Physically-based Animation Rendering with Markov Chain Monte Carlo

Yu-Chi Lai
Feng Liu
Charles Dyer

Technical Report #1653

March 2009

EUROGRAPHICS 2009 / P. Dutré and M. Stamminger
(Guest Editors)

Volume 28 (2009), Number 2

Physically-based Animation Rendering with Markov Chain Monte Carlo

Y.-C. Lai[1] and F. Liu[1] and C. Dyer[1]

[1]University of Wisconsin at Madison, U.S.A.

Abstract

Exploring temporal coherence among light transport paths is very important to remove temporally perception-sensitive artifacts in animation rendering. Using the contribution of a light transport path to all frames in an animation as the sampling distribution function allows us to adapt Markov Chain Monte Carlo (MCMC) algorithms to exploit the temporal and spatial coherence among paths in order to generate a perceptually pleasant animation. A new perturbation technique called time perturbation *is developed to explore the temporal coherence among paths. Furthermore, in order to make animation rendering plausible, we distribute iterative computational tasks to a pool of computers for parallel computation. Each task is rendered with a set of parameters adapted according to the local properties of each task. We demonstrate that this local adaptation does not introduce bias statistically. The resulting animations are perceptually better than those rendered in a frame-by-frame manner.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Raytracing

1. Introduction

A large number of movies employ visual effects using computer graphics. Photorealistic rendering is a critical element to create realistic visual effects. An animation is generally rendered as a sequence of static images in a frame-by-frame manner by using Monte Carlo algorithms describe in [PH04, DPSaB06]. As a result the rendered animations normally contain annoying temporal artifacts such as flickering and shimmering if the computation time is not long enough. Therefore, some research on global illumination has started to pay attention to enhance the temporal coherence among frames in order to remove temporal artifacts. Caching photons and irradiances of sample paths [CJ02, VMKK00, MTAS01, DBMS02, WMM*04, TMS04, SKDM05, GBP07] from preceding and following frames can greatly improve the computational efficiency as well as reduce temporal aliasing. [MA06] caches computed irradiances from different moments of time on the surfaces. The cached values are used to create a basis for temporal interpolation of irradiance in order to greatly reduce temporal artifacts. However, the fundamental problem of these techniques is that the invalidity of cached samples across frames introduces bias and error into the final result. When only the camera and the point lights are allowed to move inside an animation scene, it is easy to reuse all light paths by re-evaluating the path contributions [BSH02, HBS03, SSSK04, SC04, SCH04, HBS03] or reweighing samples based on the multiple-importance framework [MFSSk06]. However, movement limitations reduce the utility. Havran et al. [HDMS03] reused the static-object paths from bi-directional path tracing to reduce temporal flicking. For each static candidate path, they first check the validity of the path, i.e. no edge is blocked by moving objects and then the contribution of valid static paths is evaluated by recomputing the BRDF value at the first hit point from the camera. The reweighting scheme introduces bias and causes unwanted artifacts. In addition, they only intended to reuse static paths not all paths. Ghosh et al. [GDH06] applied the framework of Sequential Monte Carlo to exploit the temporal coherence among paths by reweighing the samples in previous frames to generate good samples for the current frame. However, their work is limited to environment map lighting. In this work we intend to exploit the temporal coherence among all path integrals in the entire animation whose animated entities are described by key-framed rigid transformation.

Markov Chain Monte Carlo (MCMC) algorithms such as Metropolis Light Transport (MLT) [VG97, KSKAC02], Energy Redistribution Path Tracing (ERPT) [CTE05], and Population Monte Carlo Energy Redistribution (PMC-ER) [LFCD07, LLZD08] have demonstrated the strengths of exploring the spatial coherence among path integrals when rendering a static image. However, all these algorithms are originally formulated to render a static image without considering the temporal coherence among frames in an animation. Thus, the temporal flickering artifacts is perceptually unpleasant in the rendered animations. But we expect that they are good candidates for exploring the temporal coherence among path integrals. We formulate the contribution of a light transport path to all frames in an animation as the sampling distribution for MCMC algorithms. This formulation allows us to extend from rendering a static image to rendering an animation. As a result, a new *time perturbation* method is designed to reuse path samples with similar properties at different moments of time

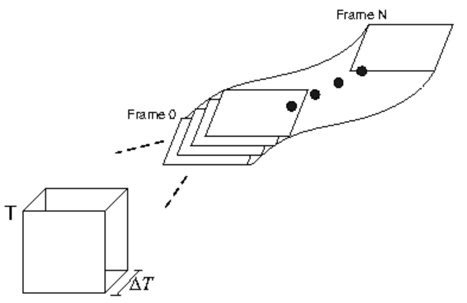

Figure 1: *The image sweep is a set of three dimensional hypercubes which correspond to the frames, $Frame_1, \ldots, Frame_N$ and formed by all image planes aligned with the movement of the camera at different frame periods. Please notice that k is used to index the frame and t is used to indicate the moment when a path is sampled. Valid path samples for frame k are generated between T and $T+\Delta T$, where T is the moment when the shutter opens. In addition, all pixel samples in the valid period must also spatially pass through the image plane $\mathcal{I}^t$ centered at the center of the camera. In other words, the image sweep is the sweep of the image plane along the camera's locomotion in space along the time between T and $T+\Delta T$.*

to explore the temporal coherence among path integrals in order to reduce the temporal artifacts of an animation. In addition to having an algorithm to explore the temporal coherence, we also need to face computational limitation when rendering the entire animation sequence. Therefore, our rendering system is designed to render an animation by distributing a subset of iterative computational tasks to a pool of computers for parallel computation. Each computational task contains initial paths traced by a general path tracing algorithm from a designated frame for energy redistribution. For efficient usage of parallel computation, we present a formulation to allow us locally adjust the rendering parameters in each task without introducing bias. In this paper we demonstrate the strength of exploiting the temporal coherence among paths by building the temporal perturbations on the PMC-ER algorithms for parallel rendering. The animations rendered with temporal perturbations are perceptually more pleasant. However, we also present a short discussion about applying the entire structure to other MCMC algorithms.

