Computer Sciences Department

Population Monte Carlo Path Tracing

Yu-Chi Lai
Charles Dyer

Technical Report #1614

September 2007

UNIVERSITY OF WISCONSIN MADISON

Population Monte Carlo Path Tracing

Yu-Chi Lai
University of Wisconsin at Madison
Graphics-Vision Lab
1210 W. Dayton St., Madison, WI53706
yu-chi@cs.wisc.edu

Charles Dyer
University of Wisconsin at Madison
Graphics-Vision Lab
1210 W. Dayton St., Madison, WI53706
dyer@cs.wisc.edu

Abstract

We present a novel global illumination algorithm which distributes more image samples on regions with perceptually high variance. Our algorithm iterates on a population of pixel positions used to estimate the intensity of each pixel in the image. A member kernel function, which automatically adapts to approximate the target ditribution by using the information collected in previous iterations, is responsible for proposing a new sample position from the current one during the mutation process. The kernel function is designed to explore a proper area around the population sample to reduce the local variance. The resampling process eliminates samples located in the low-variance or well-explored regions and generates new samples to achieve ergocity. New samples are generated by considering two factors: the perceptual variance and the stratification of the sample distributions on the image plane. Our results show that the visual quality of the rendered image can be improved by exploring the correlated information among image samples.

1. Introduction

To generate images that are close to reality has become more and more important in several different applications. Monte Carlo (MC) integratons provide us a general solution to solve integration problems involved in rendering. However, the efficiency of Monte Carlo methods is still the main concern when applying it in practice. Generally, different regions may require different numbers of samples to render a converged image. In standard MC estimators, pixel samples are uniformly and evenly distributed on the image plane. This is inefficient because low-variance regions only need a small number of samples, and high-variance regions may need a large number of samples in order to generate a converged image. As a result, in order to guarantee generating a converged image, MC needs to use a large number of samples among all regions even in low-variance regions. If we can shift those extra samples in low-variance regions to high-variance regions, we can improve the rendering quality. Our Population Monte Carlo path tracing (PMC-PT) algorithm automatically distribute more image samples to explores high variance regions .

Our algorithm iterates on a population of pixel positions on the image plane. The initial population of samples are evenly distributed on the image plane. Any information available in the previous iterations can be used to adapt member *kernel functions* that produce a new population based on the current population. The resampling process eliminates part of the population samples and regenerates new samples to achieve ergocity. We carefully design the resampling process to eliminate the well-explored samples from the current population and to generate new samples by considering two factors: the perceptually-weighted variance among the samples in each pixel and the need to stratefiedly explore the image plane. As a result, new regenerated samples are designed to locate in the perceptually important areas or to distribute on the image plane in an even manner. The procedure is then iterated: sample, iterate, resample, adapt, iterate, resample .... The result is a self-tuning unbiased algorithm which can locally explore the important visual areas on the image plane. All pixel positions generated by mutation and regeneration are used to estimate the intensity of each pixel by using a general MC ray tracing algortihm such as path tracing and bidirectional path tracing in order to generate an image. In our implementation, we use a general path tracing algorithm.

Our contribution is a new rendering algorithm, **PMC Path Tracing**(PMC-PT), based on the PMC framework. This algorithm adapts the kernel functions to determine the radius of local exploration with the information collected in previous iterations. In addition, the resampling process distributes the new samples over the entire image plane according to the perceptual importance and stratification to achieve ergocity. Samples kept during the elimination process are located in regions with high variance.

The remainder of this paper is organized as follows: section 2 reviews a number of works related to this algorithm. Section 3 presents the generic PMC frame work. Section 4 present the PMC-PT in detail. Section 5 shows the results generated by this algorithm. Section 6 discusses the limitation and relation to the existing algorithm. Finally, section 7 gives the conclusion of our algorithms.

