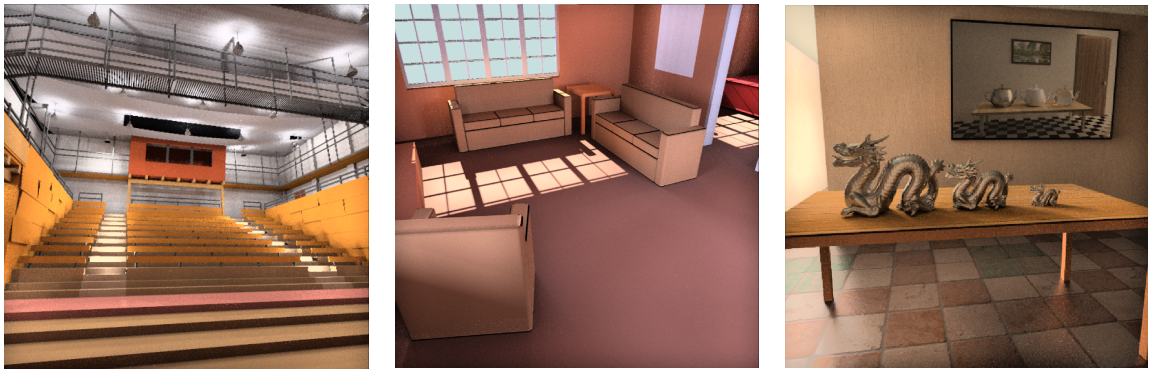
Coherent Metropolis Light Transport with Multiple-Try Mutations

B. Segovia [1,2], J.C. Iehl[2] and B. Péroche [2]

[1]ENTPE: Ecole Nationale des Travaux Publics de l'Etat, Vaulx-en-Velin
[2]LIRIS: CNRS, INSA de Lyon, Université Lyon 1, Université Lyon 2, Ecole Centrale de Lyon



(a) Candelstick Theater (b) Cabin Lit by a Simple Sky (c) The MLT Room

Figure 1: *Three* 1024×1024 *pictures rendered in* 10 *minutes with Coherent Metropolis Light Transport and about* 1000 *mutations per pixel. We accelerate the Metropolis Light Transport algorithm by adding a new class of Multiple-Try coherent mutations. With these mutations, we are now able to use ray packets or ray packet frustums to speed up the intersections and the BRDF evaluations.*

Abstract

We present in this paper an effective way to implement coherent versions of Metropolis Light Transport (MLT) by using a class of Multiple-Try mutation strategies. Indeed, even if MLT is an unconditionally robust rendering technique which can handle any kind of lighting configurations, it does not exploit any computation coherency. For example, it is difficult to cluster similar light rays into beams or cones, to perform SIMD computations on vectorized data or to efficiently use geometry caching with non-tessellated scenes. To make Metropolis Light Transport suitable to most of the currently existing commercial renderers, we therefore propose to divide the algorithm into two parts: the first one explores the entire integration space in a way very similar to the initial implementation of Metropolis Light Transport while the second one "splits" in an unbiased way each sample into a family of arbitrarily coherent samples. We finally propose to illustrate the efficiency of our approach with an example of implementation of coherent ray tracing using SIMD instructions.

1. Introduction

Simulating most of lighting phenomenas in general environments is a difficult problem. To achieve such a result, "global illumination algorithms" based on Monte-Carlo quadrature, which propose to simulate the response of a given sensor to the incoming light flux by directly sampling light paths, are certainly the most successful solutions. In this article, we present a new Monte-Carlo sampling strategy, Coherent Metropolis Light Transport (*CMLT*), which consists in sampling the integration domain proportionally to the integrand.

Conversely to the previous implementations of Metropolis Light Transport, our method proposes to reorganize the computations to make them more efficient (see Figure 3). Our goal is both to maintain the very good theoretical and numerical properties of the original algorithm and to accelerate the computation speed by adapting the *MLT* sampler to the recent advances achieved in ray tracing literature. To be more precise, we will present a new class of mutations which will allow us to combine the Metropolis Light Transport Algorithm and the use of ray packets or ray packet frustums. This better control of the ray coherency will lead to a much faster rendering technique.

The remainder of the paper is organized as follows. Section 2 presents related work and Section 3, an overview of our contribution. Section 4 reminds how the global illumination problem can be formalized as an integration problem suitable for Monte-Carlo integration and details the intrinsic limitations of Metropolis Light Transport. Section 5 exposes in details the new sampling and mutation strategies we set up to compute global illumination by combining *MLT* and ray packets. Section 6 gives all the necessary details to achieve the implementation of our technique. In Section 7, we present the results we obtained and some comparisons with other related approaches. The limitations and future work are finally given in Section 8 and Section 9 concludes.

2. Related Work

The rendering equation first introduced by Kajiya [Kaj86] is the basis of almost all rendering techniques. To solve this equation, the most efficient and versatile techniques are Monte-Carlo approaches: they are robust regardless of the scene layout and the lighting configuration and are able to compute unbiased solutions. The first Monte-Carlo global illumination algorithm, the path tracing approach proposed by Kajiya, sampled the light reaching the image plane by casting light paths backwards from the eye point. Unfortunately, as all Monte-Carlo numerical schemes, this technique suffers from limitations due to variance problems. To tackle this issue, it was therefore widely extended: the light tracing algorithm [DLW93] builds paths starting from a light source instead of the camera. Bidirectional path tracing, introduced by Lafortune and Willems [LW93] then Veach and Guibas [VG94], proposes to sample two independent paths, one generated from the camera, the other one from a light source. However, despite their robustness, most Monte-Carlo samplers are often inefficient since they do not have enough global context to quickly find all the relevant light transport paths.

To solve these issues, Veach and Guibas developed an innovative numerical algorithm, Metropolis Light Transport (*MLT*) [VG97]. Contrary to independent Monte-Carlo estimators, a Metropolis sampler is able to exploit coherency in path space and therefore to preserve the sampling context. Since 1997, several researchers have been improving and exploring the Metropolis-Hastings algorithm and its application to computer graphics. Pauly et al. [PKK00] extended it by adding new Monte-Carlo Markov Chain (*MCMC*) mutations that are able to handle participating media. Kelemen et al. [KSKAC02] simplified the implementation of *MLT* and increased the acceptance rate by making it working in the space of uniform random numbers. More recently, Cline et al. developed an efficient algorithm that combines Metropolis mutation strategies with a conventional path tracer [CTE05]. They first generate a set of path samples from the camera to the light sources, and then use a sequence of *MCMC* mutations to redistribute in an unbiased way the power of each path over the image plane. Other research works also focused on the theoretical and statistical properties of the algorithm: Szirmay-Kalos et al. analyzed the start-up bias problem of *MLT* [LPP99] while Ashikhmin et al. characterized its variance [APSS01].