2. Animation Formulation for Markov Chain Monte Carlo

When rendering an animation, the intensity of the j-th pixel of the k-th frame, I_j^k can be expressed as path integrals:

$$\begin{aligned}
I_j^k &= \int_\Omega f_j^k(\tilde{\mathbf{x}}^t)\,d\mu(\tilde{\mathbf{x}}^t) \quad and \\
f_j^k(\tilde{\mathbf{x}}^t) &= L_e(\mathbf{x}_0^t,\mathbf{x}_1^t)G(\mathbf{x}_0^t,\mathbf{x}_1^t) \\
&\quad \left\{\prod_{n'=1}^{N_v-2} f_s(\mathbf{x}_{n'-1}^t,\mathbf{x}_{n'}^t,\mathbf{x}_{n'+1}^t)G(\mathbf{x}_{n'}^t,\mathbf{x}_{n'+1}^t)\right\} \\
&\quad f_s(\mathbf{x}_{N_v-2}^t,\mathbf{x}_{N_v-1}^t,\mathbf{x}_{N_v}^t)G(\mathbf{x}_{N_v-1}^t,\mathbf{x}_{N_v}^t)W_j^k(\mathbf{x}_{N_v-1}^t,\mathbf{x}_{N_v}^t) \\
&= W_j^k(\tilde{\mathbf{x}}^t)f(\tilde{\mathbf{x}}^t)
\end{aligned}$$

where t is the moment when the light transport event happens, Ω represents the set of all valid light transport paths that begin on a point of a light surface, interact with the scene surfaces, and end at the camera for the entire animation denoted as $\tilde{\mathbf{x}}^t$, $\mu(\tilde{\mathbf{x}}^t)$ is the surface area product measure for the path $\tilde{\mathbf{x}}^t$, f_j^k is the contribution brought by a light transport path to pixel j of frame k, f_s is the bidirectional scattering distribution function, L_e is the emittance radiance of a light source, W_j^k is sensor response, G is the geometry term, and f is the radiance carried by the path. The contribution function can be decomposed into two components: sensor response, W_j^k, and the radiance carried by the path, f. Because the radiance is independent of the response of the given pixel, the radiance can be reused for all pixels (which is the same for all pixels in all frames). In other words, $b = \int_\Omega f(\tilde{\mathbf{x}}^t)d\mu(\tilde{\mathbf{x}}^t)$ can be used to express the total radiant energy delivered through the image sweep as shown in Figure 1. The intensity for each pixel can be estimated by Monte Carlo integration

$$I_j^k = \int_\Omega W_j^k(\tilde{\mathbf{x}}^t)f(\tilde{\mathbf{x}}^t)d\mu(\tilde{\mathbf{x}}^t) \approx \frac{1}{N}\sum_1^N W_j^k(\tilde{\mathbf{X}}_i^{T_i})\frac{f(\tilde{\mathbf{X}}_i^{T_i})}{p(\tilde{\mathbf{X}}_i^{T_i})}$$

where $p(\tilde{\mathbf{X}}_i^{T_i})$ is the importance sampling function used by Monte Carlo integration. The idea of Markov Chain Monte Carlo algorithms is to generate a set of correlated paths according to their radiance brought to the image sweep. Through the process, paths mutate, and newly mutated paths are accepted or rejected with a carefully chosen probability to ensure that paths are sampled according to their radiance brought to the image sweep. In other words, $p(\tilde{\mathbf{x}}^t) = \pi(\tilde{\mathbf{x}}^t) = \frac{f(\tilde{\mathbf{x}}^t)}{b}$ is implicitly used as the target distribution density. Under this formulation, we can design temporal and spatial mutations by considering the coherence among all valid paths in the entire animation.

Veach [Vea97] purposed that using importance energy, f/p, of the seed path to weigh each mutation of a Markov Chain can remove the start-up bias if the detail balance is maintained at each mutation. Therefore, we can naturally come up ERPT balance energy transfer which a hybrid algorithm combining the independently sampling path tracing algorithm with the correlatedly sampling Metropolis algorithm. The idea is to generate a set of independent samples by path tracing as seed paths. Then each seed path is used to correlatedly explore a local path space, $\Omega_{\tilde{\mathbf{x}}}$, around the seed path by Metropolis. $\Omega_{\tilde{\mathbf{x}}}$ is a sub-space of Ω and defined as the domain of all paths which can be reached through a sequence of perturbations from the seed path. This definition is to guarantee that the local exploration can reach and stay at the local stationary probability, $f(\tilde{\mathbf{x}}^t)/b_{\tilde{\mathbf{x}}}$ where $b_{\tilde{\mathbf{x}}} = \int_{\Omega_{\tilde{\mathbf{x}}}} f(\tilde{\mathbf{x}}^t)d\mu(\tilde{\mathbf{x}}^t)$. The estimation of the pixel intensity can be calculated as

$$I_j^k = \frac{1}{N_{MC}}\sum_{i=1}^{N_{MC}}\left\{\frac{1}{N_{MCMC}}\sum_{l=1}^{N_{MCMC}}\frac{f(\tilde{\mathbf{X}}_i^{T_i})}{p(\tilde{\mathbf{X}}_i^{T_i})}W_j^k(\tilde{\mathbf{X}}_{il}^{T_{il}})\frac{f(\tilde{\mathbf{X}}_{il}^{T_{il}})}{\frac{f(\tilde{\mathbf{X}}_{il}^{T_{il}})}{b_{\tilde{\mathbf{X}}_\mathbf{i}}}}\right\} \quad (1)$$

The variance analysis in [APSS04] allows us to bound the variance of the Markov Chain of a seed path as $V_{MCMC}(\tilde{\mathbf{X}}_i) \le \frac{1}{N_{MCMC}}b_{\tilde{\mathbf{X}}_\mathbf{i}}N_{pixel}\tilde{E}_j^k(1+\frac{2R(1)}{q(1-q)-R(1)})$ where N_{pixel} is the number of pixels, $\tilde{E}_j^k$ is the average importance energy of the path associated with this pixel, q is the probability that path contributes to the pixel under considerations, and $R(1)$ is the correlation between random variance indicating that two subsequent paths go through the same pixel. The variance for the entire hybrid estimator can be calculated as $VC(I_j^k) = \frac{1}{N_{MC}^2} \times N_{MC} \times \frac{1}{N_{MCMC}^2}\sum_{l=1}^{N_{MCMC}} V_{MCMC}(\tilde{\mathbf{X}}_i^{T_i}) \le \frac{1}{N_{MC}\times N_{MCMC}}b_{\tilde{\mathbf{X}}_\mathbf{i}}N_{pixel}\tilde{E}_j^k(1+$

$\frac{2R(1)}{q(1-q)-R(1)}$). Thus the variance of the entire algorithm is $O(1/N_{MC})$, and the algorithm converges to the correct answer if the number of independent samples go to infinity. The expectation of the Markov Chain is $\mathcal{E}(\tilde{\mathbf{X}}_i) = \frac{1}{N_{MCMC}} \sum_{l=1}^{N_{MCMC}} \int_{\Omega_{\tilde{\mathbf{x}}}} \frac{f(\tilde{\mathbf{x}}^t)}{\frac{f(\tilde{\mathbf{x}}^t)}{b_{\tilde{\mathbf{x}}}}} \frac{f(\tilde{\mathbf{x}}^t)}{b_{\tilde{\mathbf{x}}}} d\mu = 1$ and then the expectation of the entire algorithm becomes $\mathcal{E}(I_j^k) = \mathcal{E}(\frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} \frac{f(\tilde{\mathbf{X}}_i^{T_i})}{p(\tilde{\mathbf{X}}_i^{T_i})} = I_j^k$