2 Related Work

Currently, most global illumination algorithms are based on ray tracing and Monte Carlo integration. There exist two categories: unbiased methods such as [**?**, **?**, **?**]; and biased methods such as [**?**, **?**, **?**]. Interested readers can refer to Pharr and Humphreys [**?**] for an overview of Monte Carlo rendering algorithms. Here we focus on three specific areas related to our work: adaptive image-plane sampling, perceptual metrics, and sample reuse.

Typically, adaptive image-plane algorithms initially render the image with a small number of samples per pixel. The initial image is analyzed to label pixels as adaquately sampled or in need of further refinement. Then, the algorithms iterate on pixels requiring more samples [**?**, **?**, **?**, **?**, **?**, **?**]. However, the label on the pixels based on an initial samples introduces bias [**?**] into the final result, which is a problem when physically accurate renderings are required. We carefully design the sample distribution probability with respect to the perceptual importance and at the same time, avoid bias during the resampling process.

Many metrics have been proposed for the test to trigger additional sampling. Lee et al. [**?**] used a sample variance based metric. Dippé and Wold [**?**] estimated the change in error as sample counts increase. Painter and Sloan [**?**] and Purgathofer [**?**] used a confidence interval test, which Tamstorf and Jensen [**?**] extended to account for the tone operator. Mitchell [**?**] proposed a contrast based criteria because humans are more sensitive to contrast than to absolute brightness, and Schlick [**?**] included stratification into an algorithm that used contrast as its metric. Bolin and Meyer [**?**], Ramasubramanian et al. [**?**] and Farrugia and Péroche [**?**] used models for human visual perception, of which we use a variant. Most recently, Rigau et al. [**?**, **?**] introduced entropy-based metrics.

Our PMC-PT algorithm uses the adaptation of the member kernel function to locally explore perceptual important regions and uses resampling to achieve ergocity and exploration of high percetual variance regions. In addition, it is unbiased.

3 D-Kernel Population Monte Carlo

The Population Monte Carlo algorithm [**?**] is an adaptive algorithm that calibrates the proposed distribution to the target distribution at iteration by learning from the performance of the previous proposal distributions. The generic D-Kernel PMC sampling algorithm [**?**] which is an evolution of PMC. Our algorithm, an adaptation of the generic D-Kernel PMC algorithm, is stated in Figure 1. Our algorithm adapts the kernel function for each population path instead of a single kernel function for the entire population.

```
1  generate the initial population, t = 0
2  for t = 1, ..., T
3     adapt K_i^(t)(x_i^(t) | X_i^(t-1))
4     for i = 1, ..., N
5        generate X_i^(t) ~ K_i^(t)(x_i^(t) | X_i^(t-1))
6        w_i^(t) = π(X_i^(t)) / K_i^(t)(x_i^(t) | X_i^(t-1))
7     resampling process: elimination and regeneration
```

Figure 1. The generic D-Kernel Population Monte Carlo algorihtm.

Assume we have a population of samples denoted by $\left\{X_1^{(t)}, \ldots, X_N^{(t)}\right\}$, where t is the iteration number and N is the population size, and we wish to sample according to the target distribution, $\pi(x)$, where is $f(x) = \pi(x)h(x)$ in order to evaluate $\int_{\mathcal{D}} f(x)dx$. We start the algorithm by creating the initial population with any method that can generate these samples provided that any sample with non-zero probability under $\pi(x)$ can be generated, and the probability of doing so is known. The outer loop is responsible for adapting a member *kernel function*, $K_i^{(t)}(x_i^{(t)}|X_i^{(t-1)})$, for each member in the population, (line 3) using information from the previous iterations. The kernel fuction is used to generate a new population sample, given the current one. The inner loop takes an existing sample, $X_i^{(t-1)}$, as input and produces a candidate new sample, $X_i^{(t)}$, as output (line 5). The resampling step in line 7 consists of two steps: elimination and regeneration. It is designed to eliminate the samples with low contribution to the final result and to explore new unexplored regions. The weight computed for each sample, $w_i^{(t)}$, is essentially its importance weight. At any given iteration, an estimator of the integral can be computed and is unbiased for $\pi(h)$:

$$\tilde{f}(x) = \tilde{\pi}(h) \quad = \quad \frac{1}{N}\sum_{i=1}^{N} w_i^{(t)} h(X_i^{(t)}) \qquad (1)$$

Before applying PMC to rendering problems, several decisions must be made:

- Decide the sampling domain and population size.
- Define kernel functions and their adaption criteria.

