Dependent Tests Driven Filtering in Monte-Carlo Global Illumination

Ferenc Csonka, László Szirmay-Kalos, Csaba Kelemen and György Antal

Department of Control Engineering and Information Technology, Technical University of Budapest
Budapest, Magyar Tudósok Krt. 2, H-1117, HUNGARY
Email: szirmay@iit.bme.hu

Abstract
This paper presents a multi-phase algorithm to solve the global illumination problem. In the first phase dependent tests are applied, i.e. the random walks of different pixels are built from the same random numbers. The result of the first phase is used to identify homogeneous pixel groups in the image. The criterion of the formation of such groups is that averaging the color inside these groups should result in less error than handling the pixels independently. The second phase of the algorithm is a conventional random walk method that uses independent random samples in different pixels. The final result is calculated as the average of the results of the dependent tests and the low-pass filtered version of the independent tests. This low-pass filter averages the pixel values inside the homogenous groups. The algorithm takes advantage of the fact that the image can contain larger homogeneous regions that can be calculated from much less number of samples. Thus we can focus on those pixels where significant changes happen.

Keywords: Monte-Carlo integration, random walks, dependent tests.

1. Introduction

When attacking the global illumination problem, a complicated integral equation, called the rendering equation must be solved for each pixel. Monte-Carlo techniques trace back the solution of the integral equation to the computation of high-dimensional integrals that are estimated using random samples.

If we obtain the solution pixel-by-pixel, the computational burden will be enormous, which can hardly be justified in many scenes and applications. Assume that we are facing a large homogeneous wall that has quite homogenous illumination. The image then is also homogenous. If this image is rendered with random walks[23], nearly the same computational effort should be paid as if in all pixels completely different scenes were visible. Random walks do not exploit the coherence of the image and the scene and thus repeat the same calculations in an inefficient way. Furthermore, the results of the different pixels are usually uncorrelated since we use different random numbers for their calculation. Thus the computational error appears as a random noise which is quite embarrassing for the human observer. Obviously, a low-pass image filter would be great help to improve the image of the homogeneous wall. It would reduce the noise and would force the pixels to get closer to their common mean.

The objective of this paper is to extend this simple idea for practical cases as well, when the image is not just a single homogenous block, but smaller homogeneous regions can be identified. These homogenous regions should be automatically identified and their filtering automatically controlled by the algorithm. We want to keep the asymptotic accuracy of random walks, thus as the number of samples goes to infinity all filtering artifacts should disappear.

A key point of this method is the recognition of the homogeneous regions. Obviously, those pixels should belong to the same region, whose colors will be almost identical asymptotically, thus filtering cannot smear edges and introduce other artifacts. However, if we use classical random walk algorithms that compute the pixels using independent random numbers, then it is very difficult to tell whether or not two pixels converge to similar colors. The color difference in a given stage of the algorithm can come from two sources. It can happen that the illumination environment of the two pixels are different (e.g. two different objects are seen in them), thus their limiting colors will also be far from each other. On the other hand, the difference can also stem from the random noise as well. It means that the limiting values are similar, thus it would be worth computing the average of the two pixels and replace their colors with the average. In the first case, averaging would result in artifacts, thus should be avoided, while in the second case averaging is highly beneficial.

In order to robustly choose between the two cases, we decompose the random simulation into two phases. Both phases aim at the solution of the rendering problem, but in the first phase the random variations of the color of neighboring pixels are tried to be minimized. Thus this phase not only provides a partial solution, but can also be used to form homogeneous pixel regions. In the second phase, the result of the first phase is refined by a normal random walk algorithm. The only difference is that when the final or partial image is displayed, an averaging operation is also carried out based on the results of the first step. We have to emphasize that the averaging operation does not use a constant kernel, but it also takes into account the number of samples computed in the second phase, aiming to minimize the total error of the random simulation and the averaging itself.

In the following sections we first review the previous work on function approximation for the global illumination, then the proposed multi-phase algorithm is introduced, including the phases of dependent and independent tests.

2. Previous work

In the solution of the global illumination problem two basic mathematical techniques have critical role: integration and function approximation. Integration is responsible for the evaluation of the light transport operator while function approximation has to provide representation for the final and temporary results. In both cases we have to take into account that the underlying function is high-dimensional, high-variation and very costly to sample.

Significant research efforts have been devoted to the efficient integration and most of the global illumination algorithms optimize this task. Monte-Carlo global illumination algorithms can be improved by variance reduction techniques, and particularly by importance sampling. Importance sampling places more samples where the integrand is large[19]. Efficient function approximation has been just of secondary importance. In the global illumination setting two kinds of functions are dealt with: the 2D image function and the 4D radiance or importance function (2 dimensions are needed to identify a surface point and another two dimensions for the direction). When the problem is restricted to the diffuse case, the radiance or importance becomes 2 dimensional.

A good function approximation approach would use just a few samples where the function is flat or constant and would concentrate the samples where the function becomes “wild”, i.e. of high variation. Note that this results in different sampling densities as would be generated by importance sampling which concentrates on regions where the function is large.