Unfortunately, *MLT* and its derivatives remain slow since it seems very difficult to exploit the computation coherency or the current processor architectures and their extended instruction sets. (see Section 4 for more details about advantages and drawbacks of Metropolis Light Transport). Nevertheless, the ray tracing literature has recently known large and decisive advances . Wald et al. developed in 1999 a vectorized ray tracing implementation using *SSE SIMD* instructions [WBWS01]. Since his first implementation, Wald applied the coherent ray tracing algorithm to a large number of applications like interactive global illumination or interactive photon mapping [Wal04]. More recently, Reshetov et al. proposed a conservative extension of ray tracing, the Multi-Level Ray Tracing Algorithm (*MLRTA*) [RSH05]: by clustering rays into pyramids, they managed to achieve real time rendering with complex scenes on commodity computers. In 2006, other improvements were done and interactive or real time frame rates were also achieved with dynamic scenes [WBS06, WK06, GFW*06]. All these strategies nevertheless require that rays are coherently grouped into beams or pyramids or more generally that they are sufficiently coherent. This motivates our approach. We want to bring new numerical tools to enhance and speed up *MLT* and make it suitable to the recent advances made in ray tracing. Our second goal is to propose a new organization of the method such that *MLT* may be implemented in commercial renderers. Indeed, Christensen showed that ray tracing can be used to render very complex scenes if ray coherency (which can be tracked by ray differentials) is ensured [CLF*03, Chr06].

3. Overview of our algorithm

All the above problems therefore motivated our approach: by making *MLT* coherent, we want to produce a first step towards its implementation in production renderers. Algorithm 1 sums up our technique: it consists in extending the original implementation of Metropolis Light Transport [VG97] by adding new sequences of Multiple-Try mutations which

can be easily handled by many efficient ray tracing methods. As shown in Figure 2, we divide the method into two parts to ensure the computation coherency. First, we explore the entire integration domain by generating a set of standard light path samples with Metropolis Light Transport algorithm. Then, for each *MLT* sample, we generate a family of Multiple-Try candidates. This second part therefore creates many candidates at once around the original *MLT* sampled path and computes the intersections and *BRDF* evaluations with ray packets or ray packet frustums. With this approach, the variance of the estimators may be slightly increased but its efficiency is considerably improved.

To sum up our method, we propose to amortize the incoherent Metropolis Light Transport sampling process by using sequences of coherent, fast and parallel Multiple-Try mutations for each sample obtained with *MLT*.

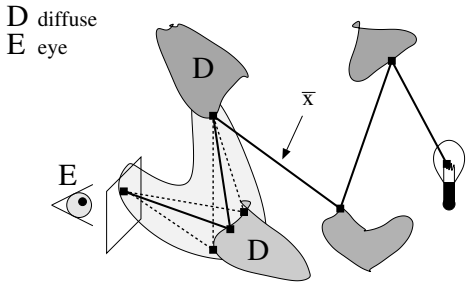


Figure 2: *Coherent Metropolis Light Transport. Here, we have mutated a part of path $\overline{x}$ given by a MLT algorithm. By using a class of Multiple-Try mutations, we generate at once, several sub-path candidates from the camera (represented by dashed lines): these mutations allow us to cluster rays into ray packets, to factorize cache accesses and so on.*

4. Solving the Light Transport Problem with Metropolis Light Transport

We present in this section a short overview of Monte-Carlo integration, reintroduce the appropriate formalism to the general light transport problem and finally detail Metropolis Light Transport algorithm since it remains the core of our new algorithm; this introduction will make the remainder of the paper easier to understand. This formalism and the notations will be used further to present our strategy in Sections 5 and 6.

4.1. Monte-Carlo Integration

The purpose of Monte Carlo integration is to compute an integral of the form:

$$I = \int_{\Omega} f(\omega)\, d\mu(\omega) \tag{1}$$

where Ω is the integration domain, f is a real valued function and μ is a measure on Ω. I is thus the mean of function f on Ω for the given measure μ.

A Monte-Carlo integrator simply consists in sampling one or several random variable families and evaluating the integrand. Let $Y = (Y_n)_{n\in N}$ denote a sequence of random variables. Under some specific conditions, Monte-Carlo integration can compute the integral by reexpressing it as the expected value of $(Y_n)_n$: $I = E(Y)$. A common way to generate a suitable random variable family (which is for example used in common path tracing algorithms) is to consider a sequence of independent random variables $X = (X_n)$ with the same probability density function p defined on (Ω, μ) and to use the weak law of large numbers such that: $I = E(Y) = E\left(\frac{f(X)}{p(X)}\right)$.

Algorithm 1 Coherent Metropolis Light Transport with main chain length n and sub-chain length m

1: Set all pixel intensities to 0
2: Compute the power P_c received by the camera and choose a first path sample $\overline{x}_1$
3: **for** $i = 1$ to n **do**
4: $\quad \overline{y}_1 \leftarrow \overline{x}_i$
5: $\quad$ **for** $j = 1$ to m **do**
6: $\quad\quad$ With a Multiple-Try mutation, "split" the current sample $\overline{y}_j$ into several candidates. Use ray packets or ray packet frustums to compute ray intersections or *BRDF* evaluations. Accumulate all the multiple-try candidates proportionnaly to the power they bring to the camera and the generalized Metropolis ratio used to generate them. {see Section 5 for more details}
7: $\quad\quad$ Choose the next sample $\overline{y}_{j+1}$.
8: $\quad$ **end for**
9: $\quad$ Generate the next candidate $\overline{x}_{i+1}$ with a Metropolis-Hastings mutation.
10: **end for**
11: Scale the pixel intensities such that the power received by the camera becomes P_c

4.2. Path Integral Formulation

As a Monte-Carlo integrator requires to formalize the problem as an integration one, Veach proposed in his PhD thesis [Vea97] to rewrite the light transport problem.

The Light Transport Equation
Veach first developed the "three point form" formulation which describes the local lighting behavior of materials:

$$L(x'\to x'') = L_e(x'\to x'') + \int_{\mathcal{M}} L(x\to x') f_s(x\to x'\to x'') G(x\leftrightarrow x') dA(x) \tag{2}$$

L is the equilibrium outgoing radiance function, $L_e(x' \to x'')$ is the emitted radiance leaving x' in the direction of x'', and $G(x \leftrightarrow x')$ is the geometric term between x and x'. It represents the differential beam between the two differential surfaces and is given by: $G(x \leftrightarrow x') = V(x \leftrightarrow x') \frac{\cos(\theta_0)\cos(\theta_i')}{||x-x'||^2}$ where $V(x \leftrightarrow x')$ is the visibility term between x and x' which is equal to 1 if x sees x' and zero otherwise. θ_0 (resp. θ_i') is the angle between $x \to x'$ and the surface normal at x (resp. x'). $f_s(x \to x' \to x'')$ is the bidirectional scattering distribution function of the material. $\mathcal{M}$ is finally the union of all scene surfaces and A is the Lebesgue (i.e. uniform) measure on $\mathcal{M}$.

Since formalizing the light transport problem as an integral equation is however not suitable to Monte-Carlo integration, we have to reexpress it.