Veach [Vea97] also purposed that we can use an equal weighting scheme if we resample the paths according to its importance energy. Thus, we can use a similar deterministic sampling strategy in [Fan06] to generate totally $\sum E_i(\tilde{\mathbf{X}})/\tilde{E}$ paths where $E_i(\tilde{\mathbf{X}}) = \frac{f(\tilde{\mathbf{X}})}{p(\tilde{\mathbf{X}})}$. In other words, each seed path has a number of Markov Chains proportional to its energy.

$$I_j^k = \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} \left\{ \sum_{m=1} N_{path} \frac{1}{N_{MCMC}} \sum_{l=1}^{N_{MCMC}} W_j^k(\tilde{\mathbf{X}}_{il}^{T_{il}}) \frac{f(\tilde{\mathbf{X}}_{il}^{T_{il}})}{\frac{f(\tilde{\mathbf{X}}_{il}^{T_{il}})}{b_{\tilde{\mathbf{X}}_\mathbf{i}}}} \right\} \quad (2)$$

where $N_{path} = N_{floor} + (E(\tilde{\mathbf{X}}_i^{T_i}) - N_{floor} + U(0,1)) \times \tilde{\mathbf{E}}$ and $N_{floor} = \lfloor \frac{E(\tilde{\mathbf{X}}_i^{T_i})}{\tilde{\mathbf{E}}} \rfloor$ The equally weighting scheme now becomes ERPT equal deposition. We can observe that $\mathcal{E}(N_{path}) = \frac{E(\tilde{\mathbf{X}}_i^{T_i})}{\tilde{\mathbf{E}}}$. The convergence and expectaction analysisis similar to the one discussed in the previous paragraph. [CTE05] described two criteria to maintain the unbiasedness of ERPT equal deposition: first, the detail balance of mutation is maintained through the process for reaching and staying in the stationary probability; second, the number of Markov Chains is proportional to the energy carried by the path. Later, we will show that although our kernel function for each path adapts from iteration to iteration, it is fixed inside a single Markov Chain. In addition, the acceptance probability is chosen to maintain the detail balance at each perturbation and thus the stationary probability requirement is achieved for us to reach convergence and maintain unbiasedness. Furthermore, our algorithm deposits the remaining energy of an eliminated path by the equal deposition methods back into the image sweep before a path is removed from the population in resampling process or terminated at the end of a task. This achieves the second criterion of unbaisedness.

3. Population Monte Carlo Energy Redistribution

When rendering an animation, we distribute the computation according to the starting time of initial paths belonging to the same frame to a task. Thus, in the following we will present PMC-ER Equal Deposition (PMC-ER-E) for a single computational task. Then we describe the details of time perturbation. Finally, we present the formulation to let us locally adjust the sampling parameters for each task without introducing bias.

3.1. Population Monte Carlo Energy Redistribution with Equal Deposition

Figure 2 shows the PMC-ER-E algorithm executed in a single computational task in a single computer. At the beginning of each computational task, a pool of pixel positions and a pool of caustics paths are generated for the initial population and the replacement paths used in the resampling process. In each inner loop, z, we spatially or temporally perturbed each member path N_{equal} times to redistribute the energy to the neighborhood of the path according to a mixture distribution:

$$\begin{aligned} K_i^{(o)}(\tilde{\mathbf{Y}}_i^{(z-1)} \to \tilde{\mathbf{y}}_i^{(z)}) &= \sum_{d_h} \alpha_{i,d_h}^{(o)} T_{spatial}(\tilde{\mathbf{Y}}^{(z-1)} : d_h \to \tilde{\mathbf{y}}^{(z)}) \\ &+ \sum_{t_h} \alpha_{i,t_h}^{(o)} T_{temporal}(\tilde{\mathbf{Y}}^{(z-1)} : t_h \to \tilde{\mathbf{y}}^{(z)}) \end{aligned}$$

When perturbing a path, we first choose one perturbation from the set of spatial and temporal perturbations according to the weights, $\alpha_i^{(o)}$, where $\sum_{perb} \alpha_i^{(o)} = 1$. The current path is perturbed to generate a new perturbed path according to the perturbation parameter d_h for a spatial perturbation and t_h for a temporal perturbation. The acceptance probability is chosen as $a(\tilde{\mathbf{Y}}^{t_1}|\tilde{\mathbf{X}}^{t_0}) = \min \left\{ 1.0, \frac{f(\tilde{\mathbf{Y}}^{t_1}) T_{op-type}(\tilde{\mathbf{X}}^{t_1})|\tilde{\mathbf{Y}}^{t_1}}{f(\tilde{\mathbf{X}}^{t_0}) T_{op-type}(\tilde{\mathbf{Y}}^{t_1}|\tilde{\mathbf{X}}^{t_0})} \right\}$ where $T_{op-type}$ is the type of perturbation chosen. We can observe that the detail balance of this perturbation is maintained by $f(\tilde{\mathbf{X}}^{t_0}) T_{op-type}(\tilde{\mathbf{Y}}^{t_1}|\tilde{\mathbf{X}}^{t_0}) = f(\tilde{\mathbf{Y}}^{t_1}) T_{op-type}(\tilde{\mathbf{X}}^{t_0}|\tilde{\mathbf{Y}}^{t_1}) \frac{f(\tilde{\mathbf{Y}}^{t_1}) T_{op-type}(\tilde{\mathbf{X}}^{\mathbf{t_1}}|\tilde{\mathbf{Y}}^{t_1})}{f(\tilde{\mathbf{X}}^{t_0}) T_{op-type}(\tilde{\mathbf{Y}}^{t_1}|\tilde{\mathbf{X}}^{t_0})}$ if $f(\tilde{\mathbf{Y}}^{t_1}) T_{op-type}(\tilde{\mathbf{X}}^{t_1}|\tilde{\mathbf{Y}}^{t_1}) \geq f(\tilde{\mathbf{X}}^{t_0}) T_{op-type}(\tilde{\mathbf{Y}}^{t_1}|\tilde{\mathbf{X}}^{t_0})$; otherwise $f(\tilde{\mathbf{X}}^{t_0}) T_{op-type}(\tilde{\mathbf{Y}}^{t_1}|\tilde{\mathbf{X}}^{t_0}) \frac{f(\tilde{\mathbf{Y}}^{t_1}) T_{op-type}(\tilde{\mathbf{X}}^{t_1}|\tilde{\mathbf{Y}}^{t_1})}{f(\tilde{\mathbf{X}}^{t_0}) T_{op-type}(\tilde{\mathbf{Y}}^{t_1}|\tilde{\mathbf{X}}^{t_0})} = f(\tilde{\mathbf{Y}}^{t_1}) T_{op-type}(\tilde{\mathbf{X}}^{t_0}|\tilde{\mathbf{Y}}^{t_1})$. After each perturbation, $E_d = R_i^k * e_d$ energy is deposited at the pixel position of the newly mutated path, $\tilde{\mathbf{Y}}_i^{(z)}$, and e_d energy is removed from the path's $E_{i,remain}$. The energy-deposit constant, R_i^k, is computed according to the properties of this frame such as the number of samples in each pixel and the number of caustics paths in order to maintain the energy delivered from each pixel or each type of paths statistically the same. At the end of each o loop, the resampling process is applied. Paths in the population are eliminated according to its $E_{i,remain}$. If there is any energy remaining in the eliminated paths, the remaining energy is deposited by equal deposition into the image sweep. To maintain the constant number of population, we replace the eliminated path with newly generated paths from the pool of stratified and variance-regeneration pixel positions or unused paths from the pool of caustics paths [LLZD08]. We also adapt α values according to the performance of perturbations. We use the acceptance probability of each perturbation as indication to determine the probability of perturbation choice [LFCD07]. After finishing the o loop, the algorithm deposits the remaining energy of the population paths onto the image sweep before exiting. **The remaining energy in eliminated paths and terminated population paths at the end of the task is deposited to the image sweep to guarantee the number of Markov Chains starting from each path proportional to its initial energy for maintaining unbiasedness.**