- Choose the techniques for sampling from the kernel functions and resampling step.

The following sections describe the application of this framework by mutating the general path tracing algorithm. This algorithm uses kernel functions with metrics to accumulate, eliminate, and regenerate samples. Then, we conclude with a general discussion on PMC for rendering problems.

4 Population Monte Carlo Path Tracing (PMC-PT)

To render an image, the intensity, $I_{i,j}$, of each pixel must be computed using Equation 2. MC in Equation 3 uses a set of path samples, $\left\{\tilde{\mathbf{X}}_1, \ldots, \tilde{\mathbf{X}}_N\right\}$, sampled from an importance function $p(\tilde{\mathbf{x}})$ to estimates the intensity, $I_{i,j}$.

$$I_{i,j} = \int_{\mathcal{I}} W_{i,j}(\tilde{\mathbf{X}}) L(\tilde{\mathbf{X}}) d\mathbf{u}(\tilde{\mathbf{X}}) \quad (2)$$

$$\hat{I}_{i,j} = \frac{1}{n}\sum_{k=1}^{N} \frac{W_{i,j}(\tilde{\mathbf{X}}_k) L(\tilde{\mathbf{X}}_k)}{p(\tilde{\mathbf{X}}_k)} = \frac{1}{n}\sum_{k=1}^{N} W_{i,j}(\tilde{\mathbf{X}}_k)\mathcal{E}[L(\tilde{\mathbf{X}}_k)] \quad (3)$$

where $\mathcal{I}$ is the image plane, $W_{i,j}(\tilde{\mathbf{X}})$ is the measurement function for pixel (i,j) – non-zero if the lens edge, $\mathbf{x}_{n-2}\mathbf{x}_{n-1}$ passes through the support of the reconstruction filter at (i,j) where n is the total number of vertices in the path – and $L(\tilde{\mathbf{X}})$ is the radiance bringing on the path, $\tilde{\mathbf{X}}$. MC can prove that $\lim_{N\to\infty} \hat{I}_{i,j} = I_{i,j}$.

The only difference among all pixels is the term of $W_{i,j}(\tilde{\mathbf{X}}_k)$. Provided $p(\tilde{\mathbf{X}})$ is known in each path which passes through the valid image plane, the global nature of $p(\tilde{\mathbf{X}})$ is not important. Thus, the rendering equation can be transformed to evaluate the expected radiance, $\mathcal{E}[L(\tilde{\mathbf{X}})]$, carried by each sampled path and then accumulate this radiance in pixels whose support of reconstruction includes the path-passing pixel position. At the final step of rendering, the accumulation in each pixel is averaged by the total number of samples dropped in the support of the pixel. Therefore, the unbiasedness can be achieved under two conditions: first, the estimator of expected path radiance is unbiased; and second, there are infinite samples falling in the support of each pixel's recontruction filter as the total number of samples goes to infinity.

When applying Equation 3 to render an image, the sample distribution on the image plane is uniform. However, the complexity of lighting varies from region to region on the image. Thus, each region requires diffrent number of samples to achieve a converged values. It is inefficient to put a large number of samples on regions with low variance due to the need of regions with high variance. PMC-PT is designed to distribute more samples for further exploration of regions with high variance. The variance of the illuminance among a population sample and its kernel-proposed descendants is used to determine the need of exploration. The higher the variance is, the higher the need of samples is. In this section, we first discuss the algorithm itself following with the detailed discussion of the adaptation and resampling process.

