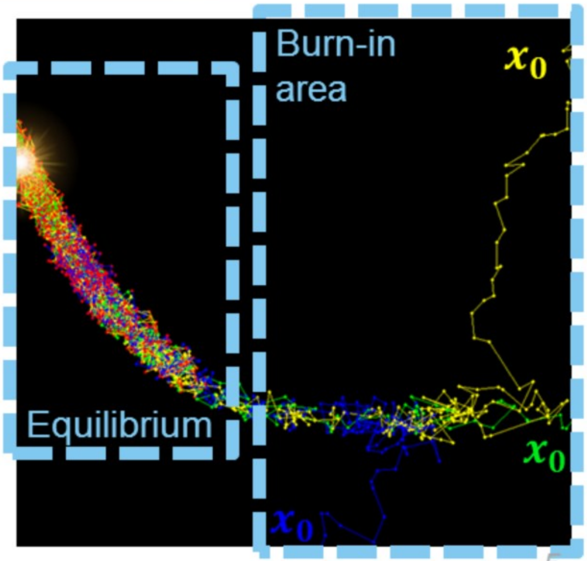
One important property of the transition probability is called *detailed balance*. It implies symmetry of the transition probability, i.e. the transition from state i to state j is equally probable as the transition back from j to i. Intuitively that means that the random walk can be reversed. A Markov chain running with such symmetric transition probability is called a *reversible* Markov chain. Physically speaking, detailed balance implies that, the random walk from state *i* to state *j* is always compensated by the reverse random walk, keeping the system in equilibrium.



Markov Chain

SIGGRAPH2013 The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- Posterior converges to the *target* distribution *if* the detailed balance obeyed and all states are reachable (*ergodicity*)
- With "bad" initial state x_0 the *start-up bias* (burn-in phase) can be significant



5

If the detailed balance is obeyed and all states of the path space can be reached by the proposed transition rules the proposal distribution then a Markov chain converges to a unique stationary distribution as number of moves $n \to \infty$.

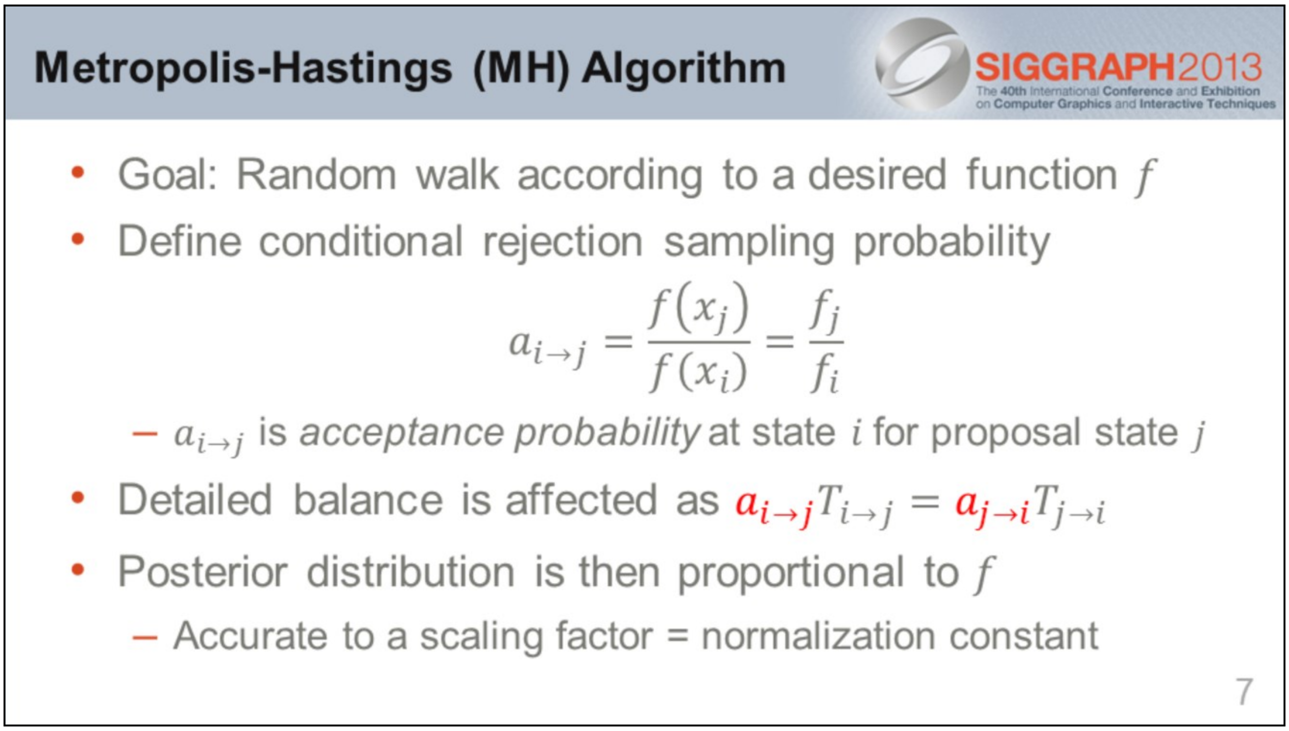
Such stationary distribution is called an *equilibrium* or *target distribution* of a constructed Markov chain and the chain is called *ergodic*.

The trajectory to the target distribution can take many steps and depends on the initial state x_0 of the Markov chain. Illustration on the right show three different Markov chains (yellow, green and blue) converging to the same equilibrium distribution from different initial states. The highlighted equilibrium zone roughly denote the high-probability region of the unimodal banana-shaped target distribution and receives the most samples.

The zone on the right half of the illustration shows the so-called "burn-in zone" – the phase when a Markov chain is located in the regions of low probability of the target distribution. If such phase it too long, it can shift the resulting posterior distribution from the target distribution, because a lot of visited states can lie in the low-probability regions of the target distribution. Thus this effect caused by a not properly chosen initial state is called *start-up bias*.

So practically it is very important to seed a Markov chain with an initial state that lies in high probability regions of the target distribution. That leads to faster convergence of the posterior distribution to the target distribution.

Metropolis-Hastings Algorithm



Metropolis-Hastings (MH) Algorithm

SIGGRAPH2013 The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- Goal: Random walk according to a desired function f
- Define conditional rejection sampling probability

$$a_{i\to j} = \frac{f(x_j)}{f(x_i)} = \frac{f_j}{f_i}$$

  - $a_{i\to j}$ is *acceptance probability* at state i for proposal state j
- Detailed balance is affected as $a_{i\to j}T_{i\to j} = a_{j\to i}T_{j\to i}$
- Posterior distribution is then proportional to f
  - Accurate to a scaling factor = normalization constant

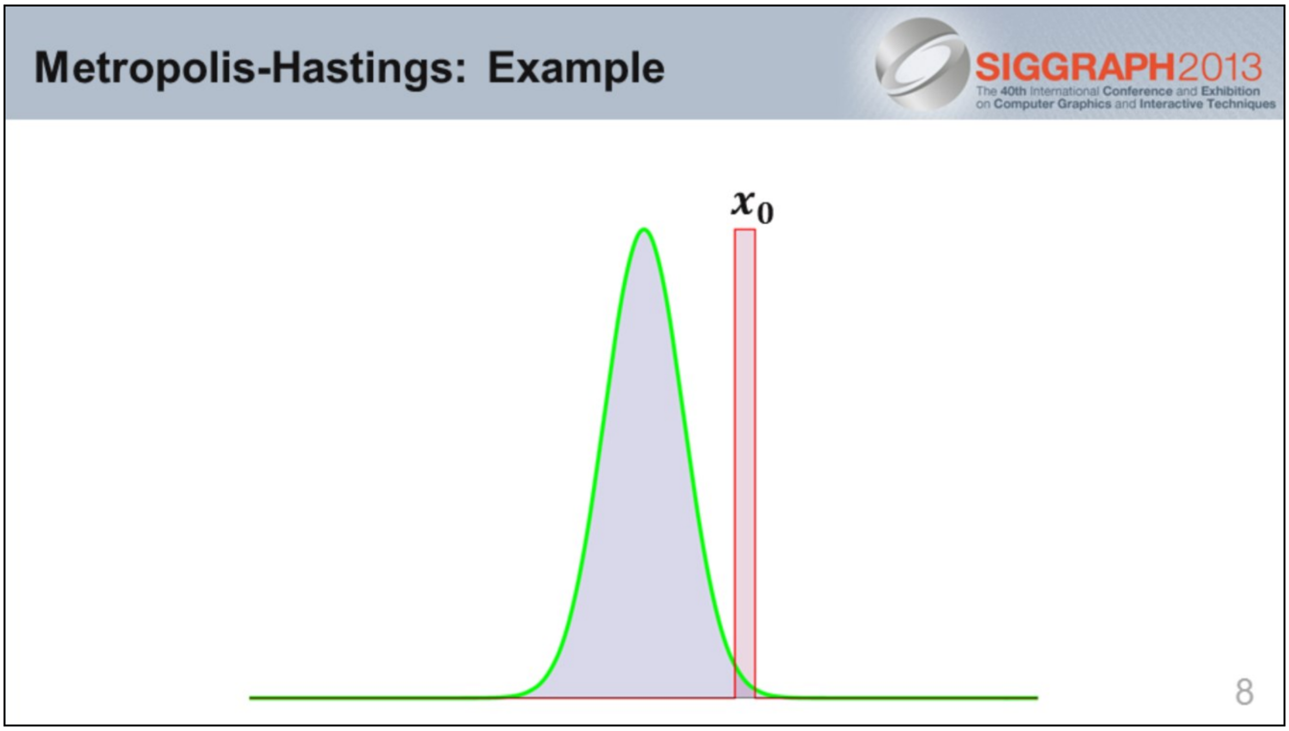
7

Assume we have some function f that can be evaluated point-wise. We cannot directly sample from f. However we still want to sample proportionally to f.

We can construct a Markov chain, whose posterior distribution converges to the function of interest f accurate to the normalization factor (since f is generally not normalized). This method was first proposed by Metropolis in 1953 and then generalized by Hastings in 1970 for arbitrary target distributions. The idea is to alter the transition distribution by affecting it by a conditional rejection sampling probability based on the desired target distribution. This probability is similar to the ordinary rejection sampling probability, however the key difference is that it is conditional on the current state, that is, $a_{i\to j} = \frac{f_j}{f_i}$ means that it's a probability of conditionally accepting the new proposal state j given current state i. It is called an acceptance probability because at each move n of the Markov chain we either accept the proposal state j with probability $a_{i\to j}$ or otherwise reject it and keep the current state i (with probability $1 - a_{i\to j}$ accordingly).

Given that the transition probability is selected such that the detailed balance is obeyed, the posterior distribution of the constructed Markov chain will converge to the desired target function f. Note that the detailed balance equation is affected by this acceptance probability. Important to note that the Metropolis-Hastings algorithm always constructs a normalized pdf, while the original target function is not necessarily (and is usually not) normalized. Thus this poses another important problem of finding the normalization constant of f, i.e. the integral of f over the whole state space. This can be as hard as the original problem. However we will see that for particular needs of light transport it can be easily estimated using one of alternative methods.

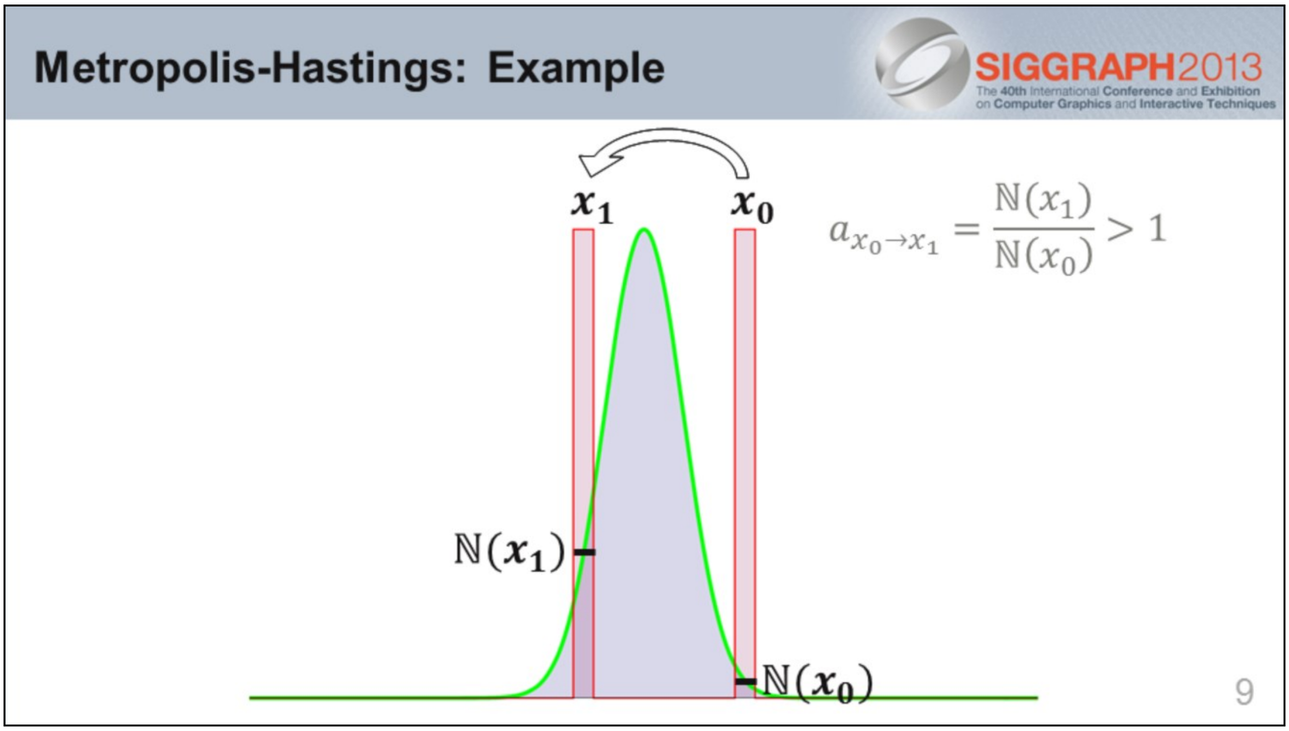
Metropolis-Hastings: Example



Here we show a simple step-by-step example of Metropolis-Hastings algorithm in action. We consider a simple 1D case of unimodal normal distribution $\mathbb{N}$ depicted in green. We will run Markov chain with uniform random walk and the acceptance probability being the ratio of values of proposal to current state, as was described.

We start with some initial state x_0 chosen to be close to the high-density region of the target function.

This already forms the posterior distribution with one bar of the histogram, depicted in red under the position of the initial state.



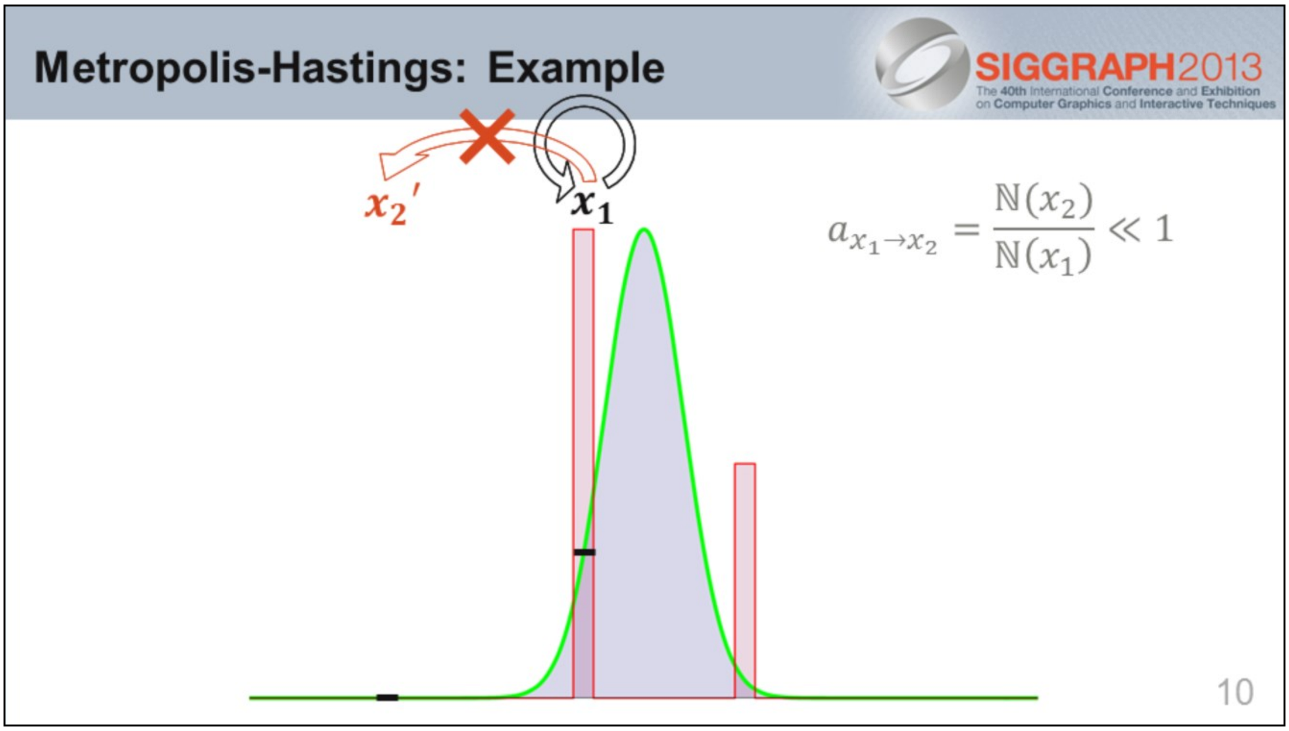
Metropolis-Hastings: Example

SIGGRAPH2013 The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

$$a_{x_0\to x_1} = \frac{\mathbb{N}(x_1)}{\mathbb{N}(x_0)} > 1$$

9

In order to make the first move we generate a proposal state x_1 with uniform random walk and compute the acceptance probability as $a_{x_0\to x_1} = \frac{\mathbb{N}(x_1)}{\mathbb{N}(x_0)} > 1$. That implies an unconditional acceptance of the new proposal state.