Function approximation in rendering was investigated first of all in the context of image processing and compression[11]. The function approximation problem also appeared in global illumination, although under different names. In order to approximate the radiance, a popular technique is the finite-element method, which approximates the radiance as a finite function series of predefined basis functions. Piece-wise constant basis functions are simple to use but an accurate approximation might require too many of them[3, 5, 20]. Higher order basis functions[27] and wavelets[2] provide more compact representation but are more difficult to use [22]. Another interesting approach is followed by the photon map[6], where the radiance is represented by point samples from where the radiance function is approximated from the values nearby[15]. The main problem of all of these techniques is that they can never be accurate enough, thus artifacts can appear where the radiance changes quickly. These artifacts include, for example, light leaks, smeared shadows and incorrect highlights[17]. Taking into account the requirements of good function approximation, the data used by the approximation would be devoted to those regions where the radiance is not trivial. Adaptive tessellation in the radiosity method, also called substructuring[16], hierarchical radiosity[1, 25], discontinuity meshing[4], and wavelet radiosity[2] all aim at this goal. Adaptive sampling[13, 24], on the other hand, is an example of optimized image space approximation. In order to improve the approximation in the image space, several filtering methods have been published[9, 14, 21].

The global illumination problem as an approximation problem in the image space got attention in [7] and was also discussed at the closing section of the Dagstuhl Seminar on Monte-Carlo Methods[8]. The method of dependent tests[18], also called correlated sampling, showed up in these papers, which inspired our approach as well.

3. The method of dependent tests

In Monte-Carlo random walk algorithms the rendering equation is solved separately for each pixel, that is, we solve as many high-dimensional integration problems as the number of pixels in the screen. From another point of view, the rendering is an approximation problem, where the image function is approximated at the pixel locations. Current random walk algorithms use different random or quasi-random numbers for the computation of different pixels, thus their errors are independent (up to the degree on which the pseudo-random series can be assumed to generate independent values). Do we really need such an independent approxima-

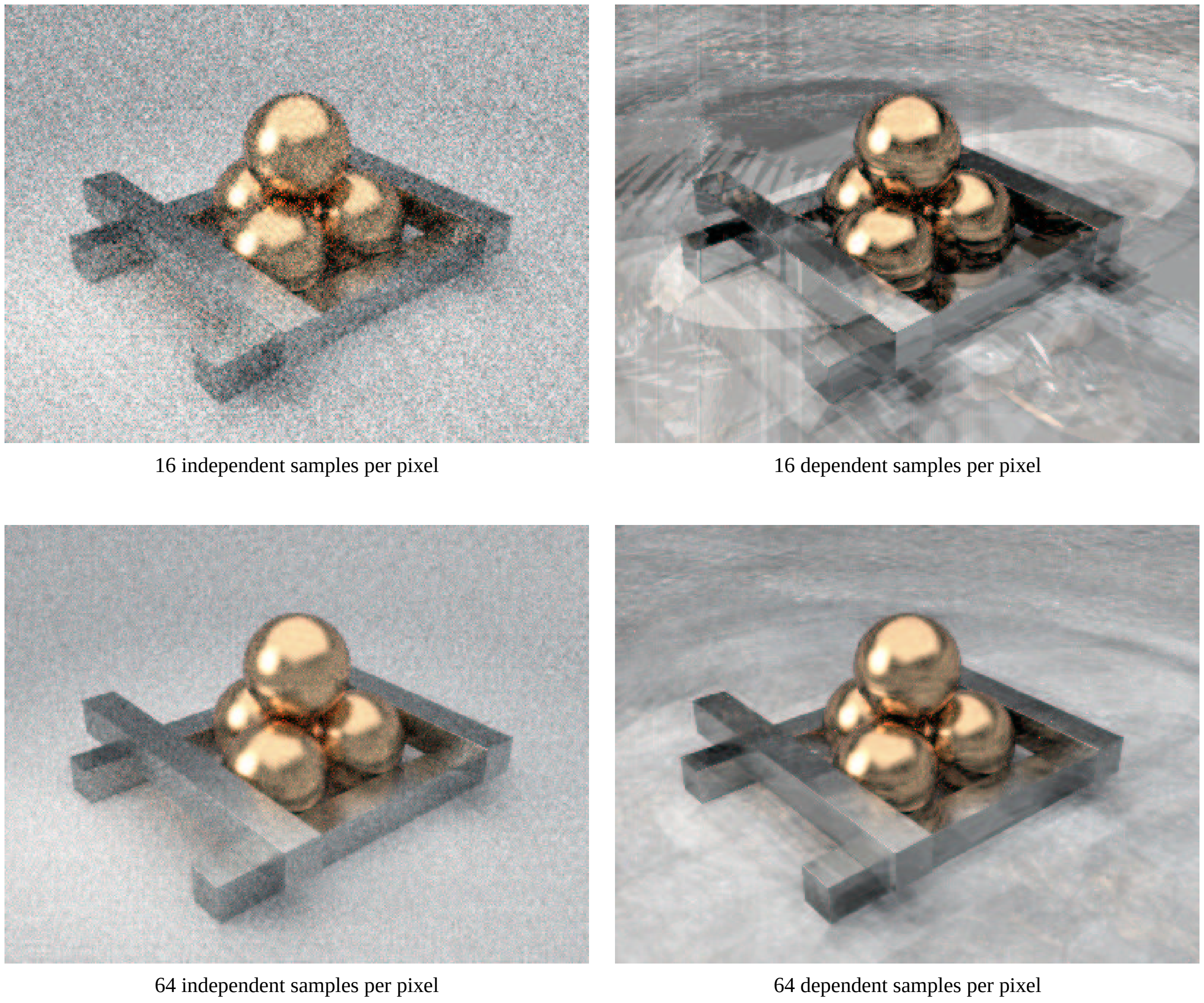
16 independent samples per pixel
16 dependent samples per pixel
64 independent samples per pixel
64 dependent samples per pixel

Figure 1: *Comparison of images rendered by path tracing with independent (left) and with dependent (right) tests*

tion? This independent approximation has both advantages and disadvantages. The primary advantage is that if we look at larger homogeneous regions (as the human eye does), the average color in these regions will be more accurate as the pixels themselves. The primary disadvantage is that when we look at the image, the dot noise on the image is rather disturbing.