4.1 PMC-PT Algorithm

Figure 2 shows the steps using PMC-PT algorithm to render an image. In the preprocess phase, the algorithm first generates a pool of stratified pixel positions used to distribute the population samples evenly on the image plane. This pool is asked to give a population of initial samples and to generate new stratified replacement samples during the resampling process in each iteration in order to guarantee that every pixel has the chance to be explored.

```
1   generate a pool of stratified pixel positions
2   generate initial population of samples in t = 0
3   for s = 1, ..., T
4       determine α_i^(s)
5       for i = 1, ..., n
6           for t = 1, ..., T_R
7               generate X_i^(t) ~ K_i^(s)(x^(t) | X_i^(t-1))
8           Compute w_i^(s) = σ_i^2 / N_used
9       resample: elimination and regeneration
10      adapt β_{x,y}^(s)
```

Figure 2. The PMC-PT iteration loop. T_R is the number of kernel iterations per resample step. σ_i^2 computes the variance of the ith population sample and all mutated descendent samples after it has been regenerated and this value is used to calculate the weight, $w_i^{(t)}$, for elimination, and N_{used} is the number of resampling loop this sample has been used since it has been regenerated.

The resampling step in line 9 is designed to cull candidate samples located in perceptually low-variance regions and keep samples located in perceputally high-variance regions. It takes the candidate population, $\left\{\mathbf{X}_1^{(s)}, \ldots, \mathbf{X}_n^{(s)}\right\}$, and produces a new population ready for the next iteration. The kernel adaption (lines 4 and 10) need not be done on

every iteration. Our examples demonstrate such cases. After exploring several values for T_R, we found a wide range of values to be effective. The optimal value depends on the population size and the relative cost of kernel perturbations compared to resampling. The aim of adding new samples in the resampling process is to eliminate the possibility of overexploring a few regions with very high variance during the mutating process in order to guarantee the unbiasedness of our algorithm. Adding new samples in this way does not add bias, because neither the mutated population nor the new samples are biased, so their union is not biased. After deciding the new pixel position either in regeneration or in mutation process, we use a path tracing algorithm to evaluate the expected radiance along the ray from eye to the pixel position, $I_L = \mathcal{E}[L(PT(\mathbf{X})]$ where $PT(\mathbf{X})$ generate a valid path, $\bar{\mathbf{X}}$, passing through the pixel position, $\mathbf{X}$, by using the path tracing algorithm. We only needs the probability, $p(PT(\mathbf{X}))$, to evaluate the expected path radiance, I_L, $T(\mathbf{x}|\mathbf{X} : d)$ is not important and is only used to distribute the samples.

4.2 Kernel Functions

The kernel function for each population member is a conditional kernel, $K_i^{(s)}(\mathbf{x}^{(t)}|\mathbf{X}_i^{(t-1)})$, that generates a sample position i in iteration t, $\mathbf{X}_i^{(t)}$, given sample i in iteration $t-1$, $\mathbf{X}_i^{(t-1)}$. we use a mixture distribution:

$$K_i^{(s)}(\mathbf{x}_i^{(t)}|\mathbf{X}_i^{(t-1)}) = \sum_{d_j} \alpha_{i,d_j}^{(s)} T(\mathbf{x}^{(t)}|\mathbf{X}^{(t-1)} : d_j) \qquad (4)$$

Each component, $T(\mathbf{x}|\mathbf{X} : d)$, mutates an existing sample to generate a new one for exploration of the image space according to the perturbing radius, d. There is a set of perturbing radiuses, d_i, given as parameters to the algorithm and each is good for different occasions. α's values are used to choose a radius for the current sample path and their values are adapted through iterations for each path. Once we choose a d_i, the perburbation takes the existing pixel position and moves to a new pixel positions uniformly sampled within a disk of radius, d, a parameter of the kernel component.