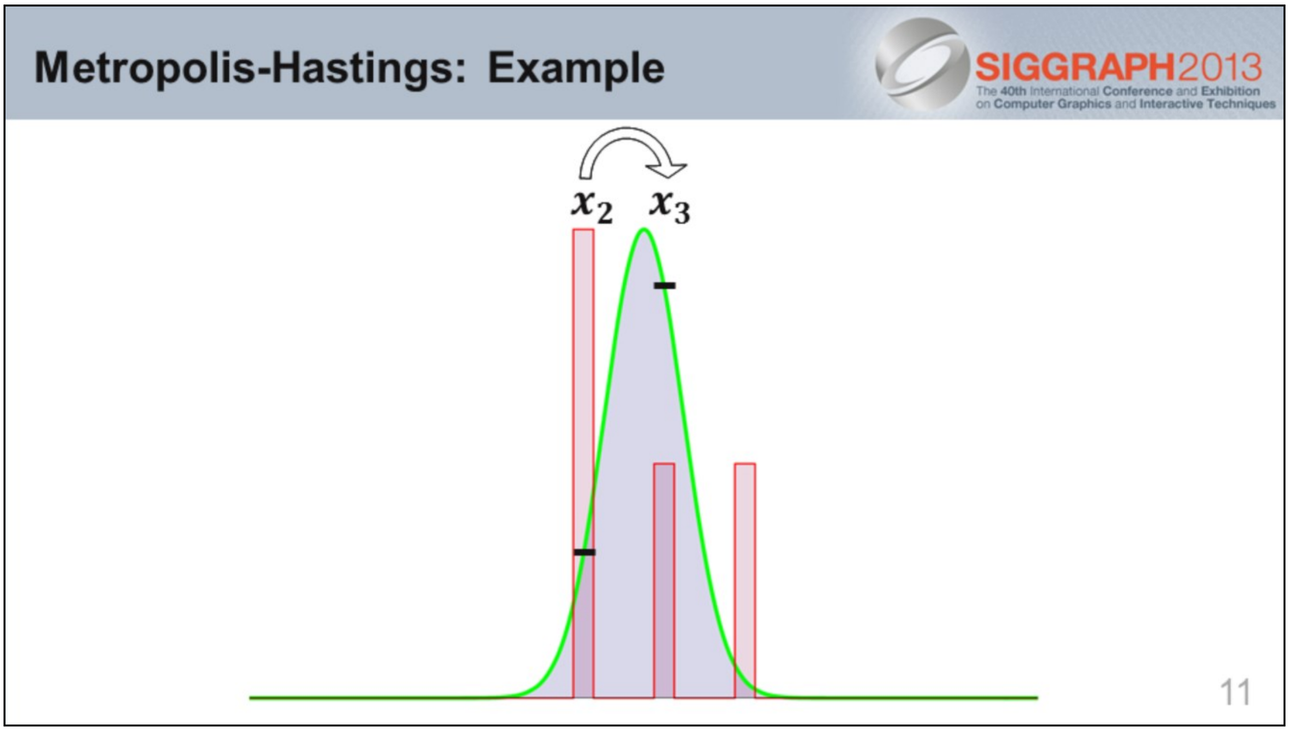
Metropolis-Hastings: Example

SIGGRAPH2013
The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques
x_2'
x_1
10

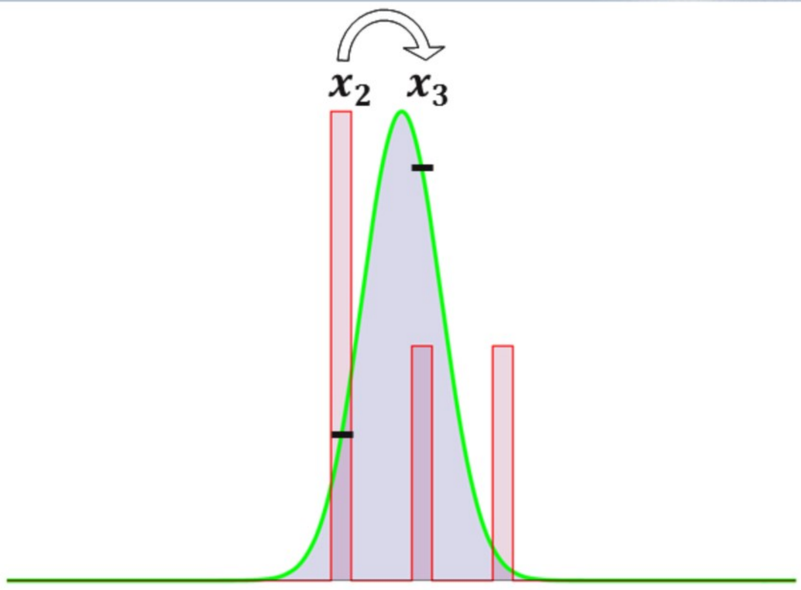
$$a_{x_1 \to x_2} = \frac{\mathbb{N}(x_2)}{\mathbb{N}(x_1)} \ll 1$$

At the second step again we generate a new proposal, this time from a low-density region. Thus the acceptance probability $a_{x_1 \to x_2} = \frac{\mathbb{N}(x_2)}{\mathbb{N}(x_1)}$ is very low. Thus if we make a move, most probably such a proposal will be rejected.

It is important to note at this point that the current state of the Markov chain does not change (x_2 becomes x_1), however every move affects the histogram. Thus the peak at the current state become more prominent.

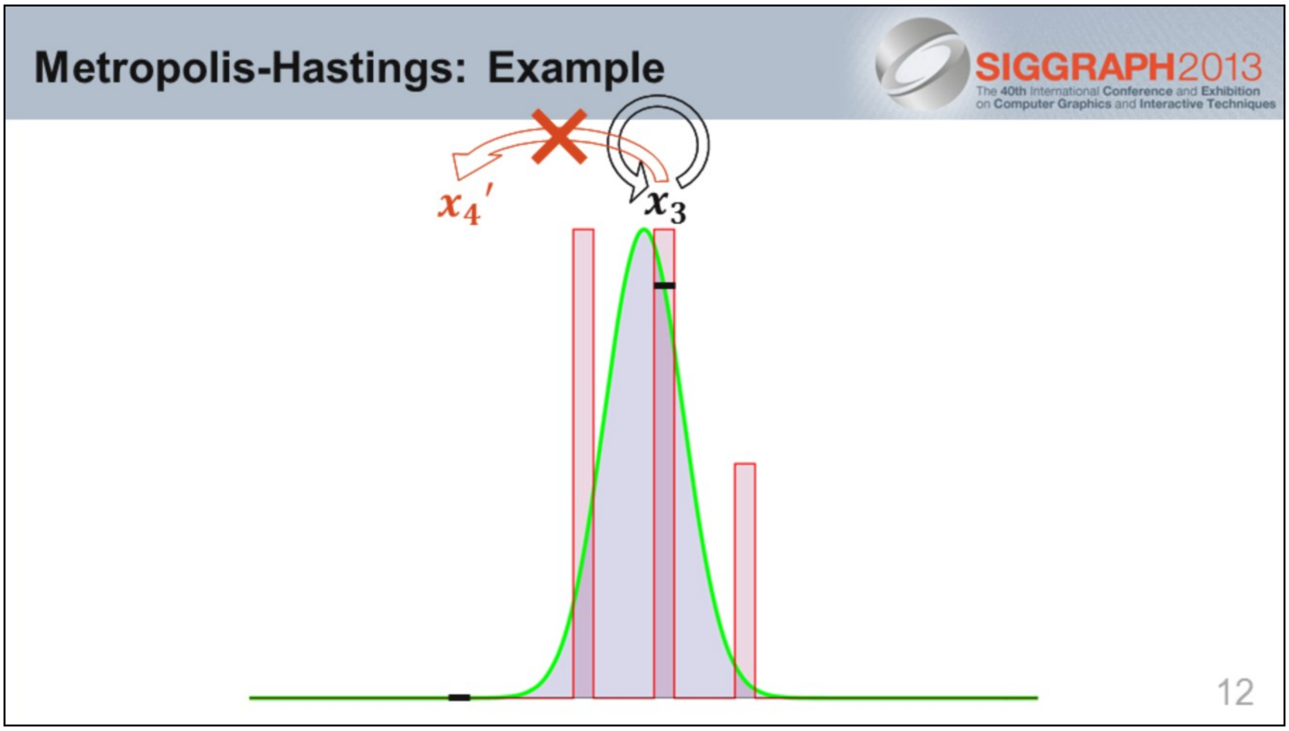


Metropolis-Hastings: Example

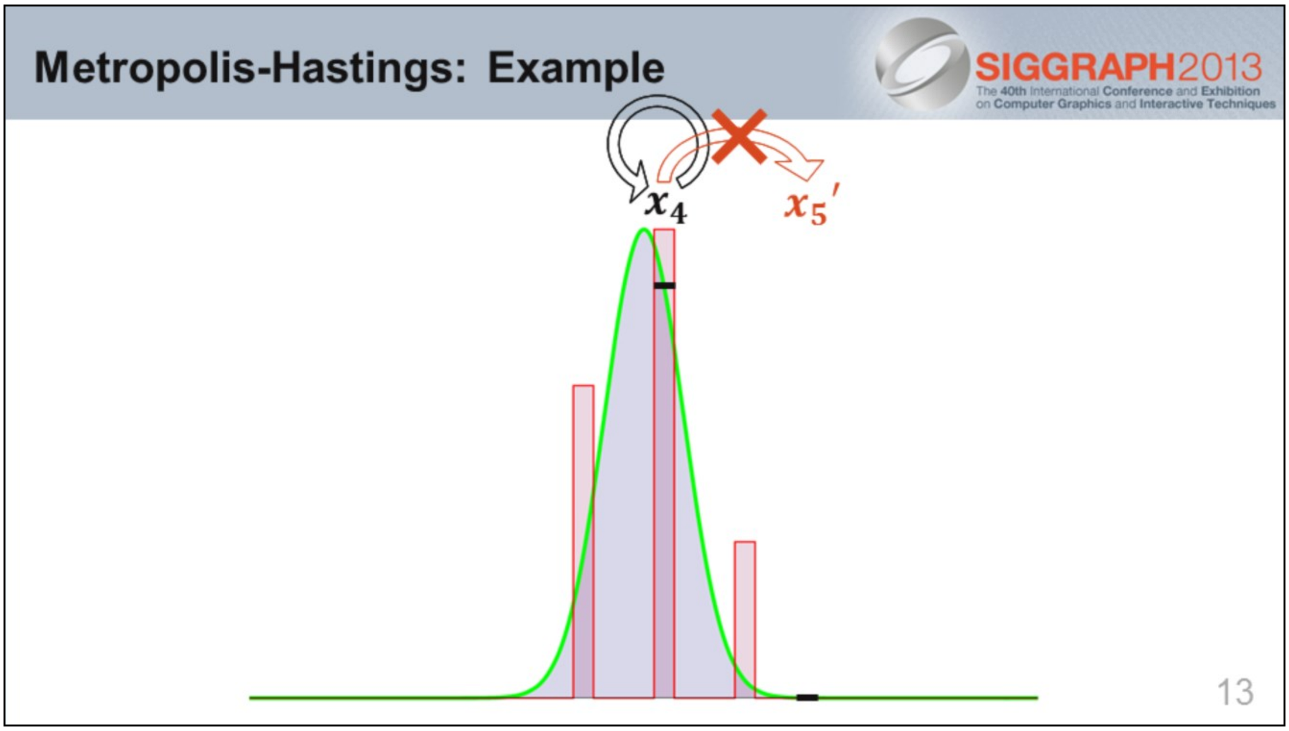


Now we generate a proposal in a high-density region. That leads to unconditional acceptance of the new proposal x_3.

Metropolis-Hastings: Example



The next proposal is also rejected due to the low value of the target function. Again that leads to one more sample added to the histogram at the current state.



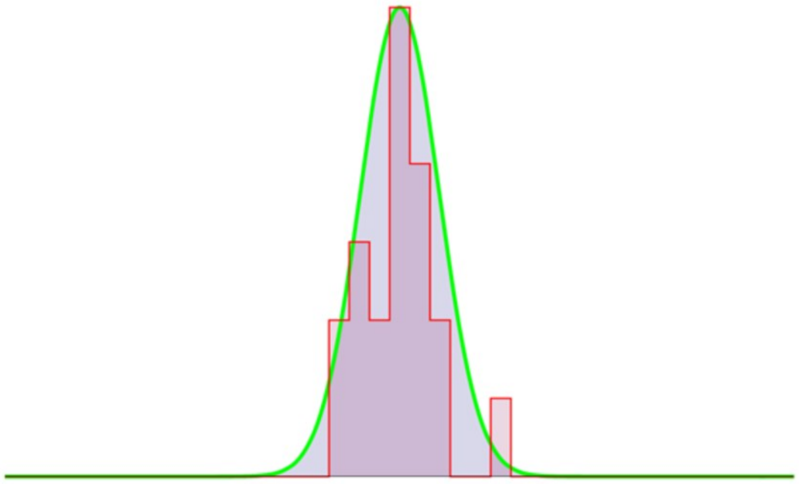
Metropolis-Hastings: Example

One more time the proposal is rejected. This is the natural behavior if the chain is stuck in a very high-density region of the target function. Note that many sequential rejections might also increase the correlation of samples, thus slowing down the convergence.