3.2. Time Perturbation

In order to exploit the temporal coherence, we design a new path perturbation strategy called *time perturbation*. In this section we first describe how to use our temporal perturbation method to reconstruct the path temporally and how to calculate the tentative transition probability accordingly.

The main idea behind time perturbation is that when a path $\tilde{\mathbf{X}}^{t_0}$ exists at time t_0, there may be a correlated path $\tilde{\mathbf{Y}}^{t_1}$ whose vertices are *object-based* rigid transformations of the vertices in the

```
get $\tilde{E}$, $e_d$, $T$, $k$, $R^k_{C2G}$, $N^k_{sample}$, $N^k_{caustics}$, $N_{equal}$, $\gamma^k$
 $N_{population}$, $N_{uniform}$ from the host
generate a pool of pixel positions and a set of caustics paths
generate initial population of paths in $s = 1$ at the moment $T$
  for $o = 1, \cdots, N_{iteration}$
  determine $\alpha_i^{(o)}$ for each perturbation
  for $i = 1, \cdots, N_{population}$
    if $E_{i,remain} + U(0,1)\tilde{E} > \tilde{E}$
      for $z = 1, \cdots, N_{equal}$
        generate $\tilde{\mathbf{Y}}_i'^{(z)} \sim K_i^{(o)}(\tilde{\mathbf{y}}^{(z)} | \tilde{\mathbf{Y}}_i^{(z-1)})$
          $\tilde{\mathbf{Y}}_i^{(z)} = (U(0,1) < a(\tilde{\mathbf{Y}}_i'^{(z)} | \tilde{\mathbf{Y}}_i^{(z)}))\ ?\tilde{\mathbf{Y}}_i'^{(z)} : \tilde{\mathbf{Y}}_i^{(z-1)}$
          deposit $E_d = R^i_k * e_d$ on $\tilde{\mathbf{Y}}_i^{(z)}$
          $E_{i,remain} -= e_d$
    $w_i^{(o)} = E_{i,remain}$
   resample the population: elimination and regeneration
 deposit remaining energy in all population paths
```

Figure 2: *The PMC-ER equal deposition iteration loop. $\tilde{E}$ is averaging path energy, e_d is deposition energy of each perturbation, R^k_{C2G} is the ratio between caustics and all paths in this frame, $N_{population}$ is the size of the population, N^k_{sample} is the total number of initial seed paths, $N_{uniform}$ is the number of stratified samples per pixel, $N^k_{caustics}$ is the number of caustics-generation paths, T is the time when the shutter opens, k is the index of the frame, $N^k_{iteration}$ is the total number of task iterations, $N^k_{variance}$ is the number of variance-regeneration paths, and γ^k_j is the variance-regeneration distribution function. The system use path tracing with a fixed number of samples per pixel to estimate $\tilde{E}$, e_d, R^k_{C2G}, and γ^k_j which is proportional to the variance of the sample radiances in the given pixel in a frame-by-frame manner. $U(0,1)$ generates a random number uniformly distributed between 0 and 1. $E_{i,remain}$ is the energy remaining in the population path i after the inner energy redistribution loops. R^i_k is the energy-adapted constant discussed in Section 3.3.*

path $\tilde{\mathbf{X}}^{t_0}$ [BS96]. The object-based rigid transformation of a surface point from t_0 to t_1 is computed as $x^{t_1} = M^{t_1}(M^{t_0})^{-1}x^{t_0}$ where M^t denotes the rigid transformation for the surface from the object space to the world space at t and $\mathbf{x}^t$ is the world coordinate of the surface point $\mathbf{x}_{obj}$ at t. A valid perturbed path $\tilde{\mathbf{Y}}^{t_1}$ can be generated if the visibility check passes in each edge. However, **this simple time perturbation method may fail if a path contains a specular vertex because the relation between the input and output direction of a specular vertex is a delta function.** Thus, a more complex scheme is developed for paths containing specular vertices.

Figure 3 shows an example of a valid time perturbation of a path with specular vertices. The step we take to perturb the path is as follow: First, we identify the specular sub-paths of the form $\{L|[(D|L)D]\}(S^+D|S^+)^+\{[D(D|E]|E\}$. Second, the specular sub-paths are reconstructed by either *sampling backward* or *sampling forward*. The criterion for choosing a sampling direction and the details of reconstruction will be described shortly. Finally, un-updated diffuse vertices are object-based rigid transformed to new locations and un-updated edges are linked and updated accordingly.