4.3 Resampling

The resampling step in this algorithm achieves three purposes: samples that locate in regions with higher variance are carried forward to the next round, it provides an opportunity to add some completely new samples into the population for exploring unexplored or perceptually high-variance regions, and the information about which perturbations are chosen inside the inner loop guides the adaption of the kernel functions. The following sections give the detail of three steps

4.3.1 Eliminate samples from the population

The weight, $w_i^{(s)}$, for each sample is used to determine the probability of survival of the path during the elimination phase. At each inner loop, we tag the expected radiance computed by the PT algorithm , $I_{L_i}^{(t)}$, for each member and then in elimination process, we evaluate the variance of these illuminances, σ_i^2. The higher variance a member has, the higher the chance it is kept in the population. This can achieve the goal of continuing to explore high-variance regions. However, we would like to avoid over-exploring the samples located in high-variance regions in order to give more chance for other samples. Thus, we weigh the variance down by the total iterations which it has being used for perturbations. Thus, the final weight is $w_i^{(s)} = \sigma_i^2 / N_{used}$

4.3.2 Regenerate new samples into the population

Regeneration is to maintain the constant number of samples in the population. It also gives us the chance to decide where we would like to explore in the next iterations. Our algorithm considers two aspects: the stratification of pixel positions and the perceptual variance of the intermediate result.

1. **Stratification**:

   Pharr [**?**] demostrates how important sampling evenly on the image plane is in reducing the variance. Thus, in the preprocess phase, we compute the total stratified number of pixel positions needed for the entire process.Then a pool of stratified pixel positions is generated according to that number. During the regeneration process, we keep asking the pool to give us the next unused stratified pixel position. This is also used to guarantee that every pixel gain its chance to be explored to achieve the second requirement of unbiasedness.

2. **Perceptual variance**:

   In order to generate new sample paths in regions with high perceptual variance. We use the value $\beta_{i,j}^{(s)}$ to indicate the degree of requirement for more samples at pixel (i, j). Pixels that require further exploration should have higher $\beta_{i,j}^{(s)}$. An appropriate criteria assigns $\beta_{i,j}^{(t)}$ proportional to an estimate of the perceptually-weighted variance at each pixel. The algorithm tracks the sample variance in illuminance

seen among samples that contribute to each pixel. To account for perception, the result is divided by the threshold-versus-intensity function $tvi(I)$ introduced by Ferweda et al. [**?**].

$$\begin{aligned} \beta'_{i,j} &= \frac{\overline{\sigma}^2_{i,j}}{tvi(I_{i,j})} \\ \beta^{(s)}_{i,j} &= \frac{\beta'_{i,j}}{\sum_{i',j' \in IP} \beta'_{(i',j')}} \end{aligned}$$

To get a pixel position, we first choose a pixel,(i, j) according to the weight of $\beta^{(s)}_{i,j}$ and then perturb the position by one pixel distance in each direction.

4.3.3 Adapt α's Values to Propose a New Path

When exploring a region with little change in lighting such as the diffuse wall far from a light, we would like to expand the exploring area, i.e. have higher probability to choose a larger perturbation radius. On the other hand, when exploring a region with complex change in lighting, such as a glossy surface with a light located near the reflection glare direction, we would like to shrink the exploring area, that is, have a higher probability to choose a smaller perturbation radius.

When a new sample is generated in the regeneration process, the $\alpha^{(s)}_{i,k}$ is set as a constant probability for each component which allows us to uniformly choose any of the perturbations. Since the goal is to decide the exploration according to the lighting detail. After initialization, the expected illuminance, $I^{(t)}_{L_i}$, of each new mutated path was tagged with the kernel mixture component that generated it and its index in the population, i. At adaptation step, we uses the tagged illuminance to evaluate the perceptual variance for each component of each member in the population to adjust the α's values according to:

$$\begin{aligned} \alpha'_{i,k} &= \begin{cases} 0 & if\ \overline{\sigma}_{i,k} = 0 \\ \frac{1}{\overline{\sigma}^2_{i,k}} & if\ \overline{\sigma}_{i,k} \neq 0 \end{cases} \\ \alpha^{(s)}_{i,k} &= \epsilon + \frac{(1-\epsilon)\alpha'_{i,k}}{\sum_{k'=1}^{n} \alpha'_{(i,k')}} \end{aligned}$$

where $\overline{\sigma}^2_{i,k}$ is the variance of a set of illuminances tagged with the k-th mixture component for i-th member in the population.