In order to remove the dot noise, the method of dependent tests use exactly the same random numbers for the computation of all pixels. Obviously, the asymptotic result and the RMS error from the converged image are the same for both the dependent and independent sampling. However, now the pixel colors are highly correlated. This is advantageous since the dot noises disappear, but this is also bad since aliasing and higher level artifacts may occur. The method of dependent tests distributes the same error differently as the normal method using independent samples. This might be better or worse for the human visual system for different scenes and sample numbers (figure 1).

We do not intend to decide now whether dependent tests or independent tests are better in pure random walk rendering. For us, the important recognition is that dependent tests introduce a strong correlation between pixel colors, thus the difference of the colors will be a low variance random variable. Thus even if we have just a few samples, the large color difference would likely mean that the colors of two pixels converge to different values, thus averaging them is harmful. However, if the color difference is small, we can believe that these pixels are worth averaging because their converged values will also be similar.

Let us thus suppose that the first phase of the new algorithm is a random walk phase applying the method of dependent tests. Suppose that n random walks are computed in each pixel and the variance of the pixel color is computed, from the $C_1, \dots, C_n$ contributions of the walks:

$$\sigma^2 = \sum_{i=1}^{n} \frac{(C_i - \tilde{C})^2}{n} \approx D^2\left[C\right],$$

where $\tilde{C}$ is the average color:

$$\tilde{C} = \frac{\sum_{i=1}^{n} C_i}{n} \approx E\left[C\right].$$

The variance of the pixel colors will be used for two different purposes. On the one hand, they can determine how many additional samples should be computed in the different pixels. Brutal force random walk algorithms would use the same number of samples in each pixel, but it is obviously not optimal. Using the estimated variance, the goal is to compute all pixels with roughly the same relative error. The application of relative rather than absolute error is justified by the fact that the human visual system is sensitive to relative errors [12, 10]. From the value of the available time, or of the total number of samples to be computed, the number of samples in a pixel is set to be proportional to the variance, i.e. the square of the standard deviation, divided by the mean value of the pixel color. On the other hand, the variance of the pixel colors will also determine whether or not two pixels can be considered as similar and be included in the same group. We assume that significantly different variances mean that the two pixels are associated with different objects or illumination conditions, thus their averaging is not recommended. It also means that the pixels in a group will have similar variance value, which will be exploited in the error optimization.

Let us define a neighborhood of M pixels around each pixel. This neighborhood may contain only the given pixel, also called the *primary pixel*, 4 pixels that form a 2×2 square around the pixel (called 4-neighborhood), 3×3 pixels (called 9-neighborhood), etc. Pixels in the M-neighborhood of pixel p will be denoted by $\mathcal{N}_{M[p]}$.

The neighborhoods are defined to include pixels whose color average and variance are similar after the dependent tests. In order to decide whether or not the neighborhood is homogenous, color differences $\Delta_q = \|C_d[p] - C_d[q]\|$ are computed between each pixel in the neighborhood and the given pixel. A pixel has a homogenous neighborhood if these Δ_q values are small. It means that the new samples of the pixels in the neighborhood are also good estimates for the pixel itself. The precise definition of what “small” means is given in the following section.

4. Error driven adaptive filtering in the phase of independent tests

We concluded that if a pixel has homogenous neighborhood, then the samples corresponding to the pixels of the neighborhood are worth including in the given pixel as well. Thus the proposed filtering operation applies weighted averaging of the pixel colors of the M-neighborhood of the pixel:

$$C_f[p] = \sum_{q \in \mathcal{N}_{M[p]}} f[q-p] \cdot C[q], \qquad (1)$$

where $C[q]$ is a pixel color in the neighborhood including the pixel p itself, and $f[q-p]$ is the filter kernel. Note that p and q are 2D vectors, thus the filter kernel is also two-dimensional. The filter preserves the total energy, i.e. the average brightness remains the same if

$$\sum_{q \in \mathcal{N}_{M[p]}} f[q-p] = 1.$$

The first question is how the filter should be set. In order to answer this question, the error of the phase of independent samples is minimized. Having carried out sampling with N samples, the standard deviation of the pixel color is $\sigma/\sqrt{N}$. Three times the standard deviation is a probabilistic error bound with 97% confidence level, thus the Monte-Carlo error of the color is

$$\varepsilon(C[p]) = \frac{3\sigma}{\sqrt{N}}.$$

When the filtering operation is also used, the error comes from two terms, i.e. from the Monte-Carlo error of the pixels and from the distortion of assuming that the colors of different pixels are similar. When computing the variance of the filtered color in equation 1, we can assume that the colors of different pixels are statistically independent, thus we obtain:

$$D^2\left[C_f[p]\right] = D^2\left[\sum_{q \in \mathcal{N}_{M[p]}} C[q] \cdot f[q-p]\right] =$$

$$\sum_{q \in \mathcal{N}_{M[p]}} f^2[q-p] \cdot D^2\left[C[q]\right] = \sum_{q \in \mathcal{N}_{M[p]}} f^2[q-p] \cdot \frac{\sigma^2}{N}.$$

In the last equation we exploited that the pixels in a group have similar variance value. The Monte-Carlo error is thus its square root (standard deviation) multiplied by three. On the other hand, when a neighboring pixel q is used to estimate the given pixel p, a distortion of $f[q-p]\Delta_q$ is also added to the error. Adding the Monte-Carlo error and the distortion (or bias) in the sense of worst-case error analysis, we obtain the following error formula:

$$\varepsilon(C_f[p]) = \frac{3\sigma}{\sqrt{N}} \cdot \sqrt{\sum_{q \in \mathcal{N}_{M[p]}} f^2[q-p]} + \sum_{q \in \mathcal{N}_{M[p]}} f[q-p]\Delta_q.$$