The Measurement Equation

Any pixel computation can be first defined as the response of a hypothetical sensor to the incident radiance field. If $W_e^{(j)}(x \to x')$ is the responsivity of sensor j and I_j, the power it receives, we can define the measurement equation by:

$$I_j = \int_{\mathcal{M}\times\mathcal{M}} W_e^{(j)}(x \to x') L(x \to x')\, G(x \leftrightarrow x')\, dA(x)\, dA(x') \quad (3)$$

The Path Integral Formulation

Using the light transport equation, the measurement equation can be recursively expanded to be expressed in an iterative way:

$$I_j = \sum_{k=1}^{\infty} \int_{\mathcal{M}^{k+1}} \Big[L_e(x_k \to x_{k-1}) G(x_k \leftrightarrow x_{k-1})\, W_e^{(j)}(x_1 \to x_0) \cdot \Big(\prod_{i=1}^{k-1} f_s(x_{i+1} \to x_i \to x_{i-1}) G(x_i \leftrightarrow x_{i+1}) \Big) dA(x_0) \ldots dA(x_k) \Big] \quad (4)$$

The measurement equation can be finally reformulated as:

$$I_j = \int_{\Omega} f^{(j)}(\bar{x}) d\mu(\bar{x}) \quad (5)$$

$f^{(j)}$ is defined for each path length k by extracting the appropriate term from expansion (4), Ω is the set of all finite length paths and μ the natural associated measure given by: $\mu(D) = \sum_{k=1}^{\infty} \mu_k(D \cap \Omega_k)$ where Ω_k is the set of all length k paths and μ_k the associated product measure given by $d\mu_k(x_0 \ldots x_k) = dA(x_0) \ldots dA(x_k)$.

With this formalism, the global illumination problem is now an integration problem which can be solved by a Monte-Carlo algorithm. Furthermore, this formulation directly handles any path of any length and therefore allows us to change path length without any theoretical difficulty.

4.3. Metropolis Sampling for Light Transport

We give here a short overview of Metropolis-Hastings algorithm and its application to the global illumination problem as introduced by Veach and Guibas [VG97].

Metropolis-Hastings (MH) Algorithm

We first recall that a sequence of random variables $(X^{(t)})_{t\in N}$ is a Markov Chain if $X^{(t)}$ depends only on $X^{(t-1)}$ through a transition function $g(\cdot|x^{(t-1)})$. The goal of the Metropolis-Hastings algorithm is to construct a Markov Chain that has given a equilibrium distribution π_∞ by applying successive mutations on its elements. This algorithm does not solve *a priori* an integration problem but may provide a very elegant variance reduction technique in the case where many correlated integrals have to be computed.

The algorithm starts at $t = 0$ with the selection of $X^{(0)} = x^{(0)}$ randomly drawn from a distribution π_0 with the only requirement that $\pi_0(x^{(0)}) > 0$. Given $X^{(t)} = x^{(t)}$, the algorithms computes $X^{(t+1)}$ as follows:

1. Sample a candidate value X^* from the transition function $g(\cdot|x^{(t)})$
2. Calculate the Metropolis-Hastings ratio $R(x^{(t)}, x^*)$, where:
$$R(u,v) = \frac{\pi_\infty(v) \cdot g(u|v)}{\pi_\infty(u) \cdot g(v|u)}$$
3. Sample a value for $X^{(t+1)}$ according to the following rule:
$$X^{(t+1)} = \begin{cases} X^* & \text{with probability } \min\{R, 1\} \\ x^{(t)} & \text{otherwise} \end{cases}$$

It is possible to show that under general conditions, the sequence $(X^{(t)})_{t\in N}$ is a Markov Chain with equilibrium distribution π_∞.

With the MH sampler, we can therefore sample almost any distribution π_∞. If we ensure the *ergodic* property of the chain (i.e. that all states are equally probable according to π_∞ after a long time passed in the chain), we are furthermore able to take *all* samples of the Markov Chain as if they exactly follow the stationary distribution. To do this, it is sufficient to ensure that $g(x|y) > 0$ when $\pi_\infty(x) > 0$ and $\pi_\infty(y) > 0$ since all states can be reached with only one mutation step with a non-null probability.

Application to Light Transport

Veach and Guibas proposed to use a MH sampler as a powerful variance reduction technique for the global illumination problem. They first evaluate the total power P_c received by the camera and then use a Metropolis sampler to compute correlated random variables with a density directly proportional to integrand $f^{(c)}$, i.e. the power transmitted by a path to the camera, as defined in equation 5. During the sampling process, they finally estimate the pixel intensities

by making a histogram of the samples taken from the sampled distribution and by finally scaling the histogram to approximate $f^{(c)}$. For a more detailed introduction to Metropolis sampling and its application to rendering, we refer to [Pha03].

Pros and Cons of Metropolis Light Transport
The decisive advantages of Metropolis Light Transport over almost all other sampling and integration techniques are:

- its theoretical elegance since it proposes a unified solution to the Light Transport Problem;
- its numerical robustness and its insensitive behavior in relation to the scene configuration;
- its unbiasedness.

Unfortunately, two major drawbacks make it unsuitable for production renderers.

- due to the very large number of random paths generated for a given picture, flickering problems are difficult to handle when rendering an animation;
- it has very poor *algorithmic* properties. Indeed, since the samples are sequentially generated, the result of sample n is needed to compute sample $n+1$ and it therefore becomes very difficult to cluster rays. Furthermore, to make the algorithm unbiased, large changes may happen from one sample to the next one making caching strategies inefficient.

We propose in this article to tackle the second problem (we will however present some ideas to handle the first one). To achieve such a result, we will introduce in the computer graphics research field a very successful sampler recently presented in computational statistics [JSL00].

5. Coherent Metropolis Light Transport

In this section, we present the core of our contribution, i.e. the new mutation family we add to enhance the speed of *MLT*.

5.1. *MTMH* Algorithm

The main reason for the poor coherency of computations while using *MLT* is the fact that all {sampling/BRDF evaluation/accumulation} processes are sequential: the results for path n are needed to evaluate the results for path $n+1$. To break the sequential aspect of the algorithm, we propose to generate many samples at once using a Multiple-Try Metropolis Hastings Algorithm (*MTMH*).

The approach is to generate a larger number of candidates thereby improving the exploration of π_∞ near x. One of these proposals is then selected in a manner that ensures that the chain has the correct limiting stationary distribution. To achieve such a result, we still use a proposal distribution g, with optional non-negative weights $\lambda(u,v)$ where the *symmetric* function λ is presented further below. To ensure the correct limiting stationary distribution, it is necessary to require that $g(x^*|x^{(t)})>0$ if and only if $g(x^{(t)}|x^*)>0$, and that $\lambda(x^{(t)},x^*)>0$ whenever $g(x^*|x^{(t)})>0$. Let $x^{(0)}$ denote the starting value, and define $w(u,v)=\pi_\infty(v)g(u|v)\lambda(u,v)$. Then, for $t\in N$, the algorithm proceeds as follows:

1. Sample p independent proposals $X_1^*\dots X_p^*$ (also called *MTMH candidates*) from $g(\cdot|x^{(t)})$
2. Randomly select a single proposal X_j^* from the set of proposals , with probability proportional to $w(x^{(t)},X_j^*)$ for $j=1,\dots,p$
3. Given $X_j^*=x_j^*$, sample $p-1$ independent random variables $X_1^{**},\dots X_{p-1}^{**}$ (also called *MTMH competitors*) from the proposal density $g(\cdot|x_j^*)$. Set $X_p^{**}=x^{(t)}$
4. Compute the generalized Metropolis-Hastings ratio:
$$R_g=\frac{\sum_{k=1}^{p} w(x^{(t)},X_k^*)}{\sum_{k=1}^{p} w(X_j^*,X_k^{**})}$$
5. Set
$$X^{(t+1)}=\begin{cases} X_j^* & \text{with probability } \min\{R_g,1\} \\ x^{(t)} & \text{otherwise}\end{cases}$$