5 Results

We compared PMC-PT with the path tracing algorithm on two Cornell Box scenes and a room scene with roughly the same number of rays shooting out from the camera. One Cornell Box scene is with a glass ball and other surfaces such as walls, being Lambertian; and the other Cornell Box scene is with a glass ball in the front, a mirror ball in the back, one mirror surfaces on the right side of the box, and the remaining surfaces being Lambertian. The room scene contains several complex objects and a glossy table to demostrate the usage of the algorithm for a complex scene. In all three cases we used a population size of 5000. There are three pertubation radiuses: 5, 10, and 50 pixels, respectively. In each step inside the inner loop, each member generates 16 resampling perturbations, and 40% of the population is eliminated and regenerated. 50% of regenerated samples are created using the stratification mechanism, and 50% are generated using the perceptual variance mech-

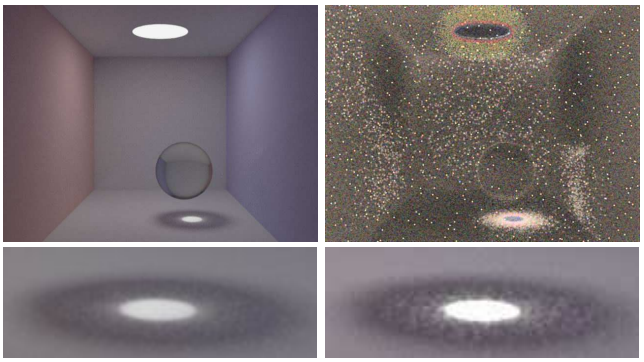

Figure 3. A Cornell Box image computed using PMC-PT with 250 iterations on the left, and the image represents the mutation strategy (red represents perturbation of radius 5 pixels, green represents pertubation of radius 10, and blue represents the perturbation of 50 pixels) used during the process on the right in the first row. To compute a converged value at the caustic part is difficult. Hence, we show the cropped image of the caustic region. The left image in the second row is the cropped image of the image rendered by our algorithm. The right image is the cropped image of an image computed with a PT algorithm with 64 spps. The strategy image shows that our algorithm will adjust the perturbed radius according to the lighting change and physical edges. The image also shows that our algorithm will put more samples on the high perceptual variance regions such as the light's edge and the caustic and shadow regions under the glass ball.

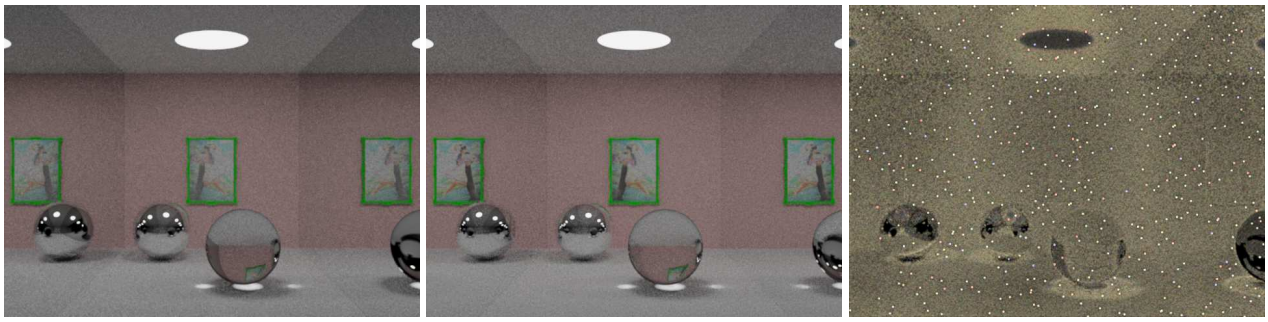

Figure 4. Another Cornell Box image with complex pattern of specular light transport paths. The left image is computed using PMC-PT with 1000 iterations; the middle image is generated using a PT algorithm by 256 spps; the right image represents the mutation strategy used in our PMC-PT algortihms. Generally, our algorithm gets a better performance at the caustic and shadow regions. The strategy image shows that our algorithm almost evenly explores all caustic and shadow regions, and it adjusts to shorter radiuses which is showed as a yellow color but the paths do not stay in the population to adjust to the shortest radius which is showed as a red color like the result of Figure 3