Let us minimize this error formula by setting the filter coefficients f appropriately, but also taking into account the requirement of energy conservation. Using the Lagrange multiplier method, according to $f[q-p]$ and λ the partial derivatives of

$$\varepsilon(C_f[p]) - \lambda \cdot \Big(\sum_{q \in \mathcal{N}_{M[p]}} f[q-p] - 1 \Big)$$

should be made equal to zero, that is:

$$\frac{\partial \varepsilon}{\partial f[q-p]} = \frac{3\sigma}{\sqrt{N}} \cdot \frac{f[q-p]}{\sqrt{\sum_{r \in \mathcal{N}_{M[p]}} f^2[r-p]}} + \Delta_q - \lambda = 0,$$

$$\frac{\partial \varepsilon}{\partial \lambda} = \sum_{q \in \mathcal{N}_{M[p]}} f[q-p] - 1 = 0.$$

In order to simplify this system of equations, the quadratic mean is approximated by the arithmetic mean:

$$\sqrt{\frac{\sum_r f^2[r-p]}{M}} \approx \frac{\sum_r f[r-p]}{M} = \frac{1}{M}.$$

This allows to express the filter coefficients in a closed form:

$$f[q-p] = \frac{1}{M} + \frac{\sqrt{N}}{3\sigma\sqrt{M}} \cdot \left(\frac{\sum_r \Delta_r}{M} - \Delta_q \right).$$

In the practical implementation we modify the theoretical results a little bit to include also those factors that have not been taken into account during the theoretical analysis. For example, the proposed weight can be negative for small variance pixels. However, we want to use only a low-pass filter, thus these negative weights are replaced by zero. On the other hand, we do not have the same confidence in pixels of different distances from the primary pixel even if their color difference turns out to be similar. Thus a conventional pyramid filter is also included that decreases the computed weight of distant pixels. In the simplest case this pyramid filter would multiply the weights by

$$d(p-q) = M/\sqrt{2} - \|p-q\|,$$

that is, a linear function is used which is maximal for the primary pixel and decreases with the distance from the primary pixel. An even better solution would be the application of a spline or Gaussian filter.

Having dropped the negative weights and multiplied them by the distance function, the weights have to be renormalized since their sum is not necessarily 1 anymore. Thus the final form of the filter kernel is:

$$f^*[q-p] = \frac{f^+[q-p] \cdot d(q-p)}{\sum_r f^+[r-p] \cdot d(r-p)}$$

where f^+ means that negative values are replaced by zero:

$$f^+ = \begin{cases} f \text{ if } f > 0, \\ 0 \text{ otherwise.} \end{cases}$$

Let us examine this filter. If q is identical to the primary pixel p, then Δ_q is zero, thus the weight of the primary pixel will always be maximum and higher than $1/M$. The weights of the neighboring pixels are characterized by

$$d(q-p) \cdot \left(\frac{1}{M} + \frac{\sqrt{N}}{3\sigma\sqrt{M}} \cdot \left(\frac{\sum_r \Delta_r}{M} - \Delta_q \right) \right).$$

A neighboring pixel may have relevant weight if

- it is not far from the primary pixel, i.e. $d(q-p)$ is large,
- the Monte-Carlo error $\frac{3\sigma}{\sqrt{N}}$ is large compared to the color differences,
- there are not too many pixels in the neighborhood, i.e. M is small,
- its color difference Δ_q is small.

This means that the algorithm automatically locates high-variance pixels in homogeneous neighborhoods and applies low-pass filtering only here. The requirement that the Monte-Carlo error should be large also means that this filtering disappears for large sample numbers and the result is unbiased asymptotically. If the Monte-Carlo error is small compared to the color differences, then the weight of the primary pixel is significantly larger than those of neighboring pixels, thus other pixels in the neighborhood are not allowed to have relevant effect on the primary pixel. Such neighborhoods are not worth forming, since they just increase the computation time but the modification coming from their corresponding filters is negligible.

However, when the Monte-Carlo error is large, the pixels of the neighborhood can have considerable weight. Since the relationship between the weight of a pixel and its color difference from the primary pixel is linear with negative scaling, those pixels that have larger color difference will get smaller weight. This eliminates bad pixels in the neighborhood, thus the neighborhood may contain a few very different pixels, that are ignored.

Since this method will automatically eliminate those pixels that are not similar to the primary pixel by setting their weights close to zero, in theory we can expand the neighborhoods without any limits. More precisely, only the similarity of the variance values would limit the regions. However, larger neighborhood means higher computational time, which is wasted if the weights of the majority of the pixels are very small. Thus the neighborhood building algorithm will keep trace of the weights of newly introduced pixels. When these weights drop below a predefined limit, the expansion of the neighborhood is stopped. An easy way to detect whether or not newly taken neighborhood pixels have sufficient contribution is to check whether the decrease of $f[0]$, that is the decrease of the weight of the primary pixel in itself, is greater than a predefined threshold as we expanded the neighborhood. The minimally required decrease is denoted by $\Delta f[0]_{\min}$.

Finally, we should note that we have assumed so far that the sampled and the approximated function value is scalar. In global illumination, however, this function value is a vector of radiances at different wavelengths. This problem can be easily solved if the absolute values of color differences are replaced by appropriate norms, for example, by the sum of the absolute values of the color components of different wavelengths. It means that σ will be an estimate of the extent of the variance ellipsoid.