Metropolis-Hastings: Example

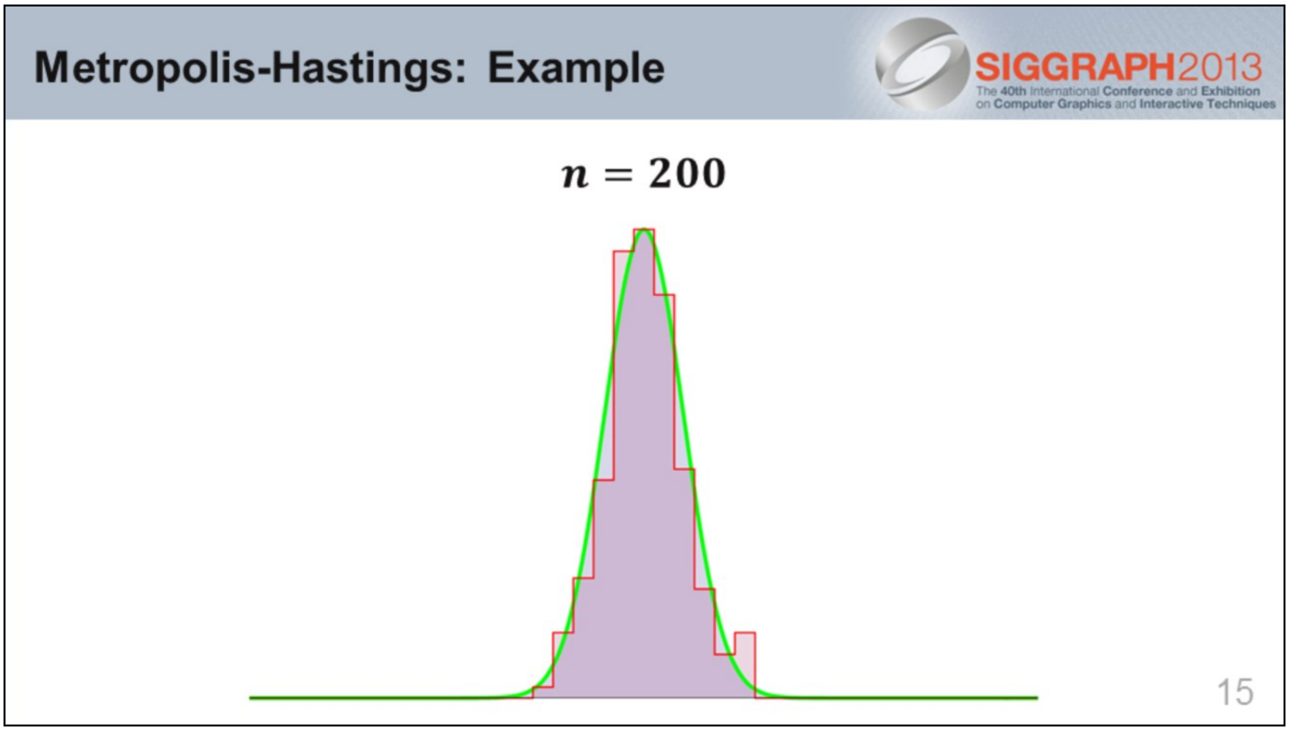
SIGGRAPH2013
The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

$n = 20$



14

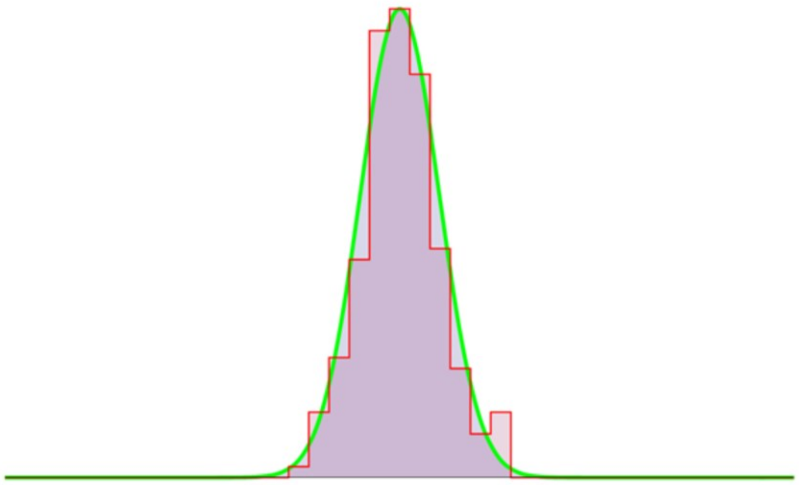
This is the posterior distribution produced by the Markov chain after 20 moves.



Metropolis-Hastings: Example

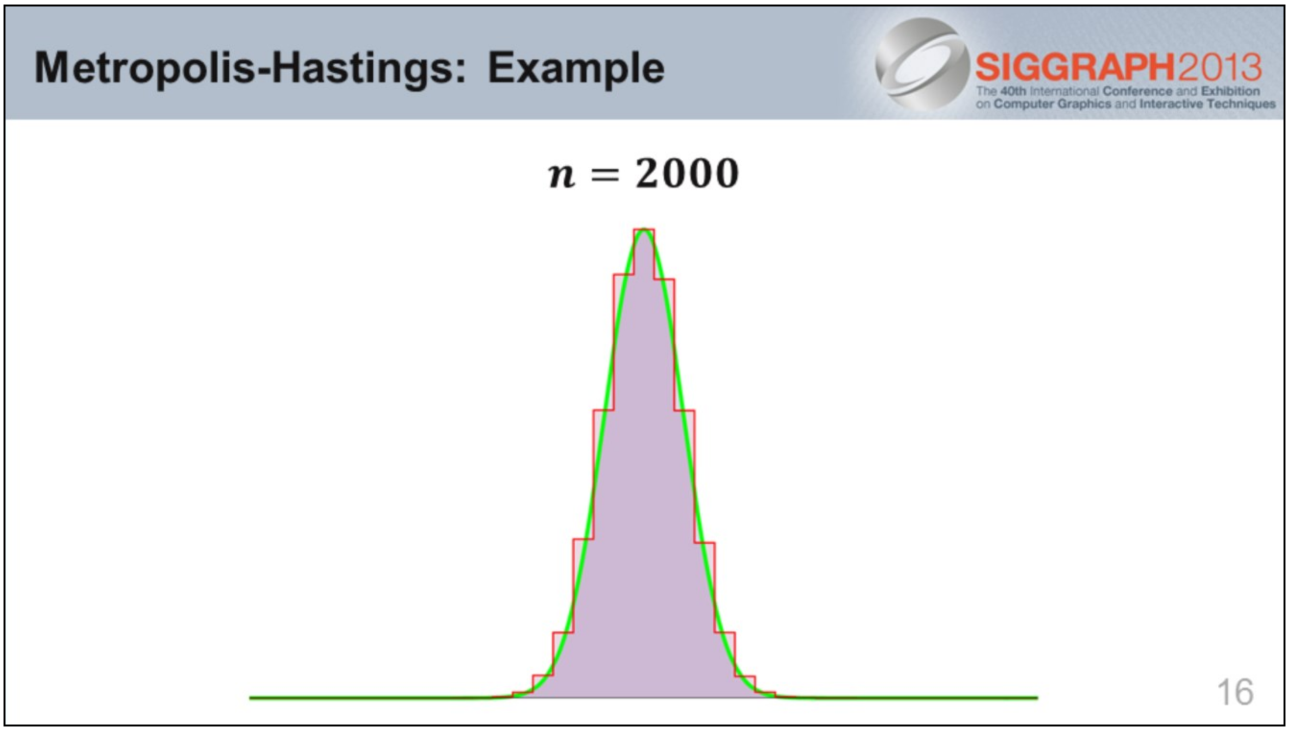
SIGGRAPH2013
The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

$$n = 200$$



15

This is the posterior distribution produced by the Markov chain after 200 moves.



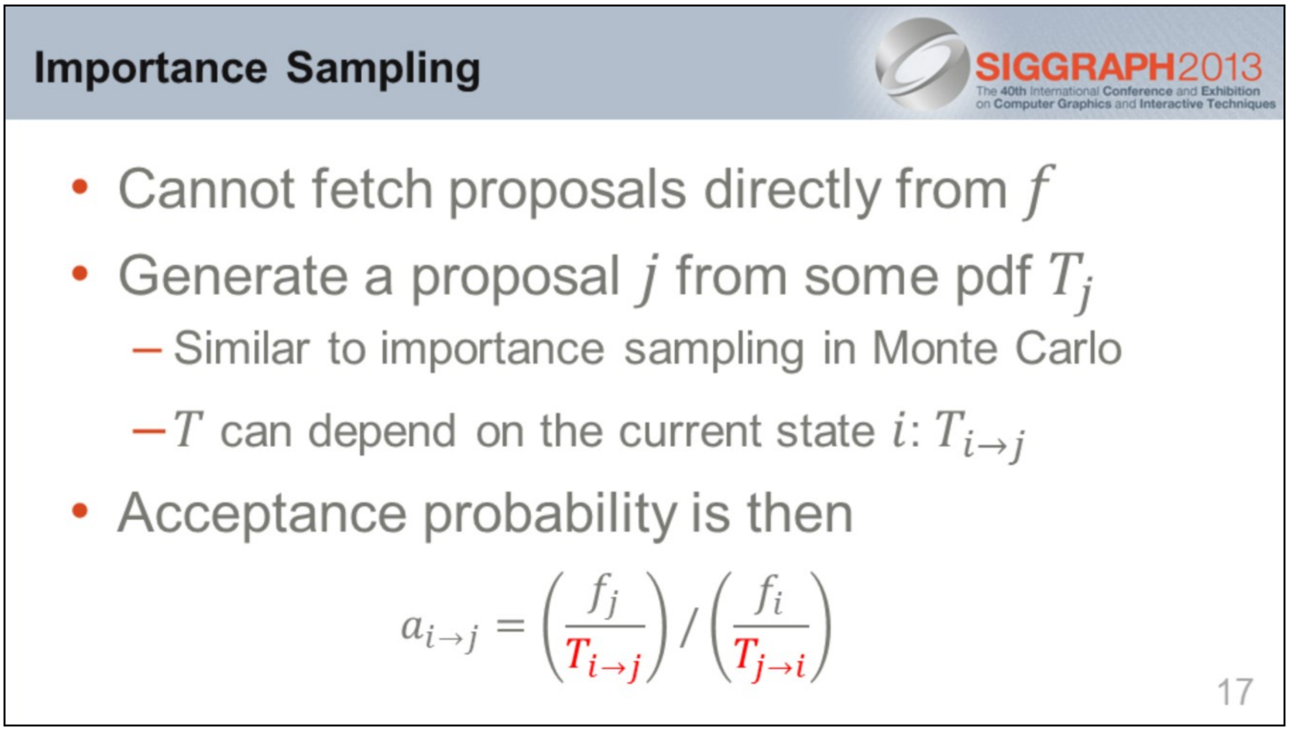
Metropolis-Hastings: Example

SIGGRAPH2013
The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

$n = 2000$

16

This is the posterior distribution produced by the Markov chain after 2000 moves. As we can see the posterior distribution closely converges to the target function.



Importance Sampling

SIGGRAPH2013 The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- Cannot fetch proposals directly from f
- Generate a proposal j from some pdf T_j
  - Similar to importance sampling in Monte Carlo
  - T can depend on the current state i: $T_{i\to j}$
- Acceptance probability is then

$$a_{i\to j} = \left(\frac{f_j}{T_{i\to j}}\right) / \left(\frac{f_i}{T_{j\to i}}\right)$$

17

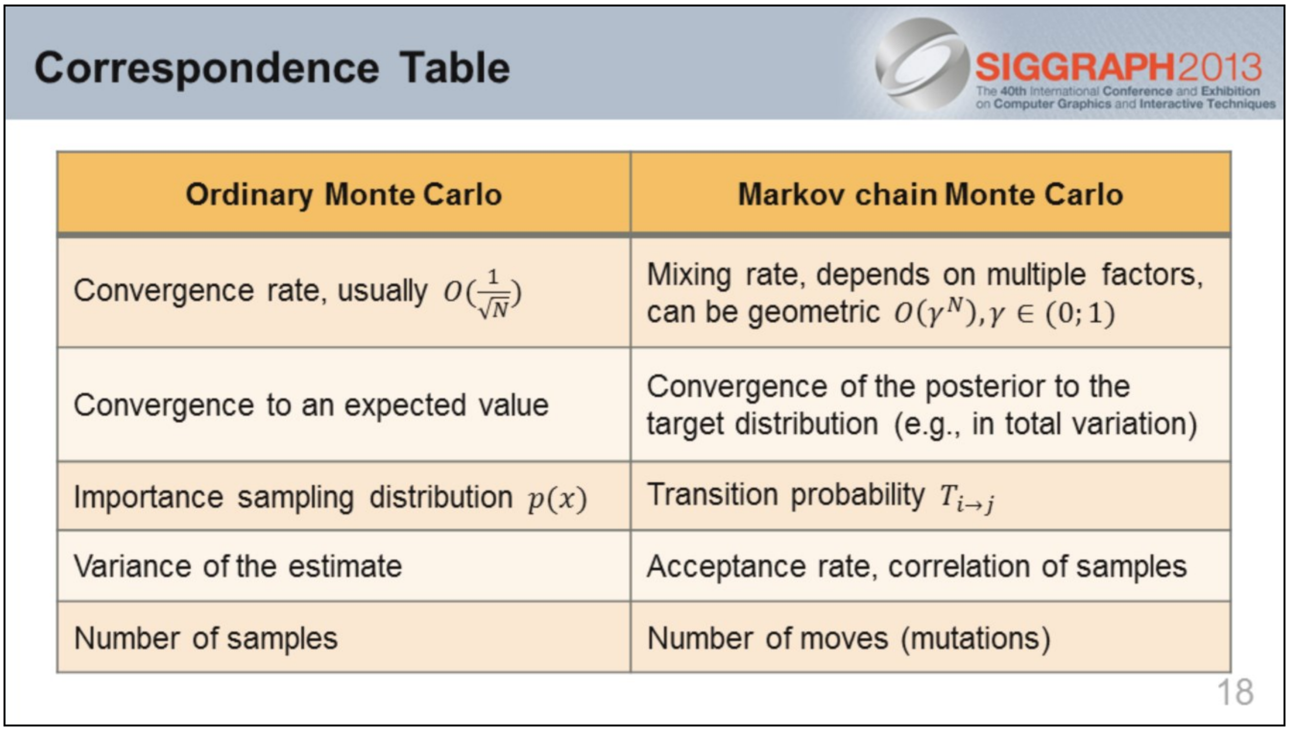
In practical situations, doing just a uniform random walk around the state space might lead to a poor exploration of state space (important features might be missing or under-sampled).

In this case, one can generate new proposals according to some importance function *T*, which is somewhat similar to *f*.

This is almost equivalent to the importance sampling technique used in Monte Carlo.

The key difference is that such importance function in MCMC can also depend and rely on the current state! This function is called transition probability function.

And the acceptance rate should account for this transition probability similarly to Monte Carlo methods (by dividing the value of *f* by the probability of sampling it).



Correspondence Table

SIGGRAPH2013 The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

Ordinary Monte Carlo	Markov chain Monte Carlo
Convergence rate, usually $O(\frac{1}{\sqrt{N}})$	Mixing rate, depends on multiple factors, can be geometric $O(\gamma^N), \gamma \in (0;1)$
Convergence to an expected value	Convergence of the posterior to the target distribution (e.g., in total variation)
Importance sampling distribution $p(x)$	Transition probability $T_{i\to j}$
Variance of the estimate	Acceptance rate, correlation of samples
Number of samples	Number of moves (mutations)

18

This table shows the correspondence between the major terms and properties used in ordinary Monte Carlo (MC) to the equivalent terms used in Markov Chain Monte Carlo (MCMC).

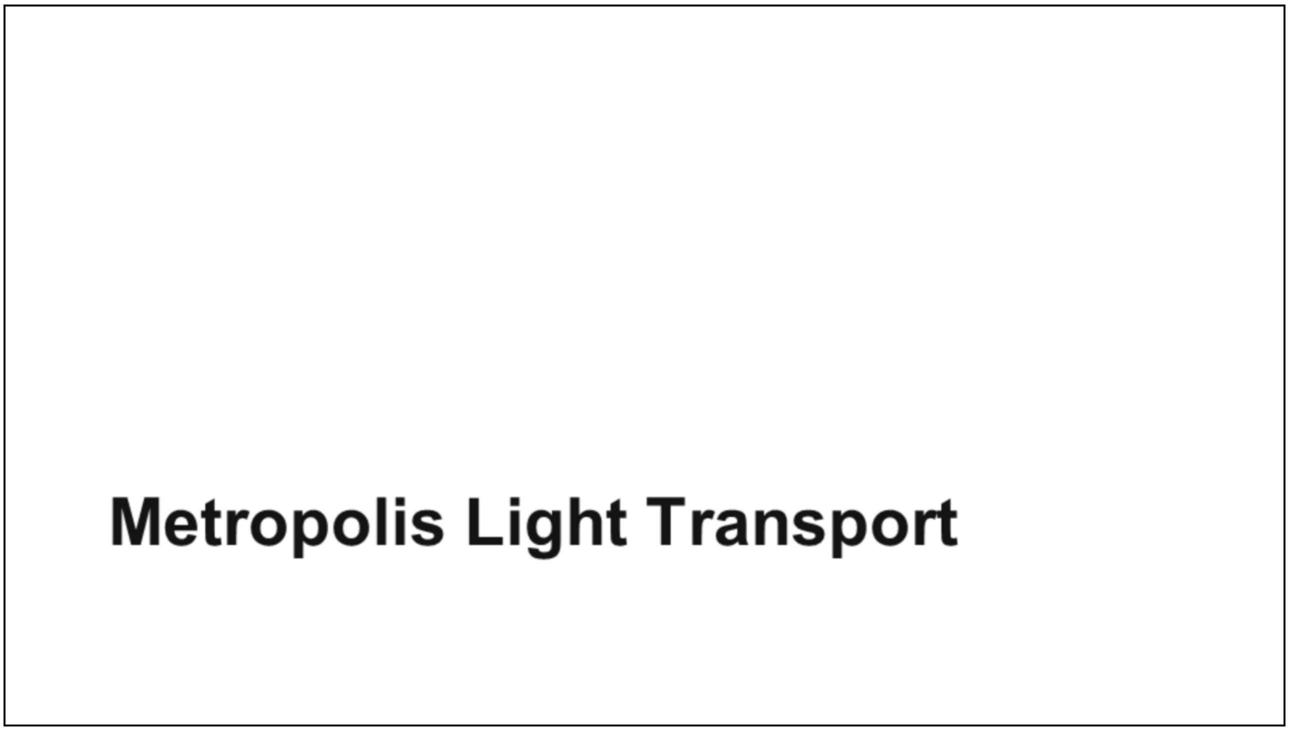
Note that the theoretical convergence of MCMC methods can be fundamentally different comparing to ordinary MC methods.

The convergence in context of Markov chains is usually referring to the convergence of the posterior distribution to the target distribution (in some norm, for example, in total variation).