The first and last step are trivial. Thus, we describe the details

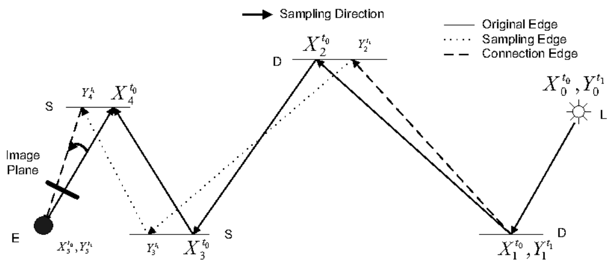

Figure 3: *This is a path with the form LDDSSE and is used to demonstrate the* sample backward *method. We would like to transform the path $\tilde{\mathbf{X}}^{t_0}$ at t_0 with 6 vertices to a new path $\tilde{\mathbf{Y}}^{t_1}$ at t_1. Since the original path has the form of case 1, we have to reconstruct the sub-path by* sampling backward. *Thus, we first object-based rigid transform the position of vertices, $\mathbf{x}_5^{t_0}$ and $\mathbf{x}_4^{t_1}$ to $\mathbf{y}_5^{t_1}$ and $\mathbf{y}_4^{t_0}$ and link the edge $\mathbf{y}_5^{t_1}\mathbf{y}_4^{t_1}$ to form the tracing ray at time t_1. Then, we extend the sub-path through the same specular bounce at $\mathbf{y}_4^{t_1}$ as the corresponding $\mathbf{x}_4^{t_0}$ to get $\mathbf{y}_3^{t_1}$ and the same specular bounce at $\mathbf{y}_3^{t_1}$ as the corresponding $\mathbf{x}_3^{t_0}$ to get $\mathbf{y}_2^{t_1}$. Since $\mathbf{x}_1^{t_0}$ and $\mathbf{x}_0^{t_0}$ are diffuse surfaces, we only need to object-based rigid transform their positions to get $\mathbf{y}_1^{t_1}$ and $\mathbf{y}_0^{t_1}$ and link the edges of $\mathbf{y}_2^{t_1}\mathbf{y}_1^{t_1}$ and $\mathbf{y}_1^{t_1}\mathbf{y}_0^{t_1}$ to form a new path $\tilde{\mathbf{Y}}^{t_1}$.*

of step 2. The criterion used to choose either *sampling backward* or *sampling forward* is:

- **Case 1**: if the path is an eye sub-path of the form $\{L|[(D|L)D]\}(S^+D|S^+)^+E$, *sample backward* from the eye vertex is chosen to reconstruct the sub-path **to ensure that the eye sub-path passes through the camera.**
- **Case 2**: if the path is a light sub-path of the form $L(S^+D|S^+)^+D(D|E)$, *sample forward* is chosen to reconstruct the sub-path **to ensure that the light sub-path starts from a valid light source.**
- **Case 3**: if the path has the form $(L|D)D(S^+D|S^+)^+D(D|E)$, *sample forward* and *sample backward* are randomly chosen to reconstruct the sub-path.

After we make the choice of the reconstruction direction, we have to reconstruct the sub-path in the chosen direction. The following we provide how to reconstruct a sub-path backward. A sub-path can be constructed forward in a similar manner except that the direction of transverse is reverse. We would like to reconstruct the sub-path, $\mathbf{x}_l^{t_0} \cdots \mathbf{x}_m^{t_0}$ of the form $D(S^+D|S^+)^+D$ by using *sampling backward*. First, the position of vertex $m-1$ and m is object-based rigid transformed from t_0 to t_1 to create $\mathbf{y}_m^{t_1}$ and $\mathbf{y}_{m-1}^{t_1}$. Next, the edge $\mathbf{y}_m^{t_1}\mathbf{y}_{m-1}^{t_1}$ is linked to form the starting ray for constructing the sub-path. Since $\mathbf{x}_{m-1}^{t_0}$ is a specular vertex, we choose a specular bounce at $\mathbf{y}_{m-1}^{t_1}$ to find the next vertex $\mathbf{y}_{m-2}^{t_1}$. Then, according to the bounce at the vertices in the original path, we use two different methods to construct the sub-path to have the same length as the original one. If $\mathbf{x}_n^{t_0}$ where $m-2 \geq n \geq l+1$ is a specular vertex, we choose a specular bounce to find the next vertex; otherwise we first transform the vertex $\mathbf{x}_{n-1}^{t_0}$ to get $\mathbf{y}_{n-1}^{t_1}$ by the object-based rigid transform method and link $\mathbf{y}_n^{t_1}$ to $\mathbf{y}_{n-1}^{t_1}$ to form the new edge.

After the path is reconstructed temporally, the tentative transition probability for the time perturbation method is computed by considering all possible ways to reconstruct the path. Since we divide into specular sub-paths and diffuse sub-paths, the tentative transition can be computed by multiplying the tentative transition probability for each sub-path. For a diffuse sub-path, the tentative transition probability is one because the transformation and the edge linking is deterministic. For a specular sub-path, the tentative transition probability is computed according to three different cases discussed in the previous paragraph. The overall tentative transition probability can be computed as:

$$T_{t_h,time} \quad (\tilde{\mathbf{X}}^{t_0} \rightarrow \tilde{\mathbf{Y}}^{t_1}) = \prod_{i=1}^{N_{sub}} T^i_{specular} \;\; and$$

$$T_{specular} = \begin{cases} T_{backward} & Case1 \\ T_{forward} & Case2 \\ 0.5 \cdot T_{forward} + 0.5 \cdot T_{backward} & Case3 \end{cases}$$

$$where \left\{ \begin{array}{ll} T_{forward} = & \frac{1}{t_h} \prod_{j=l+1}^{m-1} \left\{ \frac{G(\mathbf{y}_j^{t_1}, \mathbf{y}_{j+1}^{t_1})}{|\cos\theta_{j,out}|} : 1?\mathbf{y}_j^{t_0} \subset S \right\} \\ T_{backward} = & \frac{1}{t_h} \prod_{j=m-1}^{l+1} \left\{ \frac{G(\mathbf{y}_j^{t_1}, \mathbf{y}_{j-1}^{t_1})}{|\cos\theta_{j,in}|} : 1?\mathbf{y}_j^{t_1} \subset S \right\} \end{array} \right.$$

where$\mathbf{y}^{t_1}_{j+1}$, $T_{forward}$ is the tentative transition probability when reconstructing the sub-path forward, $T_{backward}$ is the tentative transition probability when reconstructing the sub-path backward, and $\theta_{j,in}, \theta_{j,in}$ is the angle between the normal of the surface and the direction of the incoming/outgoing light ray at $\mathbf{y}^{t_0}_j$.

3.3. Iteratively Distribute Samples in a Frame-by-Frame Manner

Since variance-regeneration and caustics-regeneration purposed in [LLZD08] can enhance the rendering efficiency, we would like to have them in our rendering system. The weighting scheme is important to guarantee unbiasedness of the final result but the computation must consider the distribution of pixel positions and caustics paths in the entire animation. This limits us to take full advantage of the parallel rendering. Carefully observing the energy distribution algorithm, we found that the energy delivered to the entire image sweep can be computed by $\tilde{E} = \sum_{i=1}^{N_{MC}} E(\tilde{\mathbf{X}}) \times N_{MCMC} \times e_d$ where the $E(\cdot)$ is the energy of a seed path. The average path energy can be expressed in a iteration-frame-based manner as $\tilde{E} = \sum_{m=1}^{N_{iteration}} \sum_{k=1}^{N_{frame}} \left\{ \sum_{n=1}^{N^{m,k}_{sample}} E(\tilde{\mathbf{X}}) \times N_{MCMC} \times e_d \right\} = \sum_{m=1}^{N_{iteration}} \sum_{k=1}^{N_{frame}} \left\{ \tilde{E}(m,k) \right\}$. where $N_{iteration}$ is the total iterations used, N_{frame} is the total number of frames in the animation, $N^{m,k}_{sample}$ is the total number of samples distributed to the k-th frame in m-th iteration, and $\tilde{E}(m,k) = \sum_{n=1}^{N^{m,k}_{sample}} E(\tilde{\mathbf{X}}) \times N_{MCMC} \times e_d$ denotes the energy delivered at frame k in iteration m. If we can keep $\tilde{E}(m,k)$ statistically unchanged in each frame of each iteration, the energy delivered to the entire image sweep can keep unchanged. This derivation allows us to calculate the energy deposition ratio in each frame according to the number of assigned stratified pixel position to each pixel, the number of variance-regeneration pixel positions in each pixel, and the total number of generated caustics-regeneration paths in that frame without introducing bias [LLZD08].