5. The algorithm

In this section the algorithmic details of the proposed method are presented. Each phase is defined by its pseudo-code. On the other hand, to demonstrate the features of the given phase, we took an example of a Cornell box-like scene and the temporary images of each phase are included in figure 2. In order to build random light-paths, we used bi-directional path tracing[26].

The proposed algorithm starts with the phase of dependent tests, which can be summarized by the following pseudo-code:

```
for each pixel $p$ do                                   // phase of dependent tests
    for $s = 1$ to $n$ do
        $C[s]$ = Dependent light path sample $s$ crossing this pixel
    endfor
    $C_d[p] = \sum_{s=1}^{n} C[s]/n$
    $\sigma^2[p] = \sum_{s=1}^{n} ||C[s] - C_d[p]||^2/(n-1)$
endfor
```

The result of the first phase obtained with $n = 50$ dependent samples per pixel is shown by the first image of figure 2. Note that dependent tests could get rid of the usual dot noises but characteristic stripes and other artifacts appear. For each pixel, this step results in a C_d value, which represents its average color and a variance value σ^2. If we have to obtain the results with at least a prescribed relative error ε_r, then the number of additional samples per pixel N can be determined as:

$$N = \left(\frac{3\sigma}{\varepsilon_r \cdot ||C_d||} \right)^2 - n.$$

This value is also used for characterizing the similarity of the neighboring pixels. Thus N is not precisely computed, but is only classified according to a few categories, e.g. 1, 10, 20, 50, 100, etc.

The phase of independent tests is like a conventional random walk algorithm except for the fact that we use different number of samples in different pixels according to the results of the first phase. The second step writes the average color into the C_i variable of each pixel:

```
for each pixel $p$ do                                   // phase of independent tests
    $C_i[p] = 0$
    for $s$=1 to $N[p]$ do
        $C$ = independent light path sample $s$ crossing this pixel
        $C_i[p]$ += $C/N[p]$
    endfor
endfor
```

The image obtained with independent tests is the second in figure 2. When this image was rendered, the prescribed relative error ε_r has been set to 0.03.

The third step of the algorithm is to form homogeneous regions based on the similarity of colors and variances obtained in the phase of dependent tests. According to the previous section, for each pixel p, the following algorithm should be executed:

```
$f^*[0] = 1$
for $m = 2$ to $m = m_{\max}$ do                        // form regions for pixel p
    // calculation of unnormalized filter coefficients
    $M[p] = m^2$
    for each pixel $q$ in the $M[p]$-neighborhood of $p$ do
        if $N[p]$ <> $N[q]$ then stop for pixel p
        $\Delta_q = ||C_d[p] - C_d[q]||$
        $f[q-p] = \frac{1}{M} + \frac{\sqrt{N}}{3\sigma\sqrt{M}} \cdot \left( \frac{\sum_r \Delta_r}{M} - \Delta_q \right)$
    endfor

    // the normalized filter coefficient of pixel p
    $f^*_{new}[0] = \frac{f^+[0] \cdot d(0)}{\sum_r f^+[r-p] \cdot d(r-p)}$

    // if the new normalized filter coefficient of pixel p is not
    // changed "too much", the region forming for this pixel is
    // stopped; otherwise the others are also computed
    if $f^*[0] - f^*_{new}[0] < \Delta f[0]_{\min}$ then
      $M[p] = (m-1)^2$
      stop for pixel p
    else
      for each pixel $q$ in the $M[p]$-neighborhood of $p$ do
          $f^*[q-p] = \frac{f^+[q-p] \cdot d(q-p)}{\sum_r f^+[r-p] \cdot d(r-p)}$
      endfor
    endif
endfor
```

Note that in the real implementation it can be exploited that larger neighborhoods include smaller ones, thus $\sum_r \Delta_r$ and $\sum_r f^+[r-p] \cdot d(r-p)$ can be computed incrementally.

The final step is the display of the result. The displayed color of a pixel is computed from its dependent color C_d and from independent colors of those pixels that belong to the homogeneous neighborhood of the given pixel, that is:

```
for each pixel $p$ do                                   // display of the results
    $C_i = 0$
    for each pixel $q$ in the $M[p]$-neighborhood do
        $C_i$ += $f^*[q-p] \cdot C_i[q]$
    endfor
    $C = (C_d[p] \cdot n + C_i \cdot N[p])/(n + N[p])$
    Display $C$ in $p$
endfor
```

This last step corresponds to the third and the fourth images in figure 2. The third image shows the filtered result of the phase of independent tests and the fourth the combination of the images obtained with dependent tests and with independent tests followed by filtering.