We have transition probability instead of importance function. Note that we have much more freedom in constructing this transition probability function, because we can also rely on the current state of a Markov chain.

The error of MH is very hard to compute, since the samples are inherently correlated. So, we cannot use just variance anymore. Acceptance rate can be a good initial indicator of the MCMC sampler performance.

And, instead of number of samples, we have number of moves made by Markov chain.



Metropolis Light Transport

Now, I'll try to explain how we can apply MH algorithm in the context of light transport.

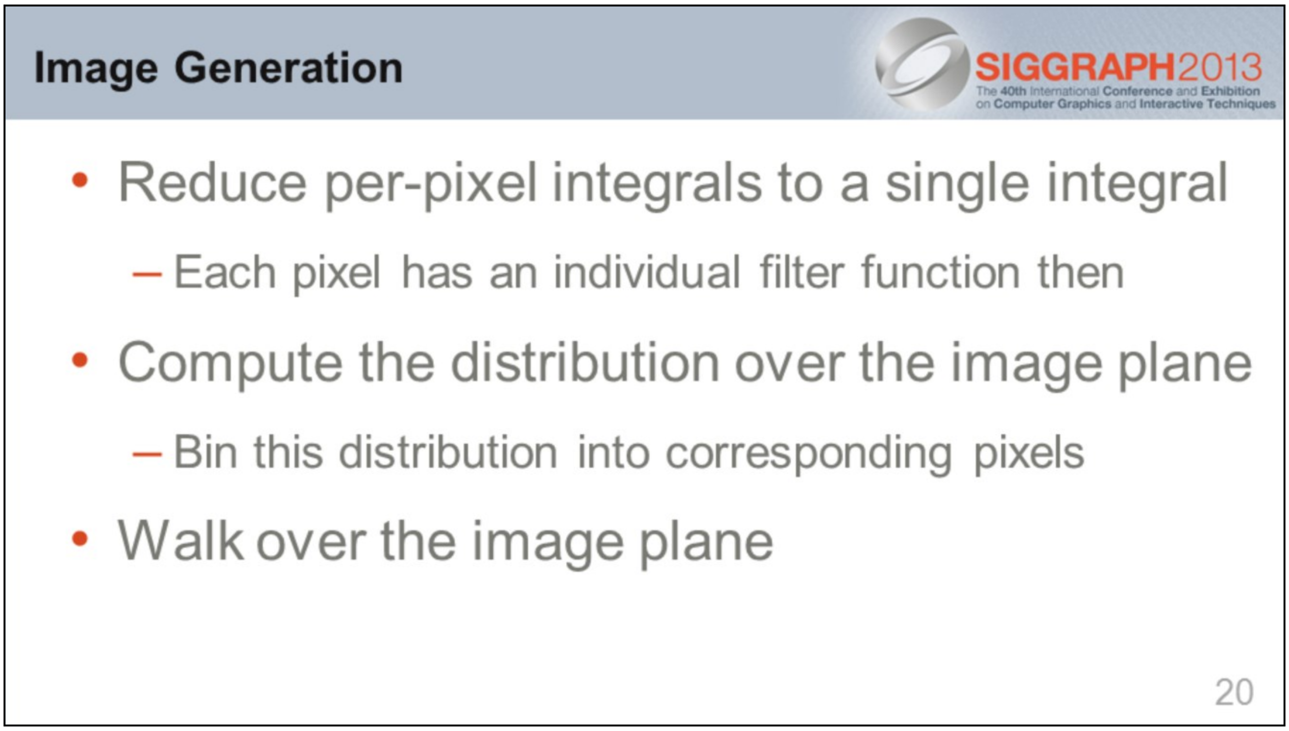


Image Generation

SIGGRAPH2013
The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- Reduce per-pixel integrals to a single integral
  - Each pixel has an individual filter function then
- Compute the distribution over the image plane
  - Bin this distribution into corresponding pixels
- Walk over the image plane

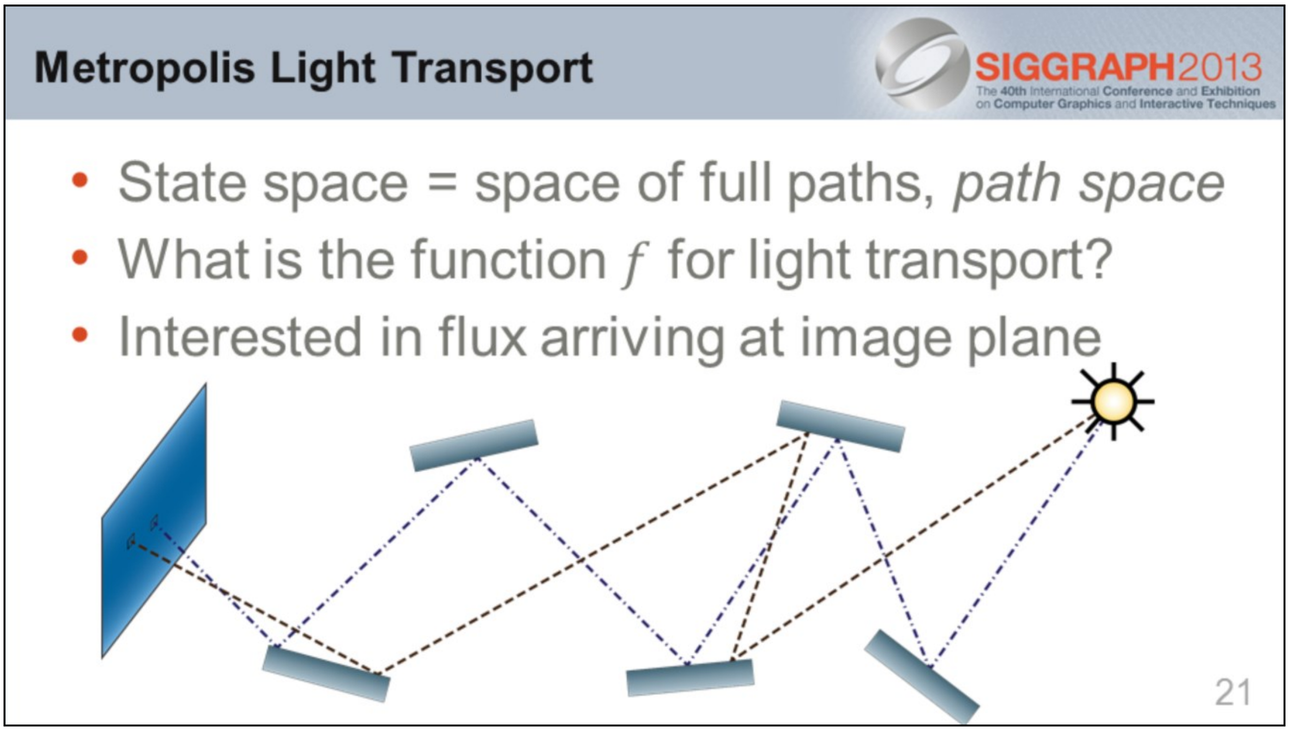
20

In order to achieve more efficient exploration of the path space and utilize the potential correlation between the separate integrals for each pixel, we reduce the task to a single integral. Each pixel integral can be then deduced by applying a pixel filter.

This single integral computes the distribution of flux on the image plane.

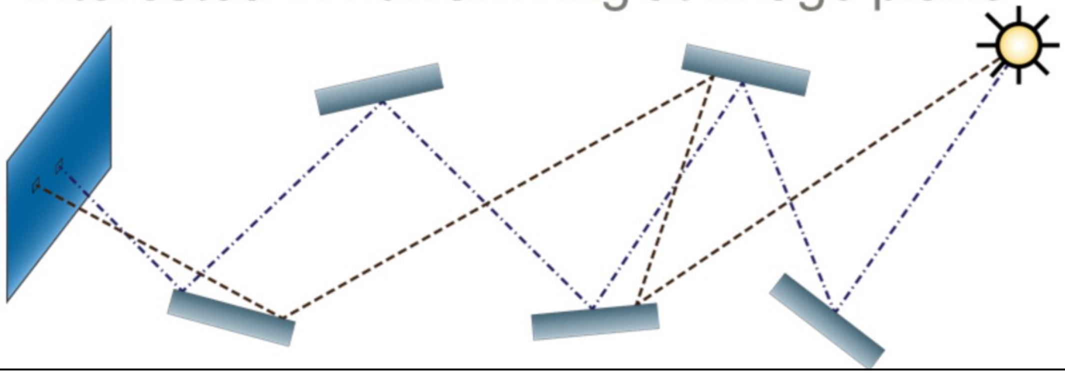
Then we can obtain the image by just distributing the corresponding samples from the posterior distribution into the corresponding bins of image pixels.

This way the MH algorithm is able to freely walk over the complete image plane while exploring important parts of the image adaptively.



Metropolis Light Transport

- State space = space of full paths, *path space*
- What is the function f for light transport?
- Interested in flux arriving at image plane



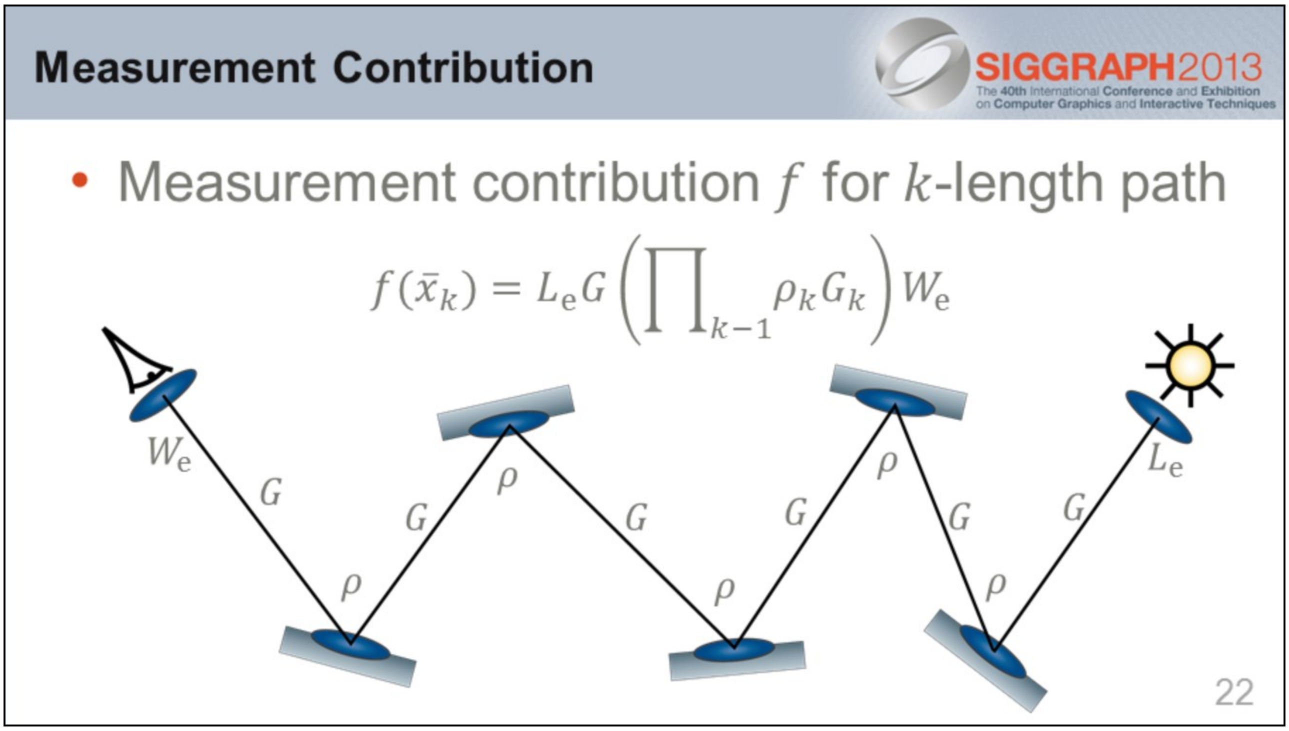
21

So, ideally, we are interested in all possible trajectories (paths) from light source to camera.

This naturally forms the state space for a Markov chain: now each state is a full path from light source to the camera. This state space is called a *path space* in light transport.

How would one define a target function for Metropolis-Hastings algorithm in context of light transport?

Ideally, we're interested in the equilibrium distribution of flux incident to the image plane.

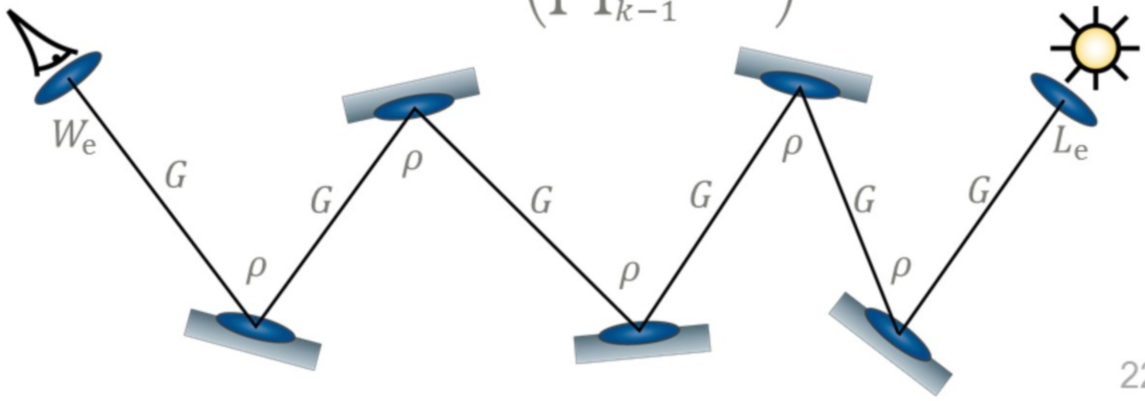


Measurement Contribution

SIGGRAPH2013 The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- Measurement contribution f for k-length path

$$f(\bar{x}_k) = L_e G \left(\prod_{k-1} \rho_k G_k \right) W_e$$

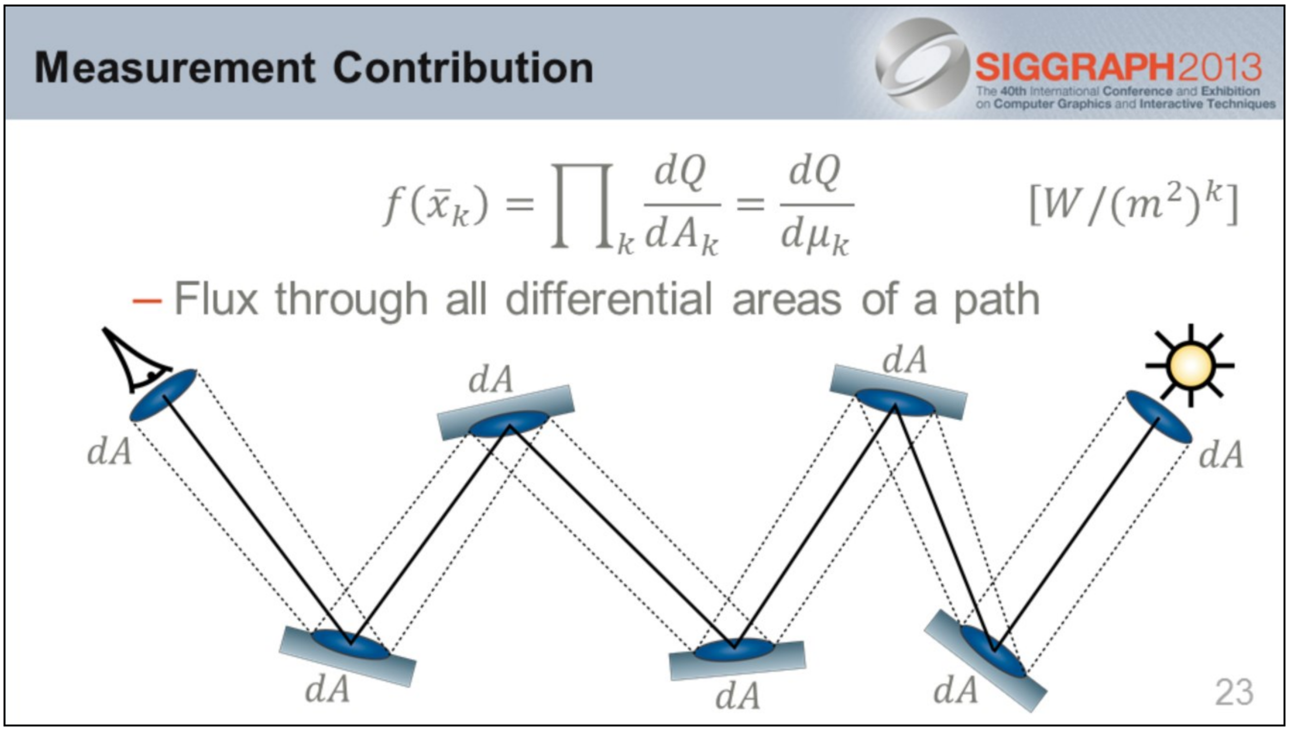


22

Thus, as was explained before by Jaroslav, we can introduce the measurement contribution function for a path x_k.