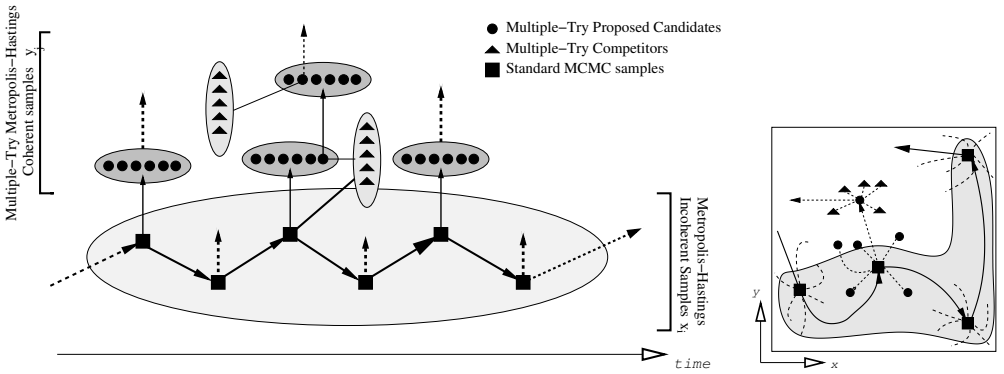
Intuitively, instead of testing two samples x and x^*, and keeping only one of them with the respective probabilities 1-min$(1,R)$ and min$(1,R)$ as done with a standard Metropolis-Hastings sampler, *MTMH* proposes to test two families of samples, $(x_1^*\dots x_p^*)$ and $(x_1^{**}\dots x_p^{**})$, and to keep only one element chosen from each family x_j^* or $x_i=x_p^{**}$ with the respective probabilities 1-min$(1,R_g)$ and min$(1,R_g)$.

Figure 3 presents the two algorithms, *MH* and *MTMH*, as they are used in our new implementation of *MLT*. As we can see, to determine if the proposed candidate is accepted or rejected, the *MTMH* competitors represented by triangles are tested against the *MTMH* candidates represented by disks.

5.2. Application to Metropolis Light Transport

MTMH can be applied in very different ways. The first obvious implementation would be to replace all *MH* mutations by *MTMH* mutations. Unfortunately, this technique would provide very poor results: if you generate for example, 10 samples for each *MTMH* mutation, you will accumulate only one candidate among the 10 ones, therefore resulting in a very poor efficiency. Furthermore, if the candidates are independently generated, the computation coherency can not be ensured since bidirectional mutations (see [VG97] for all necessary details about this mutation strategy) can explore very different parts of the sample space. What we propose here is therefore a bit different.

Unbiased Exploration of the Sample Space
We first generate a path sample family with only bidirectional mutations by using standard *MH* proposals (represented by squares in Figure 3). This sampling process



(a) (b)

Figure 3: *Coherent Metropolis Light Transport. (a) presents the time representation of the algorithm while (b) shows it in screen space where each light path is projected.*

provides a sample family with a density equal to $f^{(c)}/||f^{(c)}||$ and the resulting estimator is therefore unbiased. Conversely to the original *MLT*, we do use neither the caustic perturbation strategy nor the lens one since our goal is to explore the entire integration domain in an unbiased way and not to provide efficient estimators. This first part finally provides a set of n path samples $(\overline{x}_i)_{i\in[1...n]}$

The "Lens Sub-path" Space

For a given path $\overline{x}_i$, the lens sub-path $(x_{i,1}\dots x_{i,k})$ is the sub-path of the form $ES^*DS^*(L|D)$ (where we use Heckbert's regular expression notation [Hec90]; S, D, E, and L stands for non-diffuse, diffuse, lens and light vertices respectively); the light sub-path is on the contrary, the remainder of the path (which can be void if the lens sub-path is already connected to a light source). We can give an intuitive representation of the lens and light sub-paths: if we would want to discretize the incoming radiance field by a set of Virtual Point Lights (as done in Instant Radiosity [Kel97]), appropriate positions for these point lights would be the second diffuse surface, i.e. the ending point of the lens sub-path. This point can therefore be considered as a temporary Virtual Point Light which illuminates the first diffuse surface seen by the camera through specular reflections.

For each $\overline{x}_i$, we can therefore define lens sub-path spaces Ω_i^{ls} such as $\overline{x}$ belongs to Ω_i^{ls} if and only if the light sub-path of $\overline{x}$ is identical to the light sub-path of $\overline{x}_i$. This sub-space is quite interesting since it is much smaller and can be entirely explored by using only lens or caustics *perturbations*: once the initial path $\overline{x}$ is given, we just have to generate new sub-paths around the lens sub-path of $\overline{x}$ to sample it.

Figure 2 gives, for example, the strategy that can be used to explore a *diffuse* lens sub-path, i.e. a sub-path of the form EDD: this approach consists here to sample rays from the camera and to connect the computed intersection to the second diffuse surface of the lens sub-path of $\overline{x}$. If we are able to cluster many camera rays *before* performing any intersection or *BRDF* evaluation, we will therefore be able to factorize many common operations. As these strategies are implementation dependent, we give in Section 6 all the necessary details to implement effective perturbations to explore the lens sub-path space.

What we propose is therefore to use a $MTMH$ sampler to explore each sub-space Ω_i^{ls} and thus, to compute new unbiased estimators from the initial unbiased estimator given by the sample family $(\overline{x}_i)$. Before extensively presenting the $MTMH$ version of the lens sub-path space, we first detail a MH version and explain why it does not provide both computation coherency and low-variance estimators. We will finally show that the exploration of the lens sub-path space with $MTMH$ will give us the theoretical roots of a coherent implementation of *MLT*.

Exploring the Lens Sub-path Space with *MH*

As the samples $(\overline{x}_i)_i$ are generated proportionally to $f^{(c)}$, each of them brings an equal quantity of energy to the camera (equal to $||f^{(c)}||/n$). By counting how many path samples go through each pixel, we have, as shown by Veach and Guibas, an unbiased estimator of the power received by each pixel. From a given $\overline{x}_i$, we can then perform a MH move to generate a new path $\overline{y}_i$ with an acceptance probability equal to R_i and a transition function $q(\overline{x}_i,\overline{y}_i)$. As we use a MH move, the detailed balance is satisfied and the energy transfered from $\overline{x}_i$ to $\overline{y}_i$ is equal to the energy transfered from $\overline{y}_i$ to $\overline{x}_i$. Therefore, the estimator obtained by weighting the contributions of the two families $(\overline{x}_i)_i$ and $(\overline{y}_i)_i$ by $1-R_i$ and R_i is still unbiased.