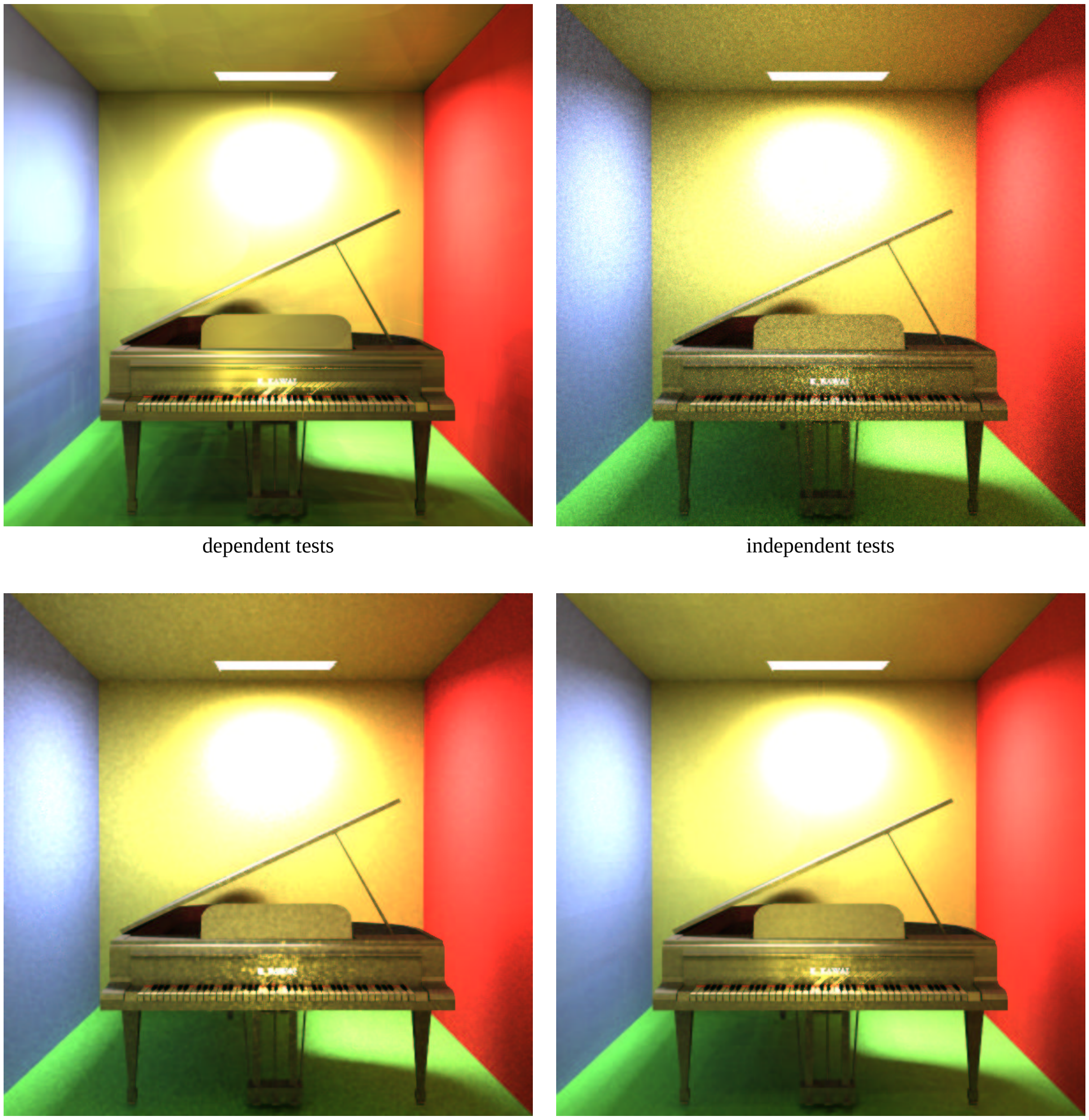

Figure 2: *Evolution of the final image in the proposed algorithm*

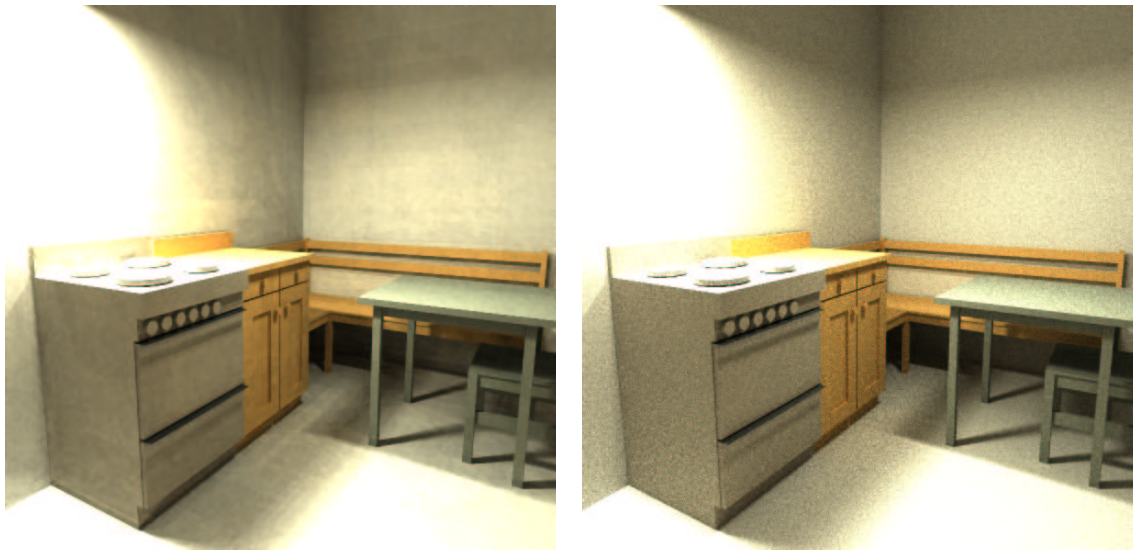

Figure 3: *Images rendered with our new method using 45 dependent tests and in average 43 independent tests per pixel (left, 94 min.) and with the original bi-directional path tracing using 150 samples per pixel (right, 159 min.)*

6. Analysis of the algorithm and simulation results

The presented algorithm has been implemented in C++ in OpenGL environment. The images have been rendered with 600×600 resolution on a PC with 1.5 GHz Pentium 4 processor. In order to evaluate the proposed method, we compared it first with bi-directional path tracing[26] in a kitchen scene (figure 3), which has 9143 patches included 2 emitters, and concluded that the new method provides better images in shorter computational time.