It consists of subsequently interleaved events: emission(L_e)->propagation(G)->scattering(f_r)->propagation(G)-> …. -> absorption by sensor (W_e)

and provides the contribution carried by the path.

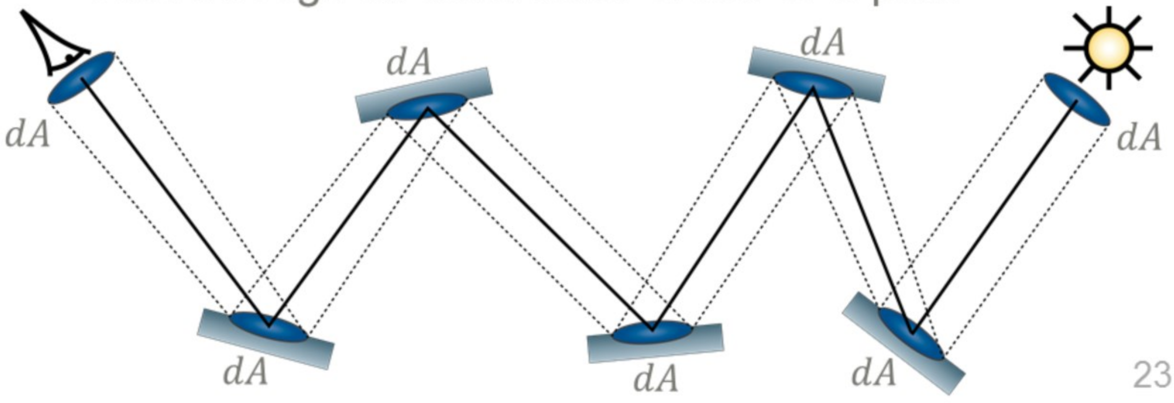


Measurement Contribution

SIGGRAPH2013
The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

$$f(\bar{x}_k) = \prod_k \frac{dQ}{dA_k} = \frac{dQ}{d\mu_k} \qquad [W/(m^2)^k]$$

– Flux through all differential areas of a path



23

Let's take a closer look at the physical meaning of the measurement contribution f.

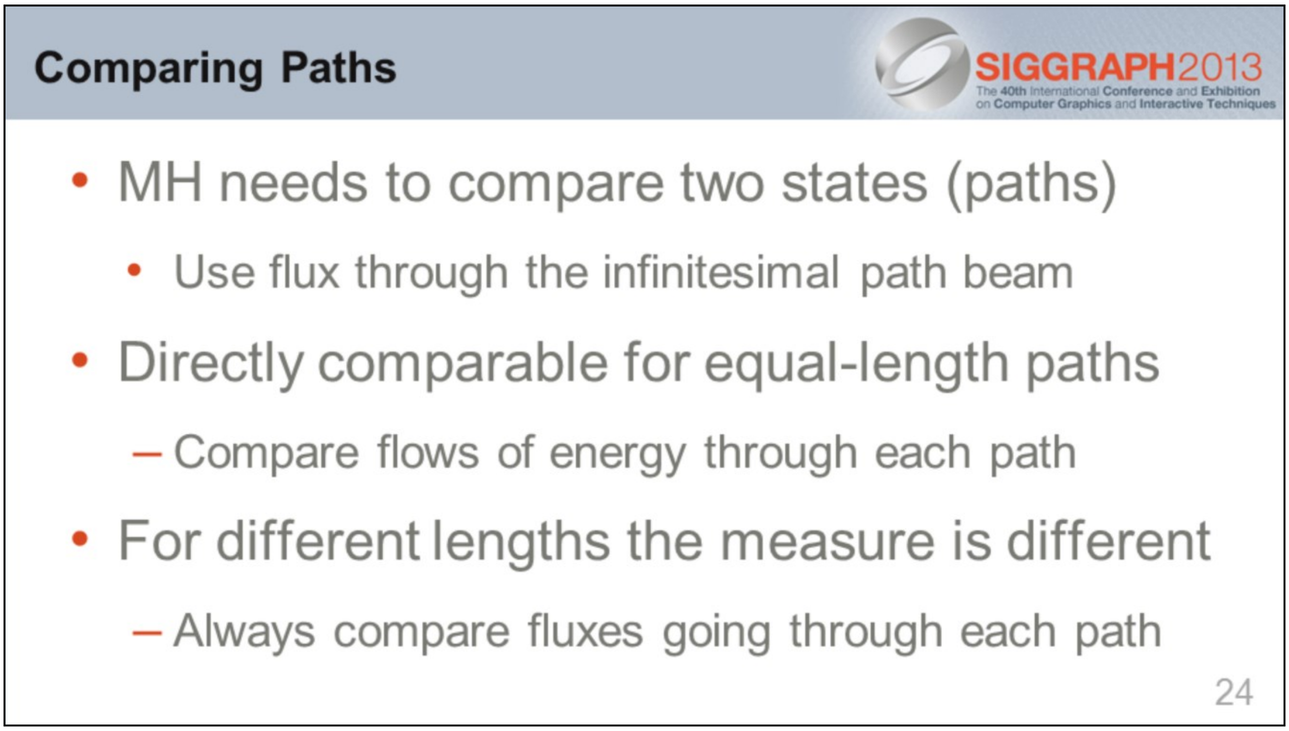
Eric Veach showed in his PhD thesis that we can define the measurement contribution as chain derivative of energy with respect to surface areas at each interaction. By folding the product, we get that the measurement contribution f is a derivative of energy function Q with respect to the area product measure μ_k.

The physical quantity of the measurement contribution is Watts per square meter to the kth power.

Intuitively, it defines an energy flow through a differential beam around the path.

In other words, we count the number of photons going through the infinitesimal beam around the path.

This definition reveals the underlying physical justification behind Metropolis light transport.



Comparing Paths

SIGGRAPH2013
The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- MH needs to compare two states (paths)
  - Use flux through the infinitesimal path beam
- Directly comparable for equal-length paths
  - Compare flows of energy through each path
- For different lengths the measure is different
  - Always compare fluxes going through each path

24

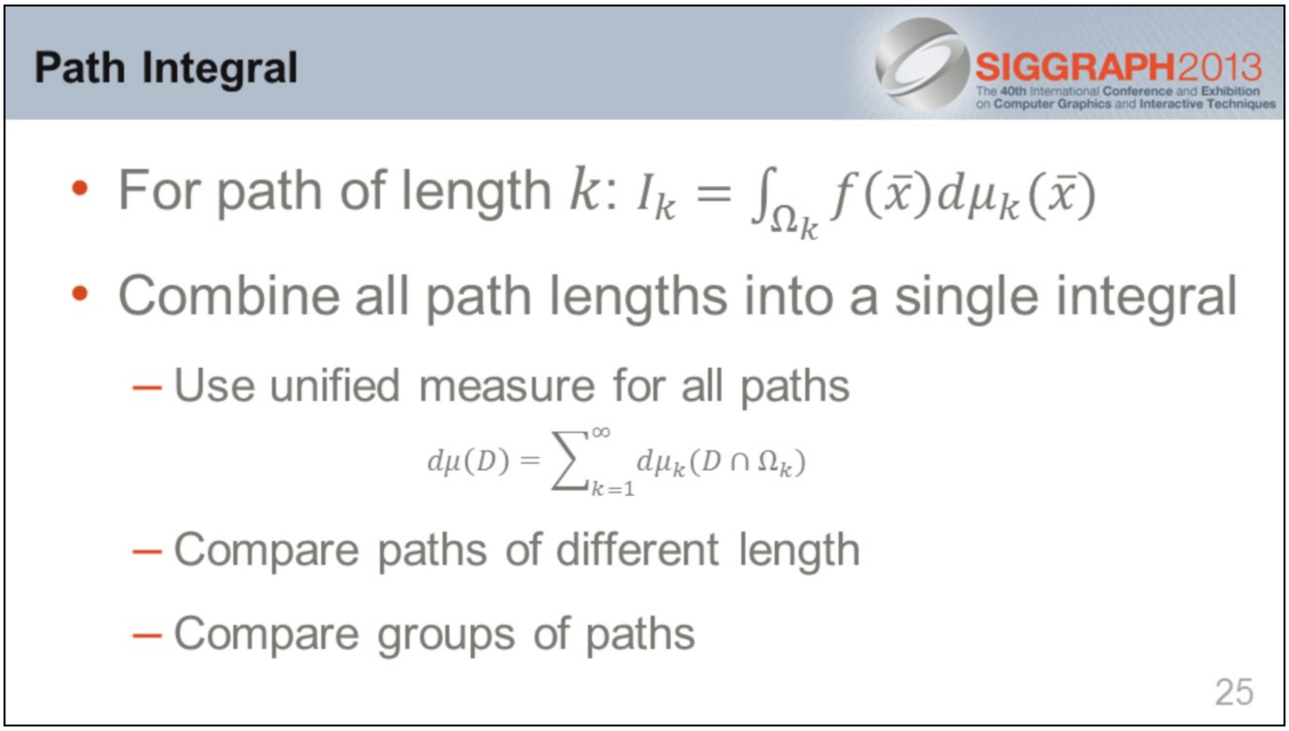
In context of MH, we always need to be able to compare two different paths.

As we know, the measurement contribution *f* provides the amount of flux going through the infinitesimal beam around the path.

This makes the paths of equal length directly comparable to each other in terms of the carried energy.

The only remaining question is how to compare paths of different lengths?

Interestingly, if we define our integration measure as a product area measure, which adjusts based on the path length, then we can directly compare the amount of energy (e.g. number of photons) going through the path.



Path Integral

- For path of length k: $I_k = \int_{\Omega_k} f(\bar{x}) d\mu_k(\bar{x})$
- Combine all path lengths into a single integral
  - Use unified measure for all paths
$$d\mu(D) = \sum_{k=1}^{\infty} d\mu_k(D \cap \Omega_k)$$
  - Compare paths of different length
  - Compare groups of paths

25

So, we can construct a MH integration process for all paths of the same length k.

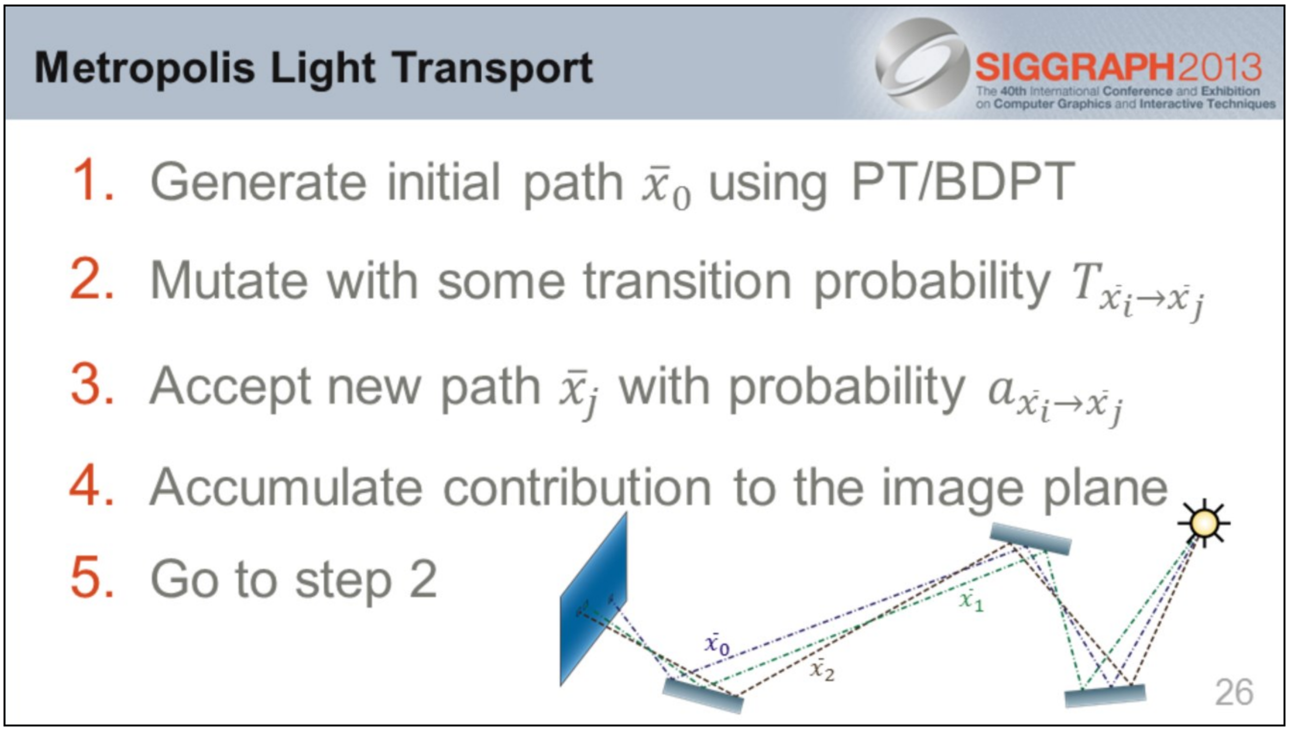
However, we need to construct a single generalized integral.

This can be done by just treating this family of integrals as a single generalized path integral.

In this case, we can introduce a generalized product area measure $d\mu$ as a sum measure.

This enables us to use a single integral for paths of all lengths for MH.

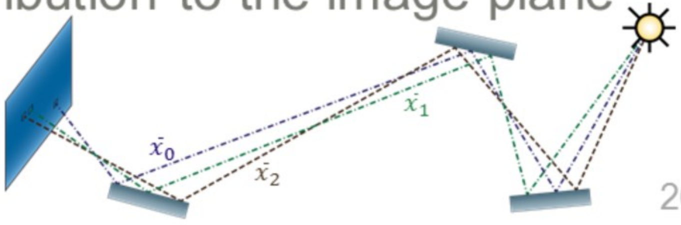
Moreover, this way we can compare all paths and even groups of paths with each other in the context of carried energy (flux).



Metropolis Light Transport

SIGGRAPH2013
The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

1. Generate initial path $\bar{x}_0$ using PT/BDPT
2. Mutate with some transition probability $T_{\bar{x_i} \to \bar{x_j}}$
3. Accept new path $\bar{x}_j$ with probability $a_{\bar{x_i} \to \bar{x_j}}$
4. Accumulate contribution to the image plane
5. Go to step 2

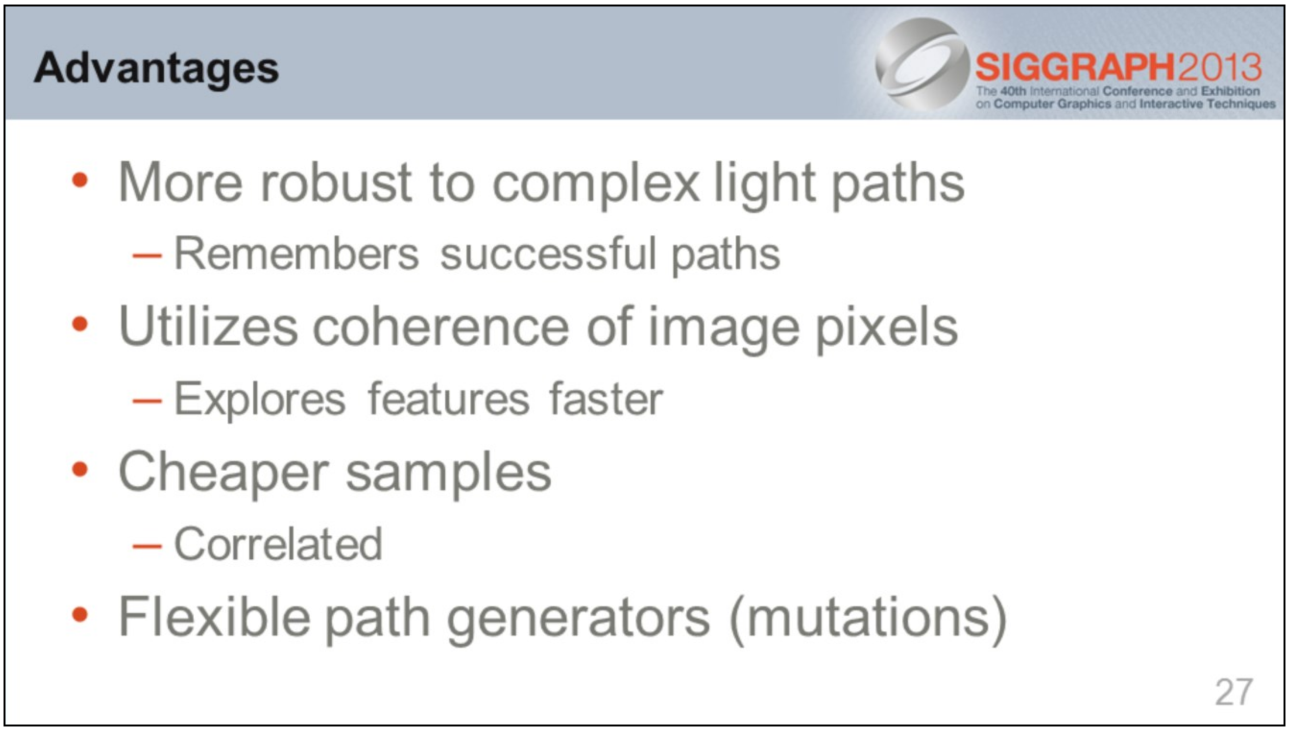


26

We showed how light transport can be reformulated for MH integration.