This assumption can lead to several strategies to explore the lens sub-path space. Cline et al. propose, for example, to build a finite length Markov-Chain from each $\overline{x}_i$ and to recursively deposit the energy according to the Metropolis ratio evaluated at each step [CTE05]. This strategy is

interesting to reduce the variance of the estimators, but as the paths are sequentially generated, it seems very difficult to cluster the generated rays. Another strategy, also suggested in [CTE05], would be to split each sample into m candidates and to proportionally deposit the energy of each generated sample. This method seems to ensure the computation coherency, but it leads to high variance estimators since the energy received by the mutated samples will vary greatly. Iterating in this manner will result in an exponential growth in the number of samples. As our goal is to maintain low variance estimators *and* to exploit the computation coherency, *MTMH* moves provide the best of the two previous approaches.

Exploring the Lens Sub-path Space with *MTMH*

With *MTMH* mutations, we are able to generate p different candidates $\overline{x}_1^* \dots \overline{x}_p^*$ (represented by disks in Figure 3) for a given initial sample $\overline{x}^{(t)}$ (see Section 5.2 for the notations). Once the single proposal $\overline{x}_j^*$ has been chosen, we generate p competitor samples $(\overline{x}_1^{**} \dots \overline{x}_{p-1}^{**}, \overline{x}_p^{**} = \overline{x})$ from $\overline{x}_j^*$. Then, the process is repeated by setting as current sample, either $\overline{x}$ or $\overline{x}_j$. This approach therefore allows to generate many samples at once and to perform an efficient Markov Chain which can efficiently explore the lens sub-path space. The last problem which has to be solved is to use and accumulate the contributions of *all* the generated samples, i.e. all the candidates $\overline{x}^*$ and all the competitors $\overline{x}^{**}$. To achieve such a result, we can make a simple remark: compared to the *MH* algorithm, *MTMH* only replaces the single candidates by families of candidates and, as indicated in [JSL00], the detailed balance is therefore still maintained. This leads to the following strategy; for each step of the *MTMH* mutations, we build $2p$ estimators:

- The first p ones are $\overline{x}_1^* \dots \overline{x}_p^*$. Their contributions are respectively weighted by $R_g \cdot w_i^*$ where $w_i^* = f^{(c)}(\overline{x}_i^*) / \sum_{k=1}^{p} f^{(c)}(\overline{x}_k^*)$
- The last p ones are $\overline{x}_1^{**} \dots \overline{x}_p^{**}$. Their contributions are respectively weighted by $(1 - R_g) \cdot w_i^{**}$ where $w_i^{**} = f^{(c)}(\overline{x}_i^{**}) / \sum_{k=1}^{p} f^{(c)}(\overline{x}_k^{**})$

This approach can be intuitively explained: as we want to maintain the detailed balance, we first accumulate each family proportionally to "its Metropolis ratio", then, we accumulate each element *inside* a given family proportionally to its energy function, i.e. $f^{(c)}(\overline{x})$.

User Parameter Values

We set $\lambda(u,v) = [g(u|v) \cdot g(v|u)]^{-1}$ to encourage certain types of proposals: by using this specific λ, $w(x_i, x^*)$ corresponds to the importance weight $\pi_\infty(x^*)/g(x^*|x_i)$. However, we noticed that the algorithm was not particularly sensitive to the value of λ.

For the Gaussian standard deviations, we finally found that appropriate values were close to the parameters proposed by Veach and Guibas in [VG97]. For example, for the radius R of the *MTMH* perturbations in screen space, setting σ between 5% and 10% of the image size (width or height) provides good results. However, they strongly depends on the number p of accumulated candidates and competitors. For more details about the numerical behavior of the strategies, please refer to Section 7.2.

5.3. Summary of the Approach

The roots of Coherent Metropolis Light Transport are therefore *MTMH* mutations. On the one hand, we explore the entire sample space with *MH* bidirectional mutations. On the other hand, for each *MH* path, we sample the corresponding lens sub-path space in a fast manner by performing a finite length sequence of *MTMH perturbations*. Compared to the standard Metropolis Light Transport, we may therefore notice two deep changes:

- the perturbations and the bidirectional mutations are reorganized. Instead of sequentially alternating {lens|caustics} perturbations and bidirectional mutations, we deterministically apply a sequence of perturbations for each bidirectional mutation;
- all *MH* perturbations are replaced by *MTMH* perturbations.

We now have to analyze a last point: how can we mutate the lens sub-paths in a fast and coherent way?

6. Implementing Coherent Metropolis Light Transport

The previous sections did not detail the implementation of our technique. We therefore present here how we perform the *MTMH* mutations to explore the lens sub-path space and how we can cluster rays to speed up the computations.

6.1. Designing an Effective Lens Sub-space Exploration

The first step is to build an effective mutation strategy to generate the *MTMH* candidates and competitors. Before detailing our method, we can make two remarks which motivated our choice:

- paths $\overline{x}$ generated by standard *MLT* (represented by squares in Figure 3) are already an effective sample set of density $f^{(c)}/||f^{(c)}||$: that motivates to explore the close neighborhood of each sample;
- the mutation strategy to generate *MTMH* mutations must not introduce extra sampling artifacts or patterns which may be slow to disappear.

These two points lead us to use Gaussian random variables with density $q_{\mu,\sigma}$ defined by $q_{\mu,\sigma} = \frac{1}{2\sigma\pi} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$. Indeed, depending on the nature of the given sub-path, we have to use two different path generation strategies:

- let assume that the lens sub-path $x_0 \dots x_k$ has the form

$ES^*DS^+(D|L)$) (it is a caustic lens sub-path): we generate a path starting from x_k, the second diffuse surface (or the light source). The direction of the segment x_k, x_{k-1} is perturbed by a random (θ, ϕ) where ϕ is a uniform random variable and θ is a Gaussian random variable with $\mu = 0$ and σ is user-defined;
- otherwise, the lens sub-path $x_0 \ldots x_k$ has the form $ES^*D(D|L)$ (it is a non-caustic lens sub-path): we perturb the old image location by moving it a Gaussian random distance R and a uniform angle ϕ.

For these two kinds of mutations, we noticed that the Gaussian strategies were the most effective. Since we want to use a large number of *MTMH* candidates per *MH* sample, a uniform density for R or θ gives very poor results. After implementing it, we noticed that bright circles which were slow to disappear may appear around duplicated *MH* samples. Other strategies were tested like linear densities, but none of them gives satisfactory results. Conversely, since all derivatives of Gaussian distributions are null at μ, the Gaussian *MTMH* perturbations offer very smooth transitions and the results we got are very similar to the ones obtained with a standard *MLT* (see Section 7).

We finally have to make an important remark concerning Gaussian densities: in our case, we do not use exact Gaussian densities, but clamped ones which are defined on a given domain Ω by:

$$q_{clamped}{}_{\mu,\sigma} = \frac{q_{\mu,\sigma}}{\int_\Omega q_{\mu,\sigma}(\omega) d\omega}$$

In the particular case where Ω is the set of screen coordinates, $q_{clamped}{}_{\mu,\sigma}$ is different for each pixel. We first precompute and store on the hard disk a Gaussian map which gives for every pixel the value of $1 \,/\, \int_\Omega q_{\mu,\sigma}(\omega) d\omega$. This allows us to sample clamped Gaussian random variables with a rejection technique and no extra computations at run-time.