The proposed algorithm depends on four critical parameters including the number of dependent test samples n, the prescribed relative error ε_r, the maximum size of a neighborhood $m_{\max}$ and the minimally required decrease of the weight of the primary pixel $\Delta f[0]_{\min}$. These parameters should be carefully set. For example, if we use too many samples in the dependent test, the artifacts of dependent tests remain visible in the final image. On the other hand, if the number of dependent tests is small, the mean color and variance estimates are not accurate, which may result in not optimal region forming. According to our practical experiences, it is not worth using larger filter kernels ($m_{\max}$) than 4×4 pixels. Although, the calculation method would guarantee the optimal determination of the coefficients of even larger kernels, the uncertainty of the results of the dependent tests can still result in excessive low-pass filtering.

In order to highlight the difference of wrongly and well tuned parameters, figure 4 shows the same part of the kitchen scene rendered with different settings. The left image was obtained with allowing maximum 3×3 regions to form ($m_{\max} = 3$ and $\Delta f[0]_{\min} = 0.1$). The RMS and the perceptual errors in this case were 17.1 and 63733, respectively (as a perceptual measure we counted the number of those pixels where the relative error of the color exceeded five percent[12]).

However, as we can see in the middle image, a much better result is generated if we allow the neighborhoods to extend to 4×4 regions, i.e. we used $m_{\max} = 4$ and the threshold $\Delta f[0]_{\min}$ was decreased to 0.01 to allow farther pixels also to contribute. This modification reduced the RMS and the perceptual errors by about 15 percent.

In addition to the filter size and the threshold parameter, the algorithm can also be controlled by the prescribed relative error. Figure 5 shows the rendered images with $0.6 \ldots 0.03$ prescribed relative errors. Note that the filtering and dependent test artifacts gradually disappear.

Another indoor scene (figure 6), which has 18134 patches included 1 emitter, is also rendered with the new method. Note that the floor is a bit mirror-like, because the specular component of the floor is set to 0.6 and the shininess is 15.

7. Conclusions

This paper presented an adaptive sampling and filtering approach to reduce the computational time spent on homogeneous regions. In order to find these homogeneous regions, we used Monte-Carlo sampling with dependent tests. This allows to minimize the effect of the Monte-Carlo noise on the decision whether or not two pixels belong to the same region. Based on the result of the dependent tests, the variance of the pixel colors are also estimated, and the number

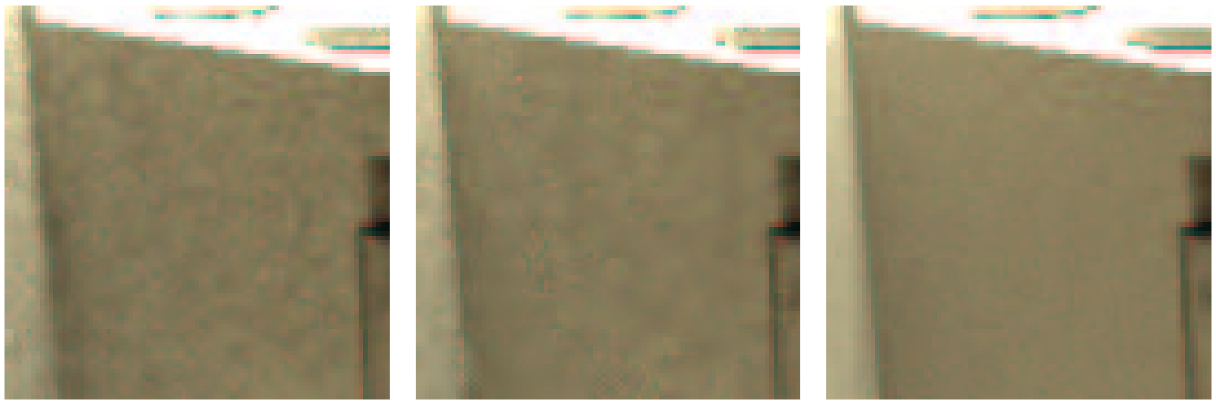
$m_{\max}$=3, $\Delta f[0]_{\min}$=0.1
$m_{\max}$=4, $\Delta f[0]_{\min}$=0.01
reference image

Figure 4: *Images of the side of the oven that compare the effects of different neighborhood size and threshold parameters*

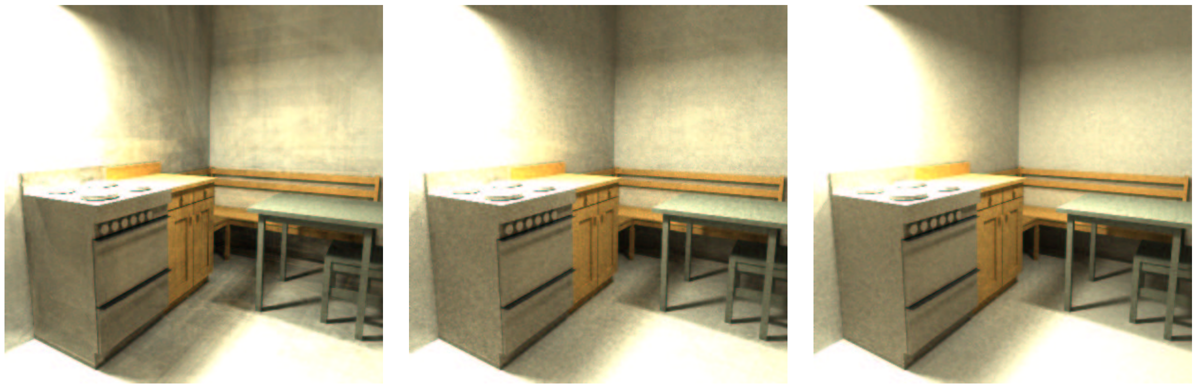

Figure 5: *Comparison of images rendered with different prescribed relative errors*

of additional samples are computed. The second phase of the algorithm is a classical random walk. The only difference is the final display of the color, since we also use a filtering operation. The domain of the filters are the homogeneous regions and the filter kernels are set to minimize the total error composed of the Monte-Carlo error and the bias of the filtering.