Now I'll outline the actual steps of the MLT algorithm:

1. We generate an initial state (full path) of a Markov chain using one of the existing sampling methods (e.g. PT or BDPT).
2. Then we start the actual mutation process. Mutate the current path using one of the available mutation strategies, compute the transition probability.
3. Compute the acceptance probability, accept the new proposed path according to this probability.
4. Accumulate the contribution of the current path to the image plane, apply pixel filter to bin the path into the corresponding pixel in-place.
5. Proceed to step 2 to cycle the random walk.



Advantages

SIGGRAPH2013
The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- More robust to complex light paths
  - Remembers successful paths
- Utilizes coherence of image pixels
  - Explores features faster
- Cheaper samples
  - Correlated
- Flexible path generators (mutations)

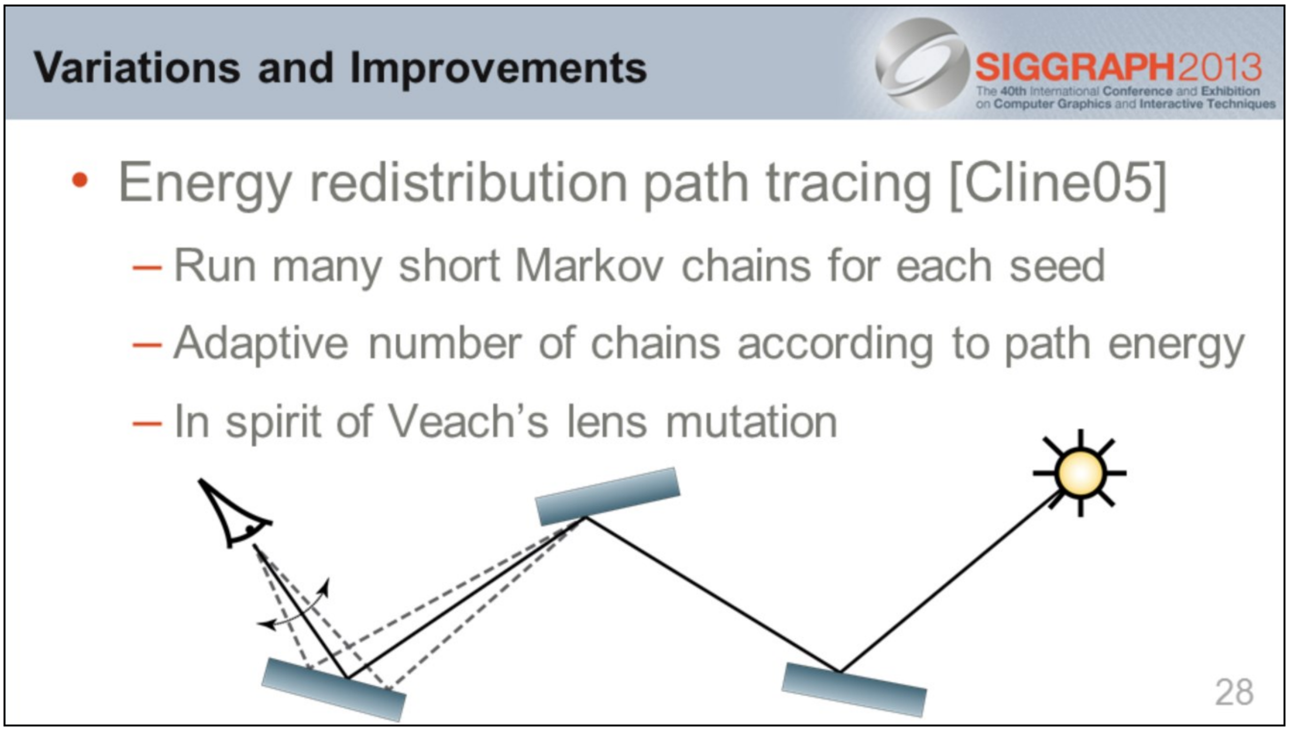
27

I'd like to emphasize that we have already quite good methods for image rendering, like BDPT.
So, why do we need yet another, way more complicated rendering method?

First of all, MLT is much more robust to the complex light paths, meaning that it tries to "remember" the successful paths. That is, the current state of a Markov chain is always a correct full path from light source to the camera.

As another advantage, Markov chain can easily explore the similar surrounding paths by perturbing the current path slightly, thus exploring the whole illumination features at a low cost. On the other hand, this can also cause some unwanted correlation of samples, slowing down the rendering convergence.

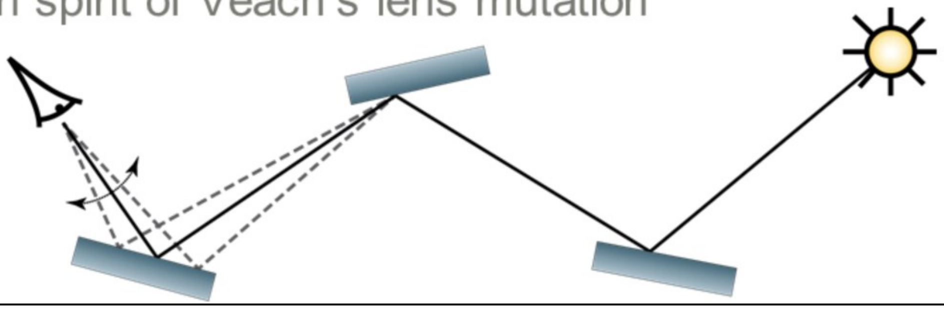
And last, but not the least, MLT framework provides us the great freedom of constructing path generators for almost any special situation.



Variations and Improvements

SIGGRAPH2013 The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- Energy redistribution path tracing [Cline05]
  - Run many short Markov chains for each seed
  - Adaptive number of chains according to path energy
  - In spirit of Veach's lens mutation



28

ERPT utilizes the fact that independent samplers, like BDPT, already provide a very good distribution.

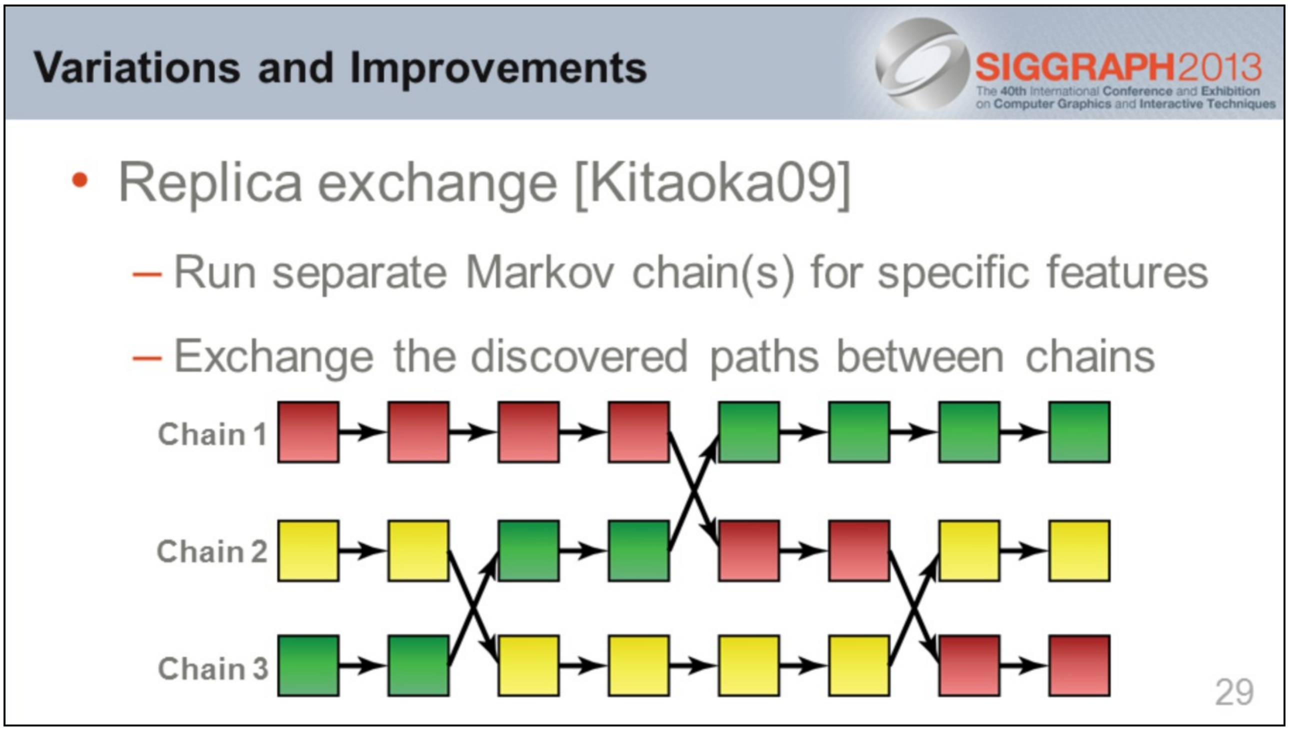
The idea is to try to redistribute the amount of energy carried by each initial path.

In order to do that, multiple Markov chains are started with the same seed path, the number of chains is computed adaptively based on the path energy.

This scheme is very similar to the lens mutation proposed by Eric Veach with the number of mutations between reseedings of the chain being very low.

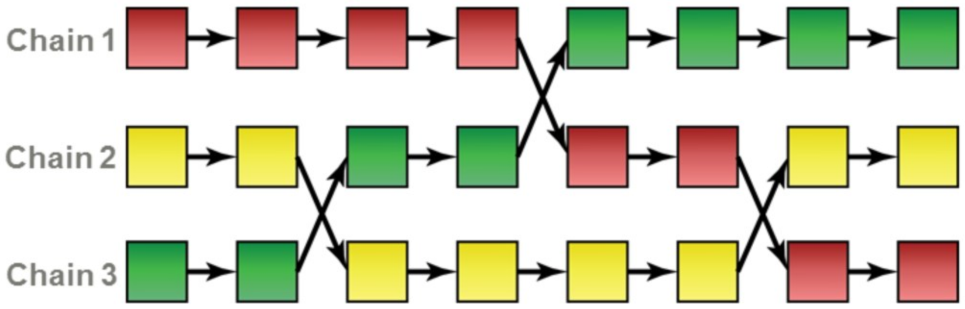
Efficiency of ERPT depends a lot on the seeding sampler – how good it is. E.g. BDPT w/o MIS provides very unbalanced sampling, leading to ERPT being stuck for a long time in some regions due to high redistribution workload.

Moreover, the distribution region is manually set, making it non-trivial to tweak the parameters to achieve the best redistribution vs. stratification trade-off.



Variations and Improvements

- Replica exchange [Kitaoka09]
  - Run separate Markov chain(s) for specific features
  - Exchange the discovered paths between chains



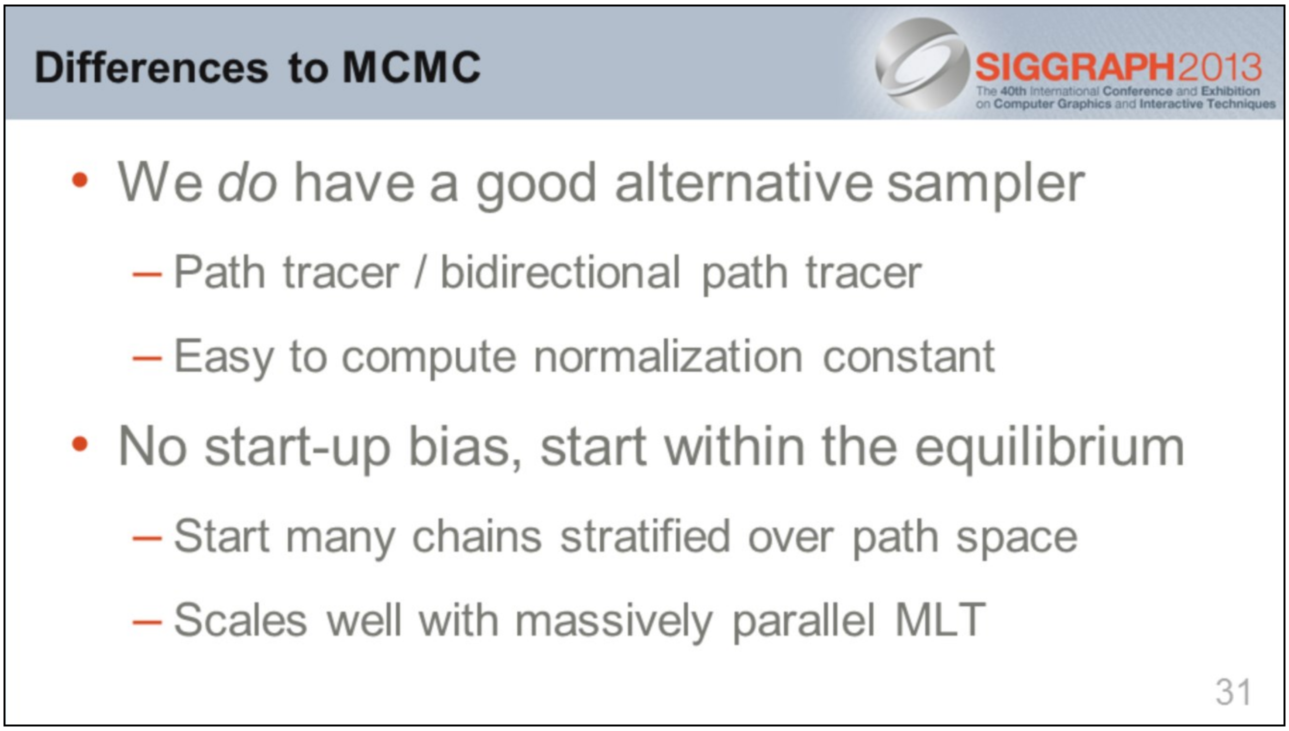
29

Replica exchange has been known for a while in statistical MCMC field and was recently introduced in context of light transport.

The idea is relatively simple. Imagine that one mutation strategy can easily discover important features and another mutation strategy can easily explore such features, but has difficulties discovering them.

In this case, we can run two separate chains with these two strategies, and once the "discovering" chain has found a feature, it passes this feature to another chain to explore it. This way the "discovering" chain is responsible for finding important features, while these features can be efficiently explored by the second chain.

Normalization and Start-up Bias in MLT



Differences to MCMC

SIGGRAPH2013 The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- We *do* have a good alternative sampler
  - Path tracer / bidirectional path tracer
  - Easy to compute normalization constant
- No start-up bias, start within the equilibrium
  - Start many chains stratified over path space
  - Scales well with massively parallel MLT

31

The key difference of MLT compared to the usual MCMC situation is that we already have methods that can generate an image well in most situations.

So, many regular MCMC problems like normalization constant estimation and start-up bias are solved easier.

For example, in order to compute the normalization constant, which is just an average flux received by the image plane, it is practically sufficient to sample a few hundred thousand paths with BDPT, which is a negligible cost comparing to the actual image rendering.