anism. The Cornell box scenes were rendered at 640×480 resolution and the room scene were rendered at 720×405 resolution. The first Cornell Box scene is rendered with 250 iterations and the α's and β's values are updated every 100 iterations. The second one is rendered with 1000 iterations and the α's and β's values are updated every 200 iterations. The room scene is rendered with 3400 iterations and the α's and β's values are updated every 200 iterations.

The images (Figure 3) demonstrate that PMC-PT expends more effort on the difficult regions – the region in the ceiling near the light, the caustic and shadow region under the glass ball – and hence has a lower variance in those regions, at the expense of a slightly higher variance in other parts of the image. This is a recurring property of the PMC-PT algorithm: PMC produces a more even distribution of noise, with lower noise levels overall but higher in some parts of the image that are over-sampled with non-adaptive techniques. The strategy image shows that our adaptation strategy automatically adjusts the perturbed radius near the highly changing lighting boundary such as the caustic and shadow region or the physical edges, such as the corners. These regions also generate higher perceptual variance. As a result, the algorithm puts more samples on these regions showed as brighter illuminace in the strategy image. The overall result is a more perceptually pleasant image compared to the corresponding image rendered by the PT algorithm using 64 spps, which is roughly the same total number of sample paths shooting from the eye.

The second Cornell Box contains complex specular tranport paths and has several caustic regions on the image. The images (Figure 4) demonstrate that PMC-PT evenly explores these difficult regions by distributing more samples on these areas. However, the direct lighting change areas, such as the caustic regions under the glass ball and the region in the ceiling near the light, have higher perceptual variance than their corresponding reflection and get more samples. Since high proportion of the population has a high perceptual variance, there is a smaller chance to stay in the valid population to adapt to the shortest radius which is showed in the strategy image as a red color. However, our algorithms still adjust the perturbed radius to the shortest two of the three which is showed in the strategy image as a yellow color, which is a combination of red and green. The priminary lighting regions are an exception because they have a higher perceptual variance than the rest of the reflecting high variance regions. The paths still stay longer and have chance to adjust to the shortest radius which can be observed at the edge of the specularity of the mirror ball.

Our algorithm improves the image quality at the shadow and caustic regions, the reflections of these regions and even the diffuse wall in the back. The strategy image shows that there are more high variance regions, and thus the samples are more evenly distributed around the image plane. Arround the caustic region and the light source, we choose a small radius algorithm to render the image. PMC-PT achieves a more perceptually pleasant image compared to the corresponding image rendered by the PT algorithm using 256 spps, which is roughly the same total number of sample paths shooting from the eye.

Since the algorithm shifts extra samples from the low variance regions to high variance regions. If the scene contains a large number of high variance regions, such as this example, the number of extra samples can be moved is