The computational overhead of the method is negligible and is really effective when the scene contains larger homogeneous regions (this is quite typical in architectural scenes) and the resolution of the image is high. So far we exploited the coherence in the image space. As a future development we intend to extend this idea for higher order reflections as well.

Looking at the method from another perspective, we can note that it is based on trading noise with bias in a way that the total error is minimized and the unbiasedness is still kept in the asymptotic case. We believe that such approaches have great potential to improve Monte-Carlo global illumination algorithms.

References

1. P. Bekaert, L. Neumann, A. Neumann, M. Sbert, and Y. Willems. Hierarchical Monte-Carlo radiosity. In *Rendering Techniques '98*, pages 259–268, 1998.

2. P. H. Christensen, E. J. Stollnitz, D. H. Salesin, and T. D. DeRose. Wavelet radiance. In *Fifth Eurographics Workshop on Rendering*, pages 287–302, Darmstadt, Germany, 1994.

3. M. Cohen and D. Greenberg. The hemi-cube, a radiosity solution for complex environments. In *Computer Graphics (SIGGRAPH '85 Proceedings)*, pages 31–40, 1985.

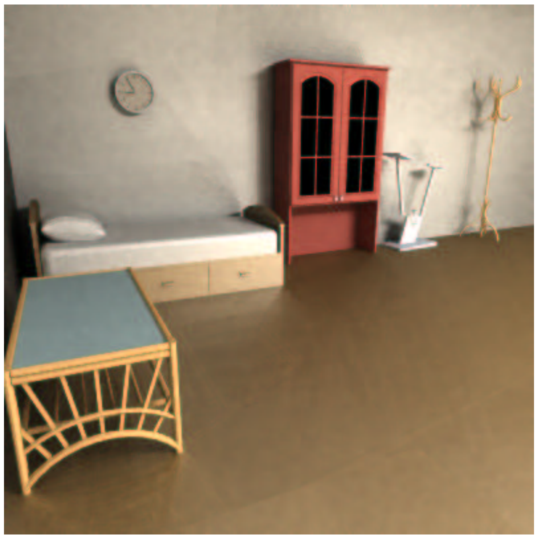

Figure 6: *An indoor scene rendered with the new method using 70 dependent tests and in average 60 independent tests per pixel (137 min.).*

4. P. Heckbert. Discontinuity meshing for radiosity. In *Third Eurographics Workshop on Rendering*, pages 203–226, 1992.

5. D. S. Immel, M. F. Cohen, and D. P. Greenberg. A radiosity method for non-diffuse environments. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, pages 133–142, 1986.

6. H. W. Jensen. Global illumination using photon maps. In *Rendering Techniques '96*, pages 21–30, 1996.

7. A. Keller. Hierarchical Monte Carlo image synthesis. Technical Report 298/99, Universität Kaiserslautern, AG Numerische Algorithmen, 1999. to appear in Mathematics and Computers in Simulation.

8. A. Keller. Correlated sampling. *Closing section of the Dagstuhl Seminar on Monte-Carlo Methods*, 2001.

9. M. E. Lee and R. A. Redner. A note on the use of nonlinear filtering in computer graphics. 10(3):23–29, 1990.

10. K. Myszkowski. The visible differences predictor: Applications to global illumination problems. In *Rendering Techniques '98*, pages 223–236, 1998.

11. W. Pratt. *Digital Image Processing*. John Wiley, 1991.

12. J. Prikryl and W. Purgathofer. Perceptually based radiosity. In *Eurographics '98, STAR — State of the Art Report*, 1998.

13. W. Purgathofer. A statistical method for adaptive stochastic sampling. 11(2):157–162, 1987.

14. H. E. Rushmeier and G. J. Ward. Energy preserving non-linear filters. In *Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 131–138, 1994.

15. P. Shirley, B. Wade, P. Hubbard, and D Zareski. Global illumination via density-estimation radiosity. In *Eurographics Rendering Workshop '95*, 1995.

16. F. Sillion and C. Puech. *Radiosity and Global Illumination*. Morgan Kaufmann Publishers, Inc., San Francisco, 1994.

17. P. Slusallek. Photo-realistic rendering — recent trends and developments. In *Eurographics '97, STAR reports*, pages 35–57, 1997.

18. I. Sobol. The use w by the Monte Carlo method. 2:717–723, 1962.

19. I. Sobol. *Die Monte-Carlo Methode*. Deutscher Verlag der Wissenschaften, 1991.

20. M. Stamminger, A. Scheel, A. Granier, F. Perez-Cazorla, G. Drettakis, and F. Sillion. Efficient glossy global illumination with interactive viewing. In *Graphics Interface'99, Kingston, Ontario*, 1999.

21. F. Suykens and Y. D. Willems. Adaptive filtering for progressive Monte Carlo image rendering. In *Winter School of Computer Graphics '00*, 2000.

22. L. Szirmay-Kalos. Global element method in radiosity calculation. In *COMPUGRAPHICS '93*, Alvor, 1993.

23. L. Szirmay-Kalos. *Photorealistic Image Synthesis Using Ray-Bundles*. D.Sc. Dissertation, Hungarian Academy of Sciences, 2