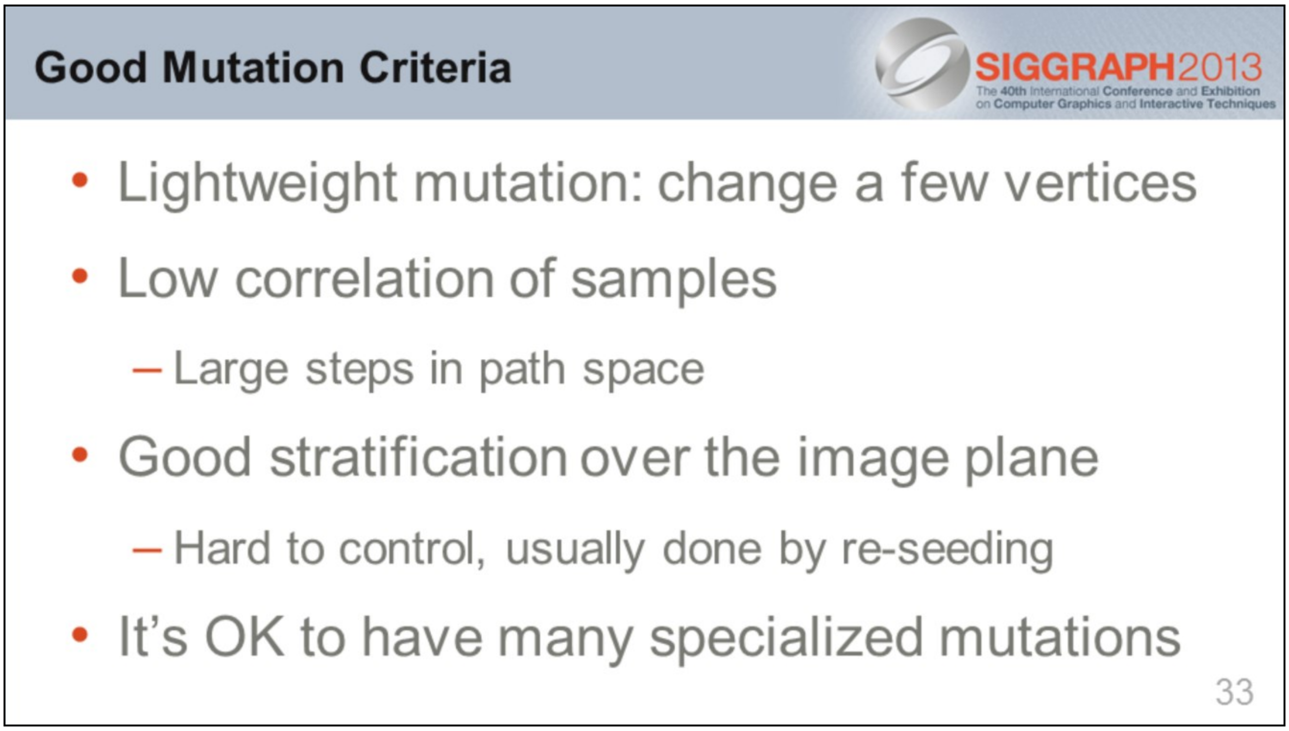
As for the start-up bias, we can seed Markov chains directly within the high-probability regions of the target distribution.

The usual practice is to collect many samples from BDPT and then seed Markov chain with one importance-sampled w.r.t. the path contribution. In case of many chains, their initial states can be also stratified with respect to the path contribution to have a good initial coverage of the path space.

And it also scales naturally well with tens of thousands of Markov chains for massively parallel devices like GPUs.

Mutation Strategies and Their Properties

Let’s try to understand how to mutate the paths.



Good Mutation Criteria

SIGGRAPH2013
The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- Lightweight mutation: change a few vertices
- Low correlation of samples
  - Large steps in path space
- Good stratification over the image plane
  - Hard to control, usually done by re-seeding
- It's OK to have many specialized mutations

33

First of all, it is important to understand what criteria should an ideal mutation strategy fulfill.

So, the mutation should be as lightweight as possible, that is, it should try to introduce minimal changes to the path, triggering as few vertex updates as possible.

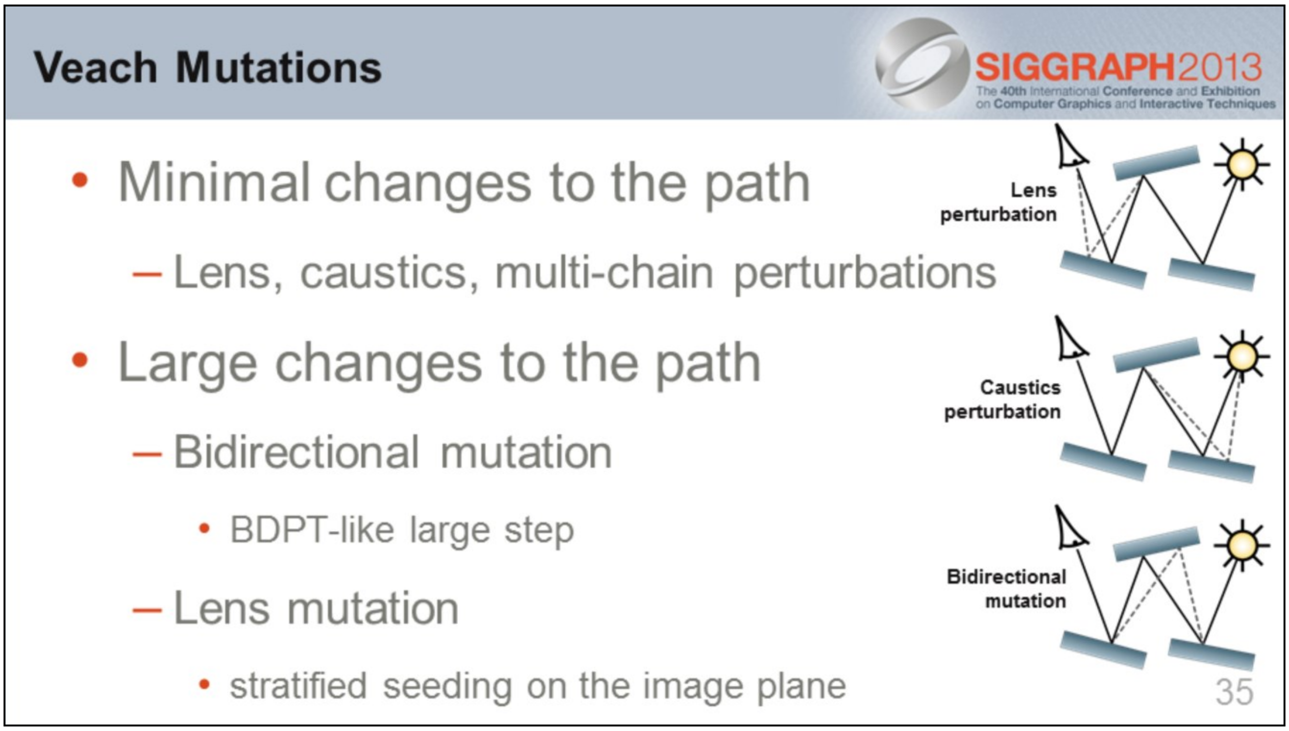
Then it should also produce a sequence of samples with low correlation, doing large steps in path space.

Also, specific to the image rendering process, the mutation should try to sample the image plane as uniform as possible. That is usually hard to control in the context of MCMC, thus the best practice is to reseed the chain with paths stratified over the image plane.

And finally, it is completely fine if the mutation can efficiently explore only some certain subset of path space, for example only caustics, leaving other features to other specialized mutations. In the end, that is one of the advantages of MLT.

Existing Mutation Strategies

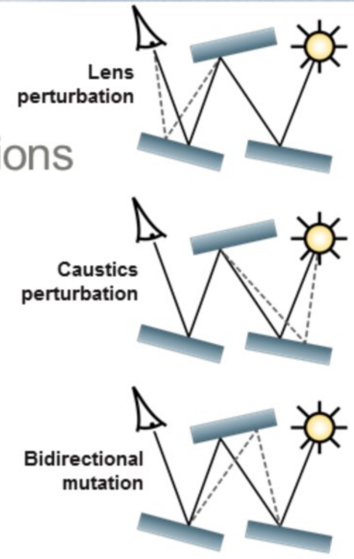
Now I’ll do an overview of the existing mutation strategies.



Veach Mutations

SIGGRAPH2013
The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- Minimal changes to the path
  - Lens, caustics, multi-chain perturbations
- Large changes to the path
  - Bidirectional mutation
    - BDPT-like large step
  - Lens mutation
    - stratified seeding on the image plane



35

Eric Veach has first introduced MLT and proposed the original set of mutation strategies.

We can roughly classify them into two groups.

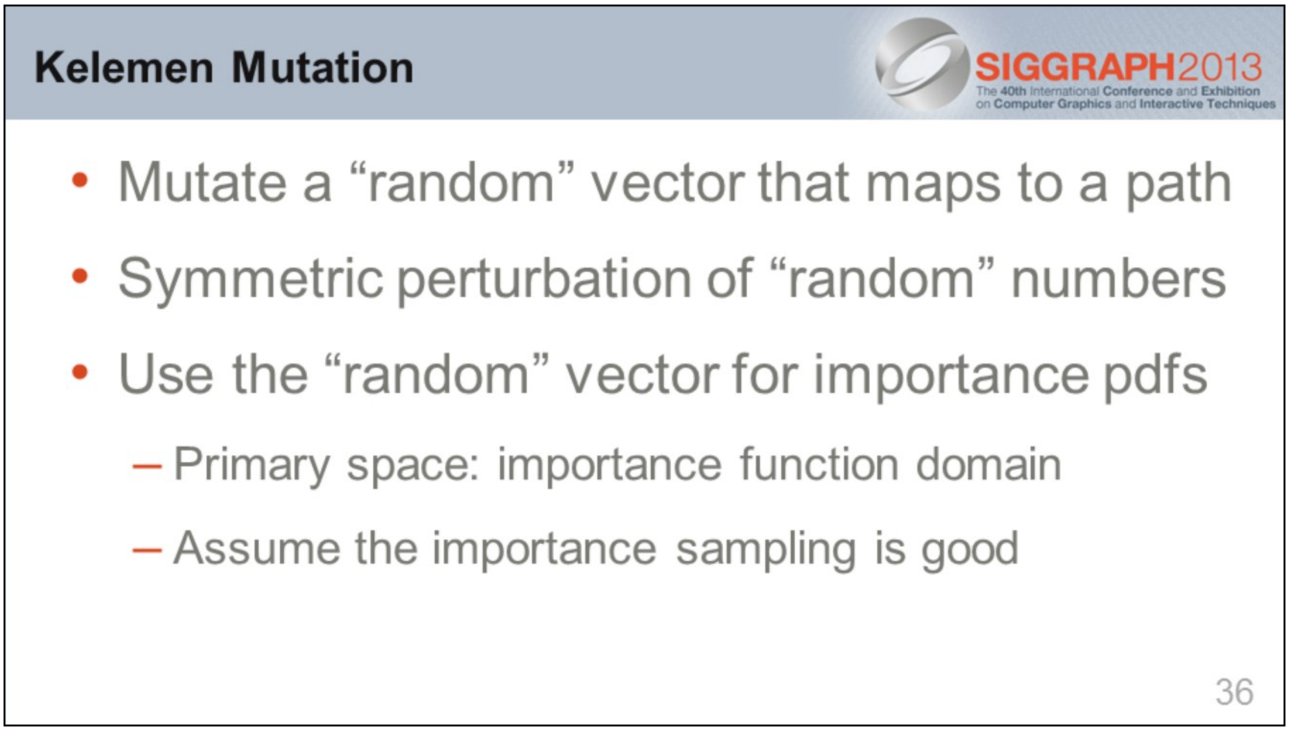
The first group perturbs the current path slightly, thus such mutations are called perturbations.

They are mostly crafted to efficiently explore the image plane and such difficult effects as caustics and chains of them.

Another group of mutations tries to do large changes to the path.

Namely, bidirectional mutation works similarly to BDPT, with the only difference that it completely resamples not the full path, but a randomly selected subpath of the current path.

Lens mutation reseeds the chain with a path from the pool of paths stratified over the image plane.


Kelemen Mutation

SIGGRAPH2013 The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

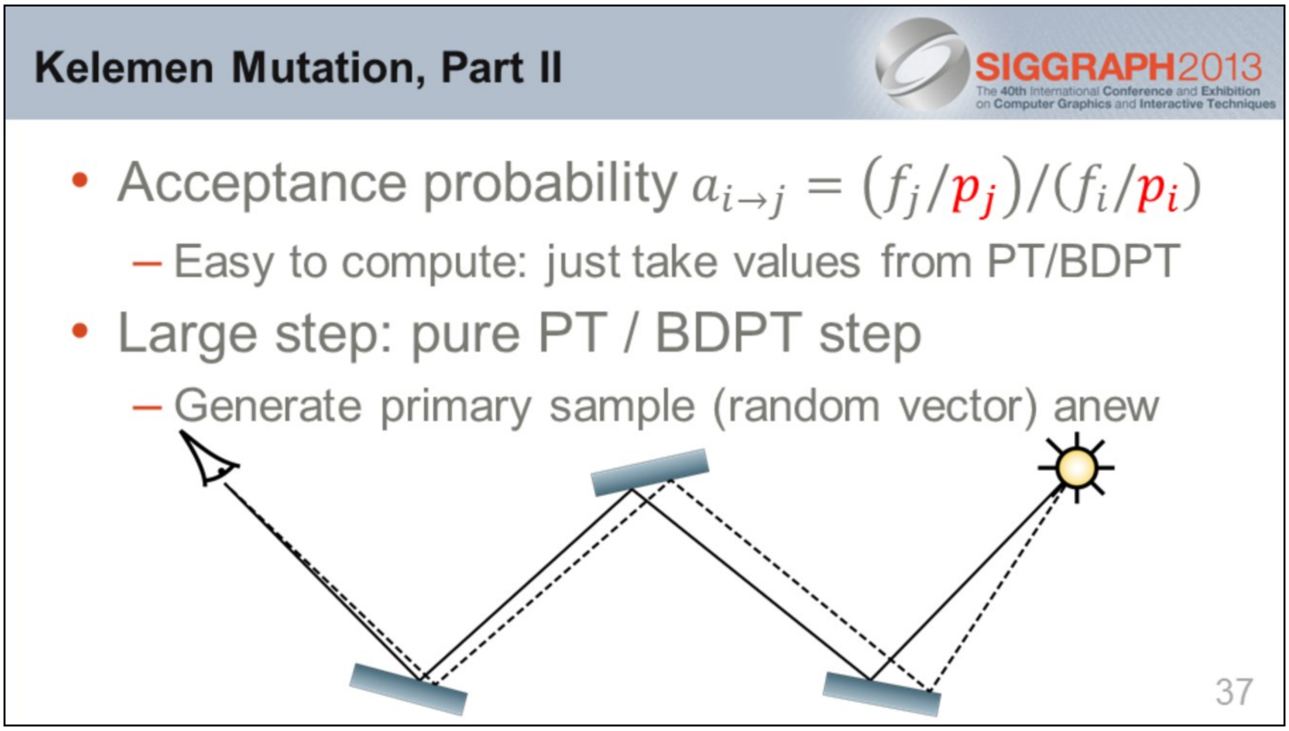
- Mutate a “random” vector that maps to a path
- Symmetric perturbation of “random” numbers
- Use the “random” vector for importance pdfs
  - Primary space: importance function domain
  - Assume the importance sampling is good

36

A popular mutation proposed by Kelemen is to mutate the paths in the so-called primary sample space, that is, the original space of the importance functions used for constructing the path in BDPT and PT.

Usually it is represented as a vector of random numbers in the unit hypercube, which is perturbed using some symmetric probability, like a multidimensional Gaussian distribution.

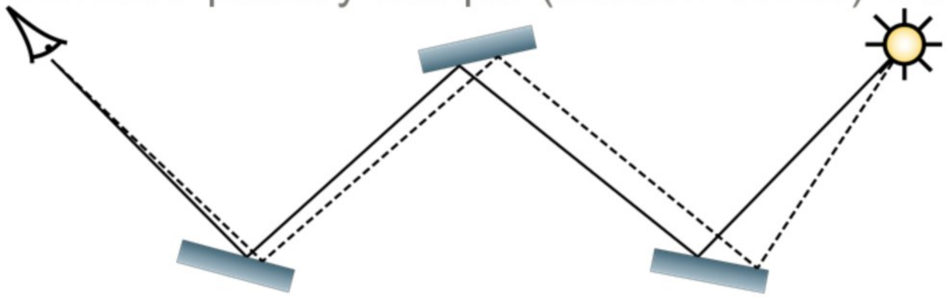
The major assumption is that the importance sampling functions already make the integrand flat enough that we can walk it using some uniform random walk in this primary space.