6.2. Making the Computations Coherent

As described before, the *MH* part of the algorithm (represented by the Markov Chain of squares in Figure 3) is incoherent and we do not intend to speed this part of the algorithm. On the contrary, we want to amortize the expensive cost of each *MH* by computing sub-sequences of *MTMH* lens sub-path samples. Several implementations are possible.

Coherent *MLT* with *SIMD* Ray Packets

In this section, we will suppose that the lens sub-path is non-caustic (the approach is quite similar for a caustic lens sub-path). Let us consider the given current path $\overline{x}$ we have to mutate and its corresponding lens sub-path $\overline{x}^{ls}$. To perform the perturbation of $\overline{x}^{ls}$, we first perform a $n \times m$ jittered and uniform sampling as presented in Figure 4. Then, by inverting the Gaussian density (a Box-Muller transform can, for example, be used), we generate a set of $n \times m$ ray segments that we use to trace the corresponding *MTMH* candidates. These complete sub-paths are obtained by perturbing the remainder of the segment in a way very similar to the multi-chain perturbation technique presented by Veach and Guibas in [VG97]. To achieve effective *SIMD* computations as presented by Wald in [WBWS01], we finally cluster 4 neighborhood rays as shown in Figure 4 and we perform the camera path tracing (or the light path tracing if we have to mutate a caustic lens sub-path) using a coherent ray tracer. We note that the path tracing part remains coherent since we cluster a set of close camera rays and we perturb the remainder of path around the *same* given camera path $\overline{x}^{ls}$. As described by Benthin in his

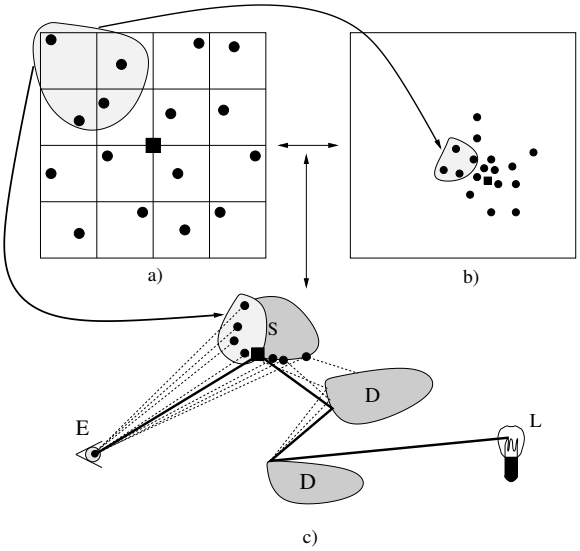


Figure 4: *Our technique to ensure ray coherency. (a) is the uniform jittered 2D sampling. (b) shows the corresponding Gaussian sampling in screen space. We can see the* 2×2 *ray packet we made with 4 neighbor rays. (c) shows some resulting camera sub-paths: we have mutated here a ESDD lens sub-path.*

PhD [Ben06], larger ray packets can also be used to speed up more effectively the intersection operations: this can also be implemented with the jittered approach and no extra theoretical difficulties.

Coherent *MLT* with Ray Packet Frustums

Very efficient algorithms were recently introduced to exploit in a better manner ray coherency by using ray packet frustums [RSH05, WBS06, WIK*06]: as the interval arithmetic supporting all these algorithms requires a common origin for all rays in the packet, we can not compute all kinds of lens sub-paths with ray packet frustums only. For *EDD* lens sub-paths, there is no specific problem, since the two ray packets have common origins (the first one is the camera while the second one is the second diffuse surface).

Nevertheless, for a *ESDSL* sub-path or other sub-path types, we can use ray packet frustums only for the rays started from the camera and the rays started from the light source: the remaining rays have to be processed by casual *SIMD* packets.

6.3. Summary and Remarks

MTMH perturbations thus allow us to generate many coherent rays at once, to perform coherent ray tracing and to use ray packet frustums when it is possible. The next section presents the results we obtained with our implementation and the tests we did to demonstrate the efficiency of our approach.

Furthermore, as indicated by Craiu and Lemieux in [CL07], the *MTMH* candidates and competitors do not even need to be independent so that the *MTMH* proposals can be directly generated with low-discrepancy number sequences and quasi Monte-Carlo techniques. This can lead to better estimators and may be really helpful to handle flickering-free rendering of animations (we give some possible ideas in Section 8).

7. Results

We implemented a complete rendering system to perform *SIMD* computations when using *MTMH* perturbations. We actually recoded the OpenRT API [DWBS03] and divided our *CMLT* renderer into two parts: the first one sequentially generates paths with a standard Metropolis Light Transport algorithm while the second one performs the sequence of *MTMH* mutations and executes the *SIMD* part of the shaders: the ray generation and the *BRDF* evaluations are completely vectorized and encapsulated in a user-friendly interface.

We finally have to notice that we did not implement a ray packet *frustum* technique so that much better performance can be expected if they were used (particularly for *EDD* lens sub-paths).

7.1. Comparison with Metropolis Light Transport

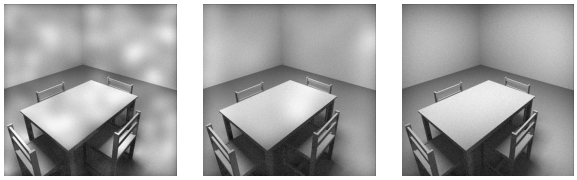
The first remark we can made is that, in a sense, our technique strictly contains *MLT*. Indeed, if you implement *MLT* and use the same number of perturbations and bidirectional mutations (it is what Veach proposed in [VG97]), implementing *CMLT* with length 1 *MTMH* sub-sequences and only 1 *MTMH* candidate will roughly provide the same results. However, *CMLT* adds three parameters which have to be analyzed:

- the standard deviation σ of the Gaussian random variables;
- the number of *MTMH* candidates (and competitors);
- the length of the *MTMH* sub-sequences.

Length of *MTMH* Sub-sequences
We first noticed that our method was almost insensitive to the sub-sequence length since our samples already follow density $f^{(c)}$. However, since the initial samples provided by a path tracer do *not* follow $f^{(c)}$, this behavior must be quite different if you combine *MTMH* sub-sequences and an Energy Redistribution Path Tracer: it would be certainly more interesting to use long *MTMH* sub-sequences. Finally, in a more complex rendering system (with for example, geometry caches), this parameter may need to be more carefully tuned.

σ and the Number of *MTMH* Candidates
Even if the algorithm remains unbiased for all values of σ, we have to carefully balance the number p of *MTMH* candidates and the size of the *MTMH* perturbations. Figure 5 shows a simple scene tested with different values of σ and $p = 256$. As we can see, generating too many *MTMH* candidates in a too small area causes very poor results. Indeed, Metropolis Hasting does not stratify well so that we directly inherit this poor stratification for small values of σ. On the contrary, the jittered sampling used for the *MTMH* perturbations greatly enhances the stratification of the sample set. In a sense, *CMLT* also offers a trade-off between variance and stratification over the image plane.

For a given implementation, it therefore primordial to carefully tune the respective values of σ and p. For all the computed pictures in this article, we set $p = 256$ and $\sigma = 5\%$ of the screen size.