4. Results

The formulation in Sec. 2 allows us to exploit the temporal and spatial coherence among paths when rendering an animation. However, rendering a physically-correct animation on a single computer seems implausible. Thus, our animation rendering system distributes iterative computational tasks to a pool of computers. Each computational task contains a set of initial seed paths from a designated frame for energy redistribution. According to the discussion in Sec. 3.3, we can locally adjust the energy deposition ratio of each path according to each frame's properties. Then, the iterative computational result from each task is collected to update the intermediate result and create the iterative computational tasks for the next iteration. The process repeats until the rendering process reaches the desired iteration.

To evaluate the performance of exploring temporal coherence among path integrals, we compared animations rendered with time perturbations against animations rendered in a frame-by-frame manner on the Cornell Box (CB) scene, a room scene, a chessboard scene, and a basement scene using the criterion of starting with the same number of initial PT paths and using the same setting for the regeneration methods. Table 1 shows the scene-specific parameters and the statistics gathered from the computation of the first frame in the first iteration for each animation. The reason behind this is that since Condor determines the distribution of computational tasks based on the load of the system and the priority of the user, the completion time of each task is not predictable. It is not fair to compare the overall rendering time between different rendering methods because the load of Condor varies from time to time. As a result even the same rendering algorithm may result in very different overall rendering time. Therefore, we use the time needed for the first frame in the first iteration as a representative of the computation time. Similarly, since intermediate computational result arrives at different time when using Condor, and we used asynchronous update of the intermediate results. Thus, we use the time to load the result from the first frame in the first iteration to update the intermediate results as a representative of the update time. Time needs for rendering an iterative task of an animation with temporal perturbations requires about 10% more than in a frame-by-frame manner. In addition, rendering with temporal perturbations requires roughly additional 60 s to update the intermediate result. In each task we allocate extra 20 frames around the center frame of the task to record the majority of perturbed samples to avoid frequently writing off-center sample records to the disk and save disk space for the off-center records, and this requires extra 200 MB memory. However, when a perturbed sample falls outside this range, the task writes a off-center record of time, pixel position, and radiance into the disk. At the end of the task, the center frames and out-of-core records are all sent back to the system. The rendering task with temporal perturbations generates additional 48 $\sim$ 54.2 times more data than in the frame-by-frame manner.

In Fig. 4, the brightness around the edge of the caustics region rendered in a frame-by-frame manner changes drastically in consecutive frames. In comparison, the caustics region in the animation rendered with time perturbations looks smooth and has similar shape and brightness in consecutive frames. In accompanied animations, the animation rendered in frame-by-frame manner has seriously flickering artifacts in the caustics regions. We demonstrate the strength of temporal exploration to create tem-

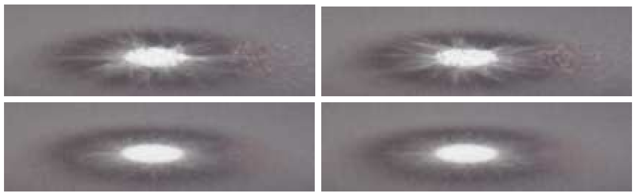

Figure 4: *The top row of images is coming from the cropped images of the caustics region of the 27th and 28th frame of the Cornell Box animation rendered in a frame-by-frame manner. The second row of images is coming from the cropped images of the caustics region of the 27th and 28th frame of the Cornell Box animation rendered with our time perturbations. Our results generate a smoother and temporally consistent caustics region in these two consecutive frames and the animation also has a smoother and more consistent caustics region. The caustics region rendered in frame-by-frame manner is noisy and looks different in shape for consecutive frames. As a result the animation has noisy and twisting caustics regions.*

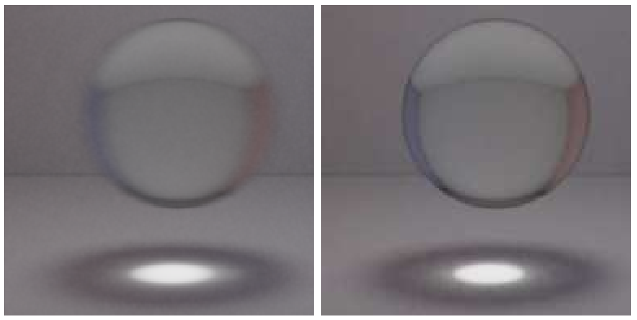

Figure 5: *These are the cropped image for 5th frames in the Cornell box scene. The left is rendered with a finite shutter open period and the right is rendered with a delta period. We can observe that the motion blur is added naturally. We can see that the caustics regions, the glass ball, and the shadow regions contain blurred areas around the edge.*

porally consistent lighting by intentionally using a less smooth caustics region. Later in CB2, we use another set of parameters to generate a smoother animation of the same CB animated scene.

Fig. 6 shows the 60-th frame of CB2, room, chess board, and basement scenes rendered with temporal perturbation. When we check the accompanied animation results, we found that the temporal artifacts in animations rendered with temporal perturbations are far less than the results without the temporal perturbations. In addition, the algorithm with temporal perturbations can generate a converged animation faster than without temporal perturbations.

For comparison reasons, we rendered a frame in a transient moment. However, our system can easily be used to render frames with motion blur by setting a non-zero shutter-open periods because our time perturbation can easily perturb a path to another path in any moment of time in the entire animation. Fig. 5 demonstrate that motion blur can be added into a CB animation.

5. Discussion

Time perturbations find correlated paths across different frames and our system needs to store the contribution of each correlated path in a set of frames. Theoretically it is possible for our algorithm to explore all temporal coherence in the entire animation but practically the system has a limited amount of memory. Therefore, we made a trade-off between the extent of temporal exploration and the amount of memory and disk usage and the time for updating each process. We use a set of temporal perturbation radii and the number of perturbations in each Markov Chain to get the samples distributed roughly in a 20-frame radius from the center of the starting frame to limit memory usage for center samples and disk space of out-of-center samples. Although this limits the extent of temporal exploration when the perturbed distance from the starting frame grows, the failure rate of temporal perturbations grows. This should have slight effect on the rate of generating temporally consistent animations. A more efficient memory and disk usage and update scheme or a better task distribution scheme is wanted in the future to avoid the limitation of temporal exploration radii and to relieve the burden of memory and disk space requirement and the time for updating task-based data.