Kelemen Mutation, Part II

SIGGRAPH2013 The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- Acceptance probability $a_{i \to j} = (f_j / p_j) / (f_i / p_i)$
  - Easy to compute: just take values from PT/BDPT
- Large step: pure PT / BDPT step
  - Generate primary sample (random vector) anew

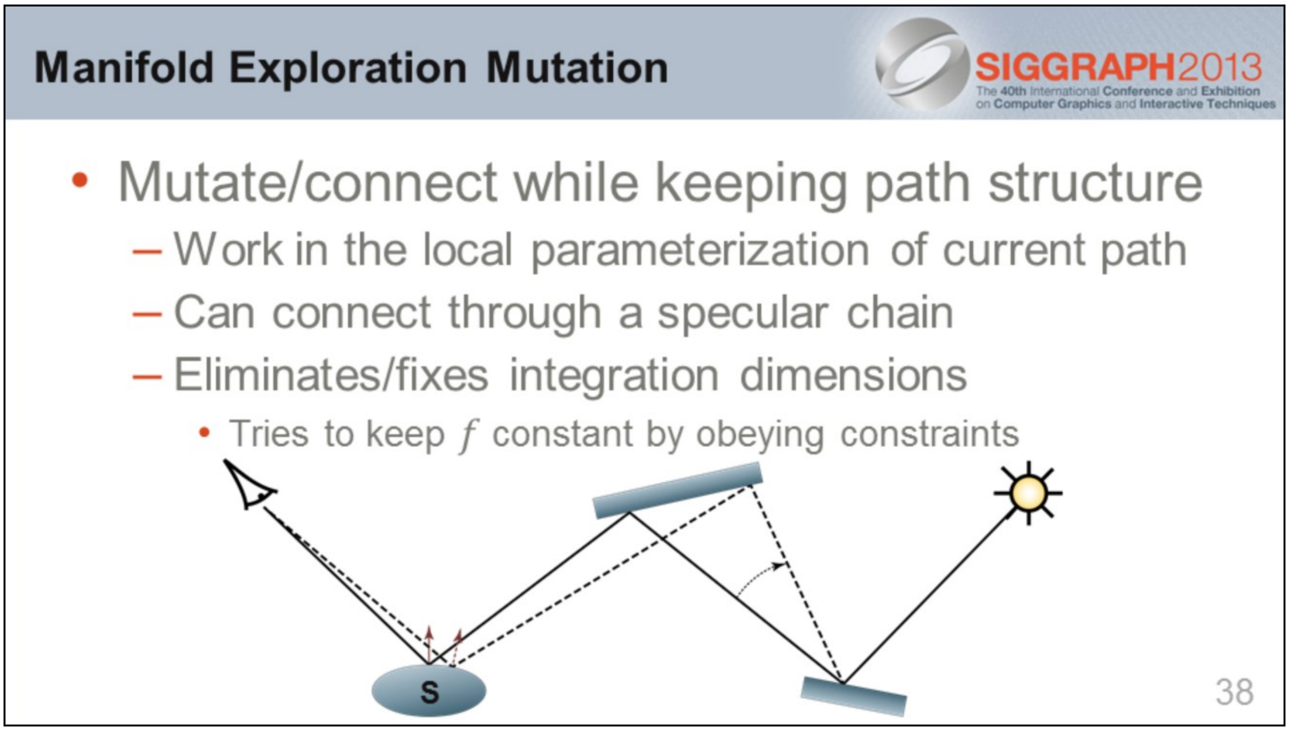


37

The good thing about this strategy is that a lot of terms in the ratio of measurement contribution to the transition probability (that is required to compute the acceptance probability) just cancel out.

Thus, since the perturbation probability is also symmetric, the final acceptance probability is computed as a ratio of the simple path throughputs computed by PT or BDPT. *[This makes it very simple to implement such a mutation strategy: just take an existing PT or BDPT, replace the random number generator by a replayeable sampler with symmetric perturbation and use the ratio of throughputs as an acceptance probability.]*

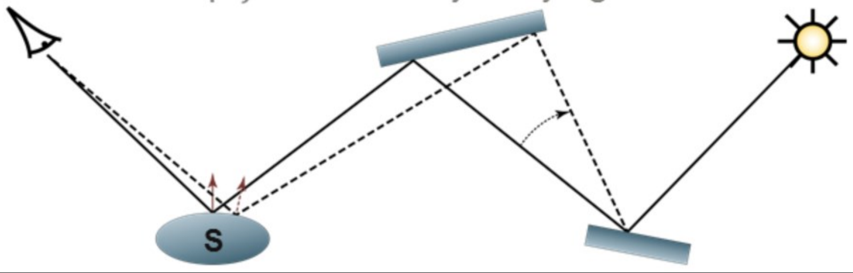
In order to discover new features quicker, we also need to do some large steps. For this reason a large step mutation was proposed. The idea here is also simple: just regenerate the complete random vector from scratch and try to construct a path. That is equivalent to just generating a random path with PT / BDPT.



Manifold Exploration Mutation

SIGGRAPH2013 The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- Mutate/connect while keeping path structure
  - Work in the local parameterization of current path
  - Can connect through a specular chain
  - Eliminates/fixes integration dimensions
    - Tries to keep f constant by obeying constraints



38

Yet another recent mutation strategy that was introduced by Wenzel Jakob is called manifold exploration.

This is a supplementary mutation strategy, which is meant to replace the set of Veach’s perturbations.

The idea here is that the path is perturbed from some vertex and then in order to construct the new subpath, first we construct a local on-surface parameterization of the current path and try to iteratively construct the new path in the space of this local tangent frame parameterization.

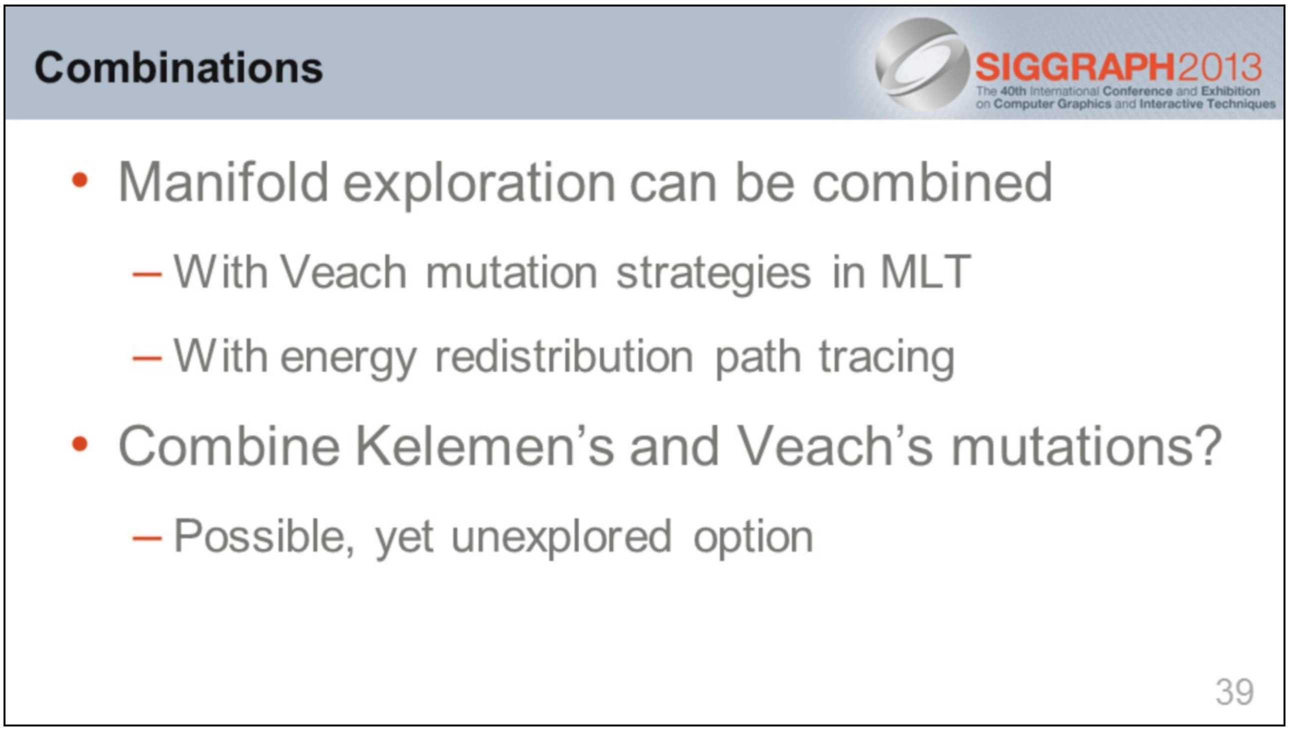
The idea comes from the differential geometry. This mutation tries to preserve hard constraints, like specular reflections, by utilizing the local knowledge about the geometry around the current path.

This way, manifold exploration can, for example, construct a connection from one point to another through a chain of specular or highly-glossy interactions.

In fact, this strategy tries to “lock in”/eliminate some of the integration dimensions (with specular/glossy interactions), while sampling others.

As a consequence, it tries to keep the measurement contribution function as constant as possible by locking or just slightly changing the terms of the measurement contribution function corresponding to the locked dimensions.

This strategy is similar to Gibbs sampling known from statistical MCMC.



Combinations

SIGGRAPH2013
The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- Manifold exploration can be combined
  - With Veach mutation strategies in MLT
  - With energy redistribution path tracing
- Combine Kelemen's and Veach's mutations?
  - Possible, yet unexplored option

39

Some strategies and methods can be combined with each other.

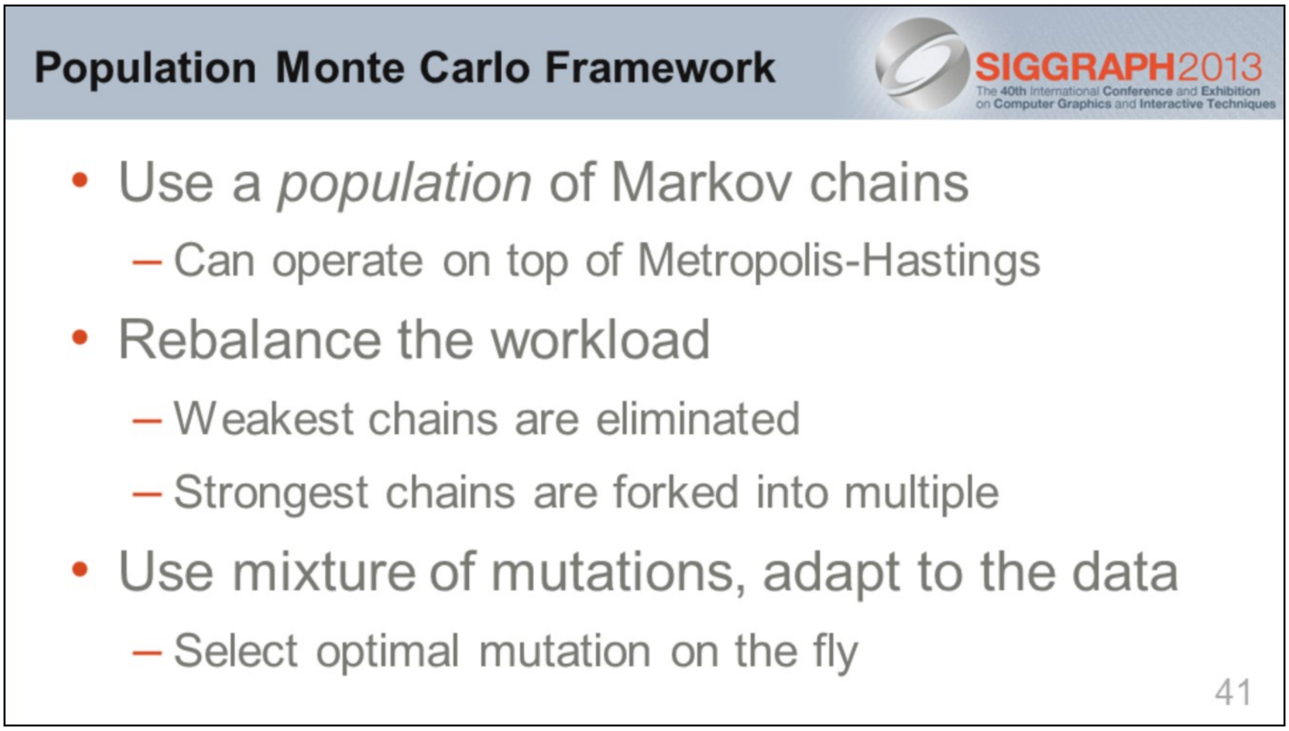
The original set of mutations can be augmented by manifold exploration.

Also the same can be done in the context of ERPT.

Moreover, another yet unexplored option is to combine the original set of mutations with Kelemen mutation.

Population Monte Carlo Light Transport

Population methods can be used on top of MLT.



Population Monte Carlo Framework

SIGGRAPH2013
The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- Use a *population* of Markov chains
  - Can operate on top of Metropolis-Hastings
- Rebalance the workload
  - Weakest chains are eliminated
  - Strongest chains are forked into multiple
- Use mixture of mutations, adapt to the data
  - Select optimal mutation on the fly

41

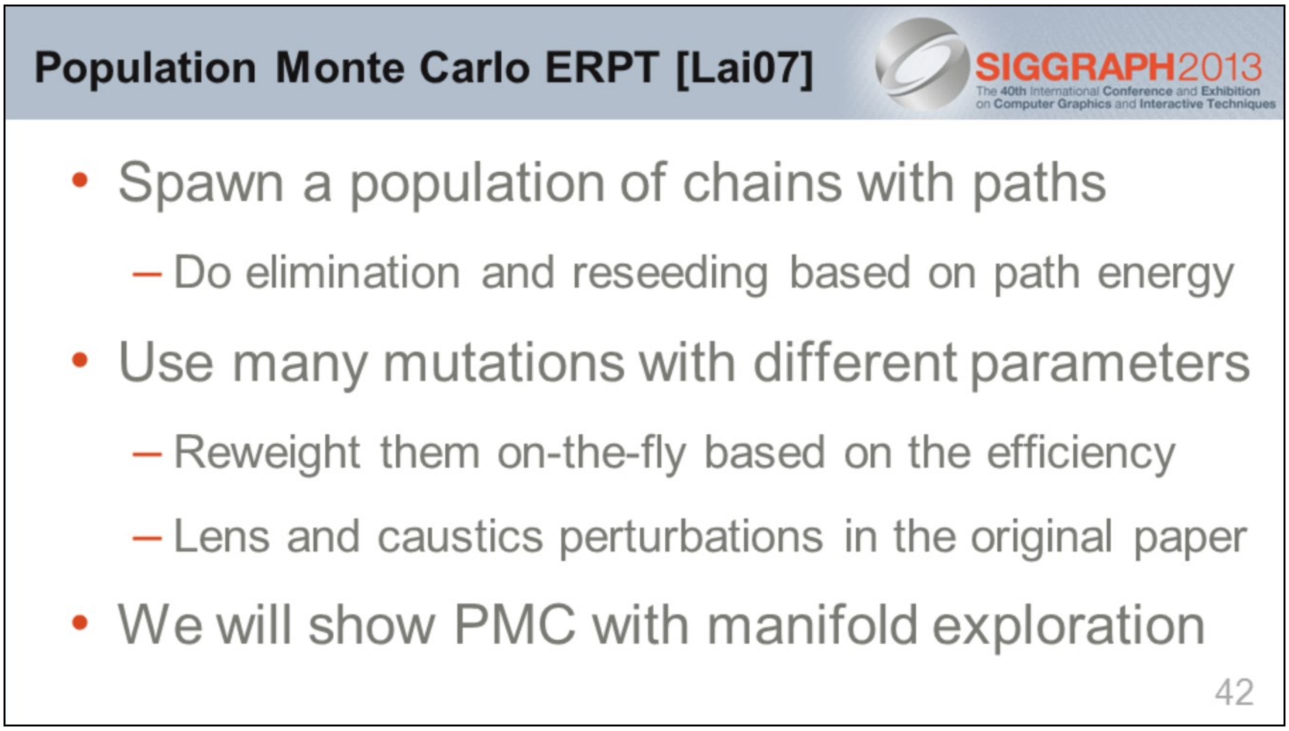
Population Monte Carlo framework stems from genetic algorithms.

Its idea is to keep a population of Markov chains (in our case it can be paths).

This method is a high-level superstructure, which can sit, for example, on top of an existing Metropolis-Hastings sampler.

Firstly, we keep only relevant samples in the population. This is done by the elimination and regeneration: the chains with a small contribution (under some threshold) are eliminated and being reseeded from the chains with very high contribution. It essentially dynamically rebalances the sampling efforts to the important places of the state space.

Moreover, we can adopt the mutation parameters (like a step size) based on the past samples and the state of the whole population on the fly.



Population Monte Carlo ERPT [Lai07]

SIGGRAPH2013
The 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques

- Spawn a population of chains with paths
  - Do elimination and reseeding based on path energy
- Use many mutations with different parameters
  - Reweight them on-the-fly based on the efficiency
  - Lens and caustics perturbations in the original paper
- We will show PMC with manifold exploration

42

Population Monte Carlo framework was applied to light transport by Lai et al. in the context of ERPT.

The process is similar to ERPT, yet it keeps the constant population of chains by reseeding the chains with low contribution from a pool of stratified paths.

The core idea is to use a set of existing mutation strategies, where each strategy can be present multiple times with different user-defined parameters, like step size. For example, the set might contain three caustics perturbation with different perturbation sizes and so on. The selection weights are then adjusted for these mutations on the fly based on the performance of each mutation. This process quickly emphasizes mutations with good performance, making the transition probability adapting to the data.

In the original paper, the authors propose to use caustics and lens perturbations.

However, in the second part of the course, we will demonstrate this method with multiple manifold exploration mutations with different perturbation parameters.

Thank You for Your attention.

Part one questions?