(a) $\sigma = 16$ pixels (b) $\sigma = 32$ pixels (c) $\sigma = 64$ pixels

Figure 5: *Different value of σ. For the 3 tests, we use 256 MTMH candidates and 256 MTMH competitors. The screen size is equal to 1024×1024 and about 20 mutations per pixel have been evaluated.*

7.2. Overall Performance

As indicated above, we perform the *MTMH* perturbations with a multi-threaded coherent ray tracer using the *SSE SIMD* instruction set. Compared to a non-vectorized implementation, we achieved a speed-up varying from 2.3 to 1.5: the acceleration actually depends on the lens sub-path length, the number of *MTMH* candidates and the size of the Gaussian perturbations. Once again, it is important to carefully tune the sampling parameters to maintain computation speed, good stratification properties and low variance. With

our current implementation, we achieved between 1 or 2 million mutations per second on most of the scenes we tested. Our implementation seems to be very competitive compared to other existing *MLT* renderers. We actually think that aggressively exploring the lens sub-path space offers a good sampling strategy since perturbations are much less computationally expensive than bidirectional mutations. We finally believe that using ray packet frustums can still bring major improvements as speed-ups superior to 10 have been recently reported [RSH05, WBS06].

7.3. Cache Simulation

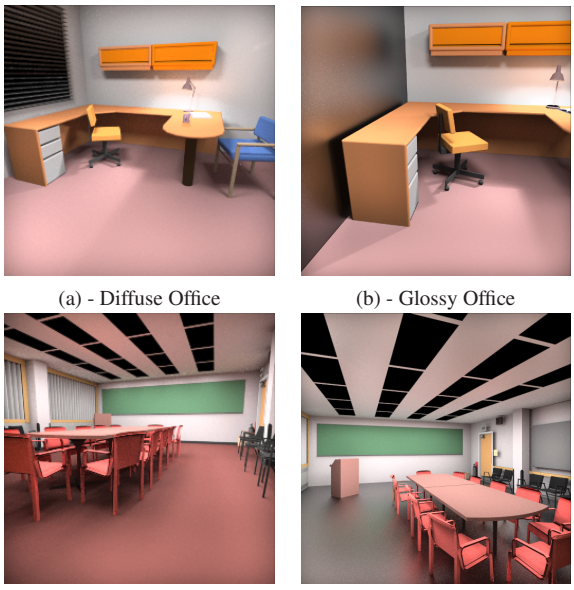
Christensen et al. recently published two papers about distribution ray tracing in complex scenes [CLF*03, Chr06]. We believe that our technique, combined with multi-resolution geometry caching, can fit in production renderers such as the ray tracing system used within the RenderMan interface. Indeed, *MTMH* mutations are, in essence, very close to a standard distribution ray tracing framework: instead of generating reflections, refractions or shadow rays for a *uniform* set of pixels, we actually use the same strategies for a *Gaussian* pixel distribution. To analyze the performance of our technique, we simulated and implemented a simple cache system.

We actually associate a fixed size cache to the triangles of our test scenes (see Figure 6). When a ray / triangle intersection is requested, we check if the triangle is in the cache: if we find it, it is a cache hit, otherwise, it is a cache miss and the triangle is inserted in the cache. To evaluate the performance, we finally use four test scenes: a purely diffuse office, the same scene but with a glossy wall, a purely diffuse conference room and the same scene with a glossy floor. The goal is to exhibit the coherence properties of the tested algorithms with various layouts. We compare our method (*CMLT*) with Metropolis Light Transport (*MLT*) and Instant Radiosity (*IR*) [Kel97] which is one of the most coherent and successful methods when combined with coherent ray tracing techniques. We noticed that *CMLT* provided results which are very similar to the results obtained with *IR*. Actually, the two methods have close algorithmic properties. For *IR*, we implemented a multi-threaded rendering tile system: each thread renders small tiles which are associated to a set of *VPLs*. At the same time, our *CMLT* rendering system finally replaces the tiles by a jittered Gaussian path distribution. Once the rays have been generated, the operations are quasi-identical.

Compared to *IR*, we however have to remind that a *CMLT* system requires to write *anywhere* in the frame buffer which can be limiting in a multi-threaded environment.

8. Limitations and Future Work

As an extension of Metropolis Light Transport, our method inherits some of its drawbacks.



(a) - Diffuse Office (b) - Glossy Office

(c) - Diffuse Conference Room (d) - Glossy Conference Room

Figure 6: *The four scenes to test the coherency of our algorithm with a simulated cache system. The* 1024×1024 *pictures presented here, are computed in about* 5 *minutes on a Core Duo and a* 2×2 *SIMD ray packet system. Office contains* 35000 *triangles and Conference Room* 200000 *triangles*

Flickering Problems

The first one is the difficulty to handle animations without flickering problems. This issue has actually two main reasons:

- as the samples are sequentially dependent, if one of them considerably changes from one frame to the next one, all the following samples in the Markov Chain will be consequently modified and major flickering issues will suddenly occur;
- as Metropolis Light Transport is a pure Monte-Carlo technique, we have to use a lot of pseudo-random numbers which are difficult to reuse in a flickering-free way.

We however think that handling flickering issues may be much easier with Coherent Metropolis Light Transport. First, it seems to be possible to store all the *MH* paths (represented by squares in Figure 3) on hard-disk. Indeed, compared to a *MLT* algorithm, most of the computed paths are generated with *MTMH* perturbations and the *MH* paths obtained with bidirectional mutations only are much less numerous. When we compute a new frame, we can therefore re-read the stored paths, mutate them with a sequential sampler, and make them follow the density of the current frame as done by Ghosh et al. to compute illumination with environment maps in [GDH06]. In other

	Diffuse Office			Glossy Office			Diffuse Conference			Glossy Conference		
	4 Tri	16 Tri	128 Tri	4 Tri	16 Tri	128 Tri	4 Tri	16 Tri	128 Tri	4 Tri	16 Tri	128 Tri
MLT	55%	61%	62%	38%	40%	45%	65%	77%	79%	70%	74%	77%
CMLT	87%	92%	99%	82%	91%	98%	80%	95%	98%	81%	95%	99%
IR	87%	92%	99%	81%	90%	98%	81%	96%	98%	82%	95%	99%

Table 1: *Some cache hit statistics with the four test scenes presented in Figure 6. We use 3 cache sizes: 4, 16, and 128 triangles. Coherent Metropolis Light Transport (CMLT) easily outperforms standard Metropolis Light Transport (MLT). It roughly behaves as Instant Radiosity (IR) does.*

words, if we have a set of paths which follow the density $f_n^{(c)}$ of frame n, a sequential sampler can mutate these paths to make them follow $f_{n+1}^{(c)}$. As the *MTMH* samples do not need to be independent, we can moreover use deterministic quasi random sequences of samples which can easily be regenerated from one frame to the next one. These ideas may thus provide satisfactory results to handle flickering problems since most of the paths of the previous frame could be "reprojected" in an unbiased way into the next frame.

Other Sampling Strategies

The *MTMH* mutations strategies can finally certainly be applied to other samplers: there must be no specific problems to handle participating medias, motion blurs or a spectral representation of materials. Furthermore, *MTMH* mutations can certainly be combined to an Energy Redistribution Path Tracing algorithm [CTE05]. Instead of performing a sequence of *MH* perturbations, it would be not too difficult to replace them by a sequence of coherent *MTMH* candidates as we did with *MLT*.

Implementing a Distributed Framework

One difficult challenge is to efficiently implement a Metropolis Light Transport algorithm with a computer cluster. With our current implementation, it seems impossible to achieve interactive or real time frame rate, since running separate *MLT* on separate threads / computer is not appropriate to a distributed frame work (it would require a too large bandwidth to gather all pictures). A good approach would perhaps be to use an Interleaved Sampling technique [KH01] and therefore to compute the contributions of different pixel sub-sets on different machines. This would require only few changes into the different *MLT* samplers. Combining *MLT* and Interleaved Sampling could also provide good results with inhomogeneous multi-core architectures like Cell processors.

9. Conclusion

We present in this paper, a coherent extension of Metropolis Light Transport which can easily be combined with most of the recent advances achieved in the field of ray tracing. By adding Multiple-Try perturbations, we can coherently compute most of the intersection and *BRDF* operations by efficiently using ray packets or ray packet frustums. We now believe that an interactive and even real time *MLT* system, can be implemented: if the remaining problems previously presented are resolved, we could finally have an unconditionally robust rendering system which could handle any scene of any complexity and as *MLT* initially did, any lighting layouts without the necessity to use aggressive filters or bias the method.

References

[APSS01] ASHIKHMIN M., PREMOŽE S., SHIRLEY P., SMITS B.: A variance analysis of the Metropolis Light Transport algorithm. *Computers and Graphics* (2001), 287–294.

[Ben06] BENTHIN C.: *Realtime Ray Tracing on Current CPU Architectures*. PhD thesis, Computer Graphics Group, Saarland University, 2006.

[Chr06] CHRISTENSEN P.: Ray tracing for the movie "cars". In *IEEE Symposium on Interactive Ray Tracing* (2006), pp. 1–6.

[CL07] CRAIU R. V., LEMIEUX C.: Acceleration of the multiple-try metropolis algorithm using antithetic and stratified sampling. *Statistics and Computing* (2007).

[CLF*03] CHRISTENSEN P. H., LAUR D. M., FONG J., WOOTEN W. L., , BATALI D.: Ray differentials and multiresolution geometry caching for distribution ray tracing in complex scenes. *Computer Graphics Forum (Proceedings of Eurographics 2003)* (2003), 543–552.

[CTE05] CLINE D., TALBOT J., EGBERT P.: Energy redistribution path tracing. In *SIGGRAPH '05 (Proceedings of the 32th annual conference on Computer graphics and interactive techniques)* (2005), pp. 1186–1195.

[DLW93] DUTRE P., LAFORTUNE E. P., WILLEMS Y. D.: Monte carlo light tracing with direct computation of pixel intensities. In *Compugraphics '93 (Proceedings of Third International Conference on Computational Graphics and Visualization Techniques)* (1993).

[DWBS03] DIETRICH A., WALD I., BENTHIN C., SLUSALLEK P.: The OpenRT Application Programming Interface – Towards A Common API for Interactive Ray

Tracing. In *Proceedings of the 2003 OpenSG Symposium* (2003), pp. 23–31.

[GDH06] GHOSH A., DOUCET A., HEIDRICH W.: Sequential sampling for dynamic environment map illumination. In *Proceedings of the 17th Eurographics Symposium on Rendering* (2006), pp. 116–126.

[GFW*06] GÜNTHER J., FRIEDRICH H., WALD I., SEIDEL H.-P., SLUSALLEK P.: Ray tracing animated scenes using motion decomposition. *Computer Graphics Forum* (2006), 517–525.

[Hec90] HECKBERT P. S.: Adaptive radiosity textures for bidirectional ray tracing. In *SIGGRAPH '90 (Proceedings of the 17th annual conference on Computer graphics and interactive techniques)* (1990), pp. 145–154.

[JSL00] JUN S. LIU FAMING LIANG W. H. W.: The multiple-try method and local optimization in metropolis sampling. *Journal of the American Statistical Association* (2000), 121–134.

[Kaj86] KAJIYA J. T.: The rendering equation. In *SIGGRAPH '86 (Proceedings of the 13th annual conference on Computer graphics and interactive techniques)* (1986).

[Kel97] KELLER A.: Instant radiosity. In *SIGGRAPH '97 (Proceedings of the 24th annual conference on Computer graphics and interactive techniques)* (1997), pp. 49–56.

[KH01] KELLER A., HEIDRICH W.: Interleaved sampling. In *Proceedings of the 12th Eurographics Workshop on Rendering* (2001).

[KSKAC02] KELEMEN C., SZIRMAY-KALOS L., ANTAL G., CSONKA F.: A simple and robust mutation strategy for the metropolis light transport algorithm. In *Computer Graphics Forum (Proceedings of Eurographics 2002)* (2002), pp. 531–540.

[LPP99] LÁSZLÓ S.-K., PÉTER D., PURGATHOFER W.: On the start-up bias problem of metropolis sampling. In *Proceedings of Winter School of Computer Graphics Conference* (1999).

[LW93] LAFORTUNE E. P., WILLEMS Y. D.: Bidirectional path tracing. In *Compugraphics '93 (Proceedings of Third International Conference on Computational Graphics and Visualization Techniques)* (1993).

[Pha03] PHARR M.: Chapter 9: Metropolis sampling. In *Monte-Carlo Ray Tracing, SIGGRAPH'03 Course 44, course notes* (2003).

[PKK00] PAULY M., KOLLIG T., KELLER A.: Metropolis light transport for participating media. In *Rendering Techniques 2000 (Proceedings of the 11th Eurographics Workshop on Rendering)* (2000).

[RSH05] RESHETOV A., SOUPIKOV A., HURLEY J.: Multi-level ray tracing algorithm. *ACM Transaction on Graphics* (2005), 1176–1185.

[Vea97] VEACH E.: *Robust Monte-Carlo Methods for Light Transport Simulation*. PhD thesis, Standford University, 1997.

[VG94] VEACH E., GUIBAS L.: Bidirectional estimators for light transport. In *Proceedings of Eurographics Rendering Workshop* (1994), pp. 147–162.

[VG97] VEACH E., GUIBAS L.: Metropolis light transport. *SIGGRAPH '97 (Proceedings of the 24th annual conference on Computer graphics and interactive techniques)* (1997), 65–76.

[Wal04] WALD I.: *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Computer Graphics Group, Saarland University, 2004.

[WBS06] WALD I., BOULOS S., SHIRLEY P.: Ray Tracing Deformable Scenes using Dynamic Bounding Volume Hierarchies (revised version). *ACM Transactions on Graphics* (2006).

[WBWS01] WALD I., BENTHIN C., WAGNER M., SLUSALLEK P.: Interactive rendering with coherent ray tracing. *Computer Graphics Forum (Proceedings of Eurographics 2001)* (2001).

[WIK*06] WALD I., IZE T., KENSLER A., KNOLL A., PARKER S. G.: Ray Tracing Animated Scenes using Coherent Grid Traversal. *ACM Transactions on Graphics* (2006).

[WK06] WÄCHTER C., KELLER A.: Instant raytracing: The bouding interval hierarchy. In *Rendering Techniques 2006 (Proceedings of the 17th Eurographics Symposium on Rendering)* (2006).