Currently we use visual inspection to check the rendering animations. This is time consuming and less preferred. There are several animation measurement metrics developed for video compression. The main issue for video compression is that data loss causes the artifact of blockiness. However, the main issue for MC algorithms is that noise pops up randomly in the scene and causes serious temporal artifacts. A proper metric must take into account this independent disturbance. Such a MC-based perceptual metric can help us evaluate the rendering results and further adjust our kernels to concentrate on perceptually important regions.

In the current implementation we used key-framed rigid transformations to animate entities in the scene. It is obvious that our algorithm can be easily adjusted to render objects with key-framed properties such as material, light intensity, and other information. In addition, it is easy to extend the implementation to include skin-skeleton animation and morph animation. For skin skeletons each vertex's position is related to the skeleton position and orientation at the moment. We can use the intersection point's parameters (u,v) and three vertices' positions to compute the position of corresponding positions for different moments of time. Similarly, the morph animation also uses (u,v) to compute the corresponding vertex position at different moments. This mechanism allows us to compute the same corresponding position for each intersection point for different moments of time.

Our algorithm made a trade-off between the rendering speed and the image quality by choosing the averaging path energy to determine the number of Markov Chains for each seed path. As a result, the image quality of the dark regions is generally more noisy. When rendering a static image, the dark regions are perceptually less important and, therefore, the perceptual noise is hard to notice. However, since our perception is sensitive to temporal inconsistency, the noisy dark regions become perceptually important when rendering an animation. Although temporal exploration can reduce the variance in dark regions, the variance in dark regions is still perceptible because the number of Markov Chains distributed to the dark regions is low. Kelemen et al. [KSKAC02] used multiple importance framework to combine the independent

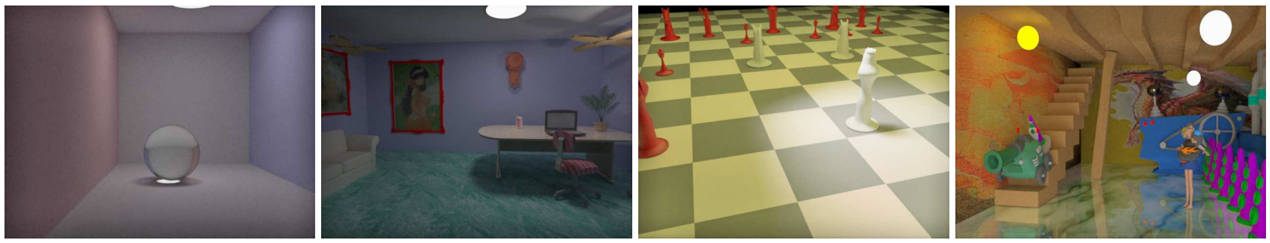

Figure 6: *The images from left to right are the 60-th frame generated from CB, room, chess board, and basement scenes using PMC-ER algorithms with temporal perturbations.*

Scene	Method	$N^{host}_{iteration}$	$N_{uniform}$	$N_{variance}$	$S_{caustics}$	Clt(s)	Dt(s)	Dk (M)	Mk (M)	Err
CB1	frame-by-frame	1	4	0	0	2532	1	5	256	1.93e-2
	time perturbation					2788	59	240	453	1.82e-2
CB2	frame-by-frame	1	4	36864000	0	3213	1	5	266	1.5e-2
	time perturbation					3559	62	243	463	6.29e-3
Room	frame-by-frame	4	4	36864000	1.25	1743	1	5	606	3.94e-2
	time perturbation					1844	65	263	803	8.37e-3
Chess	frame-by-frame	8	4	36864000	1	2093	1	5	366	5.98e-2
	time perturbation					2244	63	253	563	4.51e-1
Basement	frame-by-frame	8	4	36864000	1	2743	1	5	746	2.61e-1
	time perturbation					2954	68	271	943	2.08e-2

Table 1: *Measurements comparing temporal exploration with frame-by-frame. In all cases, we used a population size of 5000, three spatial perturbations having radii: 5, 10, and 50 pixels and two temporal perturbations having radii: 0.066 and 0.165 s. In the inner loop z, we perturbed each member 20 times and eliminated 40% of the population based on its remaining energy and regenerated new paths to maintain constant number of members in the population. In the preprocess we used 16 samples per pixel (SPPs) for estimating $\tilde{E}$, e_d, R^k_{G2C}, and γ^k. In the rendering process we chose $N^{host}_{iteration}$ as the total number of iterations, $N_{uniform}$ as the number of sample per pixel for stratified regeneration, and $N_{variance}$ as the size of the variance-regeneration samples used in an iteration. We computed the number of caustics-regeneration in each frame as $S_{caustics} \times N^k_{expect}$. Thus, we use $(N^{host}_{iteration}, N_{uniform}, N_{variance}, S_{caustics})$ to specify the parameters used to render the animation scenes. The seventh column is the time needed to finish the first frame in the first iteration in each animation. The eighth column is the time required to update the data from the first frame in the first iteration. The ninth column is the disk usage for the data when finishing the first frame in the first iteration. The tenth column is the memory consumption in computing the task of the first frame in the first iteration. The eleventh column is the perceptual error defined in [Fan06] for the entire animation.*

path samples and correlated path samples to relieve this problem. This inspires us to think about taking advantage of the fact that our algorithm generates a set of independent sample paths before the energy redistribution step. We would like to develop a hybrid algorithm combining balance transfer and equal deposition. In the algorithm balance transfer is designed to explore the low-energy paths and equal deposition is designed to spending more computation on exploring the high-energy paths. This should be able to relieve the problem of the relatively noisy dark regions. In addition, they also purposed to map the creation and mutation of paths to a high-dimension uniform random number cube to increase the success rate of mutations for enhancing rendering efficiency. Our time and lens perturbation is designed to use small perturbation on the time and image plane domain to increase the perturbation success rate. A comparison in the success rate between local perturbations and uniform-cube perturbations is needed in the future. We also would like to explore the possibility of combining the uniform-cube perturbation methods with our adaptation algorithm and spatial and temporal perturbation method by systematically perturbing the random variables used to control the perturbation of time and the perturbation of lens sub-paths to further increase the success rate of perturbation.