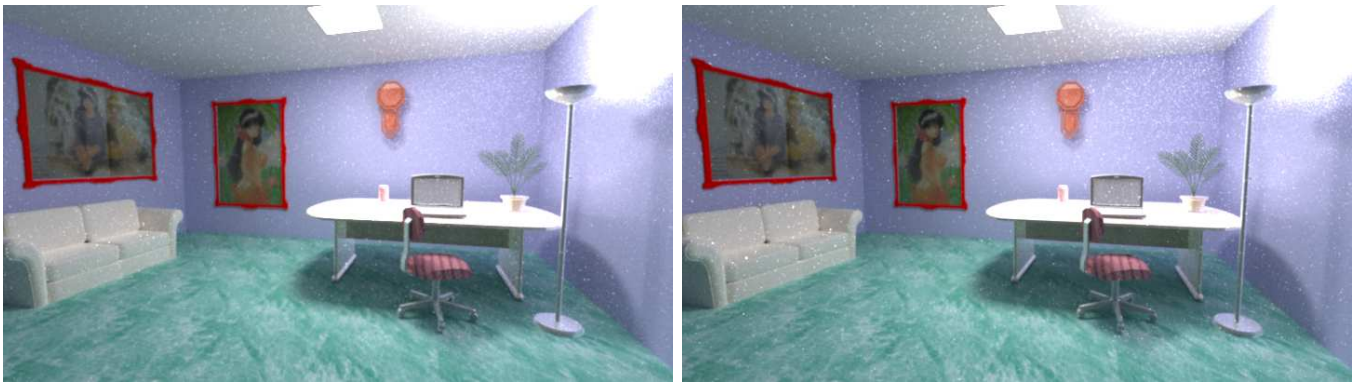

Figure 5. A room scene computed using our PMC-PT at the left and PT at the right. PMC-PT has fewer artifacts overall. By giving more samples to the high variance regions, PMC-PT is an improvement over PT.

relatively small. Although we still can gain improvement but the improvement is less obvious than those which have fewer high variance regions.

Finally, Figure 5 demonstrate that PMC-PT can be used to render a complex scene. We notice that the ceiling and the wall near the long lamp contain higher variance in the image rendered by PT algorithm. However, the PMC-PT distributes more samples on these regions and thus, the regions seem to be smoother. The overall artifact is smaller in the image rendered by PMC-PT than in the image rendered by PT using 1024 spps.

6 Discussion

The most important variable parameter in our algorithms is the survival rate in the resampling process. A small survival rate reduces the number of samples kept in the population, which results in a faster exploration of the sample domain but at the cost of a large amount of iteration information being lost during the resampling process. On the other hand, a larger survival rate means that more iteration information related to paths is kept during the iteration. However, the rate to explore the entire sample domain is slow.

The population size and resample rate can also further affect the rendering speed and the final result. If the total number of variance-criteria resampling samples for the T_R-loop (i.e $T_R * N_{Population} * r_{Resampling} * r_{Variance}$) is large enough, we can reduce the cost of generating a sample from the variance image, β's, by using a deterministic sampling method. In addition to efficiency, with deterministic sampling, the sample destribution is relatively stratified according to the sampling probability. Thus, we expect that a large size of population can further improve the rendering efficiency.

Mixtures are typically formed by combining several components that are each expected to be useful in some cases but not others. The adaption step then determines which component are useful for a given input. The most notable limitation of PMC is the high adaptive sample counts required for each iteration when the kernel has many adaptable parameters. This prevents us from using a larger number of different perturbing radiuses. Such a strategy would be appealing for efficiently rendering a scene with geometries having very different sizes appearing on the image plane, but the adaptive sample count required to adequately determine the mixture component weights would be too large. Instead we use three perturbation radiuses for all images rendered.

7 Conclusion

We have presented a novel approach, PMC-PT, by showing how to adapt PMC method to distribute the samples for a general path tracing algorithm. PMC-PT automatically distributes more samples over regions with high perceptual variance found in the previous iteration. It adjusts the importance sampling density to generate better samples for reducing perceptual variance of a high variance region and also use resampling process to eliminate samples locating in low-variance or well-explored regions and regenerate seed samples for achieving ergocity by the criteria of perceptual variance and stratification. There are several future research directions. Our PMC-PT only uses the information related to the regions on the image plane but does not use the correlation among similar paths. We would like to further explore the possibility of the correlated sampling in this aspect. In addition, all samples initialized the α's values to

a constant value. However, we can record the alpha used previously in an image because spacial correlation will give us similar α's values in most places in the image plane. We can reuse the α information to reduce the process of probing to estimate a proper set of α's values. Population Monte Carlo is new to the graphics society. We believe that it can provide further research opportunities in Computer graphics community.