Although we demonstrate the strength of temporal exploration based on PMC-ER, the temporal exploration and local adjustment is easily adapted into the MLT and ERPT frameworks. The new time perturbation method can be added into the choice of the mutation methods in the ERPT and MLT algorithms with the implementation of the image sweep. Parallel rendering the animation with all regeneration methods and locally adjusting parameters is important to get a converged animation quickly. Since the ERPT algorithm has a similar parallel energy-distribution structure, we can use the same task distribution framework. Each task contains the paths starting from the same frame. In the preprocess we can estimate γ^k and R^k_{G2C} for each frame with $\tilde{\mathbf{E}}$ and e_d. The computation of energy deposition ratio discussed in Sec. 3.3 can be directly applied. However, applying parallel local adjustment to MLT is different. We should first generate a seed path per frame and then a pool of replacement paths consisting of variance-regeneration and caustics-regeneration paths of the same frame. Then, during the mutation process, we can replace the current seed path with one of the paths from the pool similar to lens replacement mutation. The acceptance probability can be computed accordingly to decide whether the exploration path switches to the new replacement path. This should achieve a similar result as presented in our demonstration.

6. Conclusion

In this paper our animation system is built on the PMC-ER framework to demonstrate the enhancement in perceptually rendering efficiency by exploring the temporal coherence among all light transport paths in the entire animation. Our system is based on the Condor system, which allows us to iteratively render an animation in a parallel manner. Our system demonstrates the ability of adjusting rendering parameters locally in each iterative task without introducing bias into the final result. In addition, motion blur can be naturally added into each frame when using our temporal perturbations. At the end a short discussion of applying the temporal perturbations and parallel rendering manner to other MCMC algorithms is described. The results show that exploration of all kinds of coherence among path integrals is definitely the correct direction to enhance rendering efficiency.

References

[APSS04] Ashikhmin M., Premoze S., Shirley P., Smits B.: A variance analysis of the metropolis light transport algorithm. *Computer and Graphics 25*, 2 (2004), 287–294.

[BS96] Besuievsky G., Sbert M.: The multi-frame lighting method - a Monte-Carlo based solution for radiosity in dynamic environments. In *Rendering Techniques '96* (1996), pp. 185–194.

[BSH02] Bekaert P., Sbert M., Halton J.: Accelerating path tracing by re-using paths. In *Rendering Techniques '02 (Proceedings of the Thirteenth Eurographics Workshop on Rendering)* (2002), pp. 125–134.

[CJ02] Cammarano M., Jensen H. W.: Time dependent photon mapping. In *Rendering Techniques '02 (Proceedings of the Thirteenth Eurographics Workshop on Rendering)* (2002).

[CTE05] Cline D., Talbot J., Egbert P.: Energy redistribution path tracing. In *SIGGRAPH '05* (2005), pp. 1186–1195.

[DBMS02] Dmitriev K., Brabec S., Myszkowski K., Seidel H.-P.: Interactive Global Illumination Using Selective Photon Tracing. In *Proc. of the 13th Eurographics Workshop on Rendering* (2002), pp. 25–36.

[DPSaB06] Dutre P., Peter Shirley and K. B., Bekaert P.: *Advanced Global Illumination*. A K Peters Ltd, 2006.

[Fan06] Fan S.: *Sequential Monte Carlo Methods for Physically Based Rendering*. PhD thesis, University of Wisconsin-Madison, 2006.

[GBP07] Gautron P., Bouatouch K., Pattanaik S.: Temporal radiance caching. In *IEEE Transactions on Visualization and Computer Graphics* (2007), vol. 13, IEEE Computer Society, pp. 891–901.

[GDH06] Ghosh A., Doucet A., Heidrich W.: Sequential sampling for dynamic environment map illumination. In *Proc. Eurographics Symposium on Rendering* (2006), pp. 115–126.

[HBS03] Havran V., Bittner J., Seidel H.-P.: Exploiting temporal coherence in ray casted walkthroughs. In *SCCG '03: Proceedings of the 19th spring conference on Computer graphics* (New York, NY, USA, 2003), ACM, pp. 149–155.

[HDMS03] Havran V., Damez C., Myszkowski K., Seidel H.-P.: An efficient spatio-temporal architecture for animation rendering. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Sketches & Applications* (New York, NY, USA, 2003), ACM, pp. 1–1.

[KSKAC02] Kelemen C., Szirmay-Kalos L., Antal G., Csonka F.: A simple and robust mutation strategy for the metropolis light transport algorithm. vol. 21, pp. 531–540.

[LFCD07] Lai Y., Fan S., Chenney S., Dyer C.: Photorealistic image rendering with population monte carlo energy redistribution. In *Eurographics Symposium on Rendering* (2007), pp. 287–296.

[LLZD08] Lai Y., Liu F., Zhang L., Dyer C.: Efficient schemes for monte carlo markov chain algorithms in global illumination. In *International Symposium on Visual Computing* (2008).

[MA06] Meyer M., Anderson J.: Statistical acceleration for animated global illumination. *ACM Trans. Graph. 25*, 3 (2006), 1075–1080.

[MFSSk06] Mendez-Feliu A., Sbert M., Szirmay-kalo L.: Reusing frames in camera animation. *Journal of WSCG 14*, 3 (2006).

[MTAS01] Myszkowski K., Tawara T., Akamine H., Seidel H.-P.: Perception-guided global illumination solution for animation rendering. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM Press, pp. 221–230.

[PH04] Pharr M., Humphreys G.: *Physically Based Rendering from Theory to Implementation*. Morgan Kaufmann, 2004.

[SC04] Sbert M., Castro F.: Reuse of paths in final gathering step with moving light sources. In *International Conference on Computational Science 2004* (2004), pp. 189–196.

[SCH04] Sbert M., Castro F., Halton J.: Reuse of paths in light source animation. In *CGI '04: Proceedings of the Computer Graphics International* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 532–535.

[SKDM05] Smyk M., Kinuwaki S., Durikovic R., Myszkowski K.: Temporally coherence irradiance caching for high quality animiation rendering. In *Computer Graphics Forum (Proceedings Eurographics 2005)* (2005).

[SSSK04] Sbert M., Sz'ecsi L., Szirmay-Kalos L.: Real-time light animation. In *Computer Graphics Forum (Proceedings Eurographics 2004)* (2004), vol. 23(3), pp. 291–300.

[TMS04] Tawara T., Myszkowski K., Seidel H.-P.: Exploiting temporal coherence in final gathering for dynamic scenes. In *CGI '04: Proceedings of the Computer Graphics International (CGI'04)* (2004), IEEE Computer Society, pp. 110–119.

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

[VG97] Veach E., Guibas L. J.: Metropolis light transport. In *SIGGRAPH '97* (1997), pp. 65–76.

[VMKK00] VOLEVICH V., MYSZKOWSKI K., KHODULEV A., KOPYLOV E.: Using the visual differences predictor to improve performance of progressive global illumination computations. vol. 19, ACM, pp. 122–161.

[WMM*04] WEBER M., MILCH M., MYSZKOWSKI K., DMITRIEV K., ROKITA P., SEIDEL H.-P.: Spatio-temporal photon density estimation using bilateral filtering. In *CGI '04: Proceedings of the Computer Graphics International (CGI'04)* (2004), IEEE Computer Society, pp. 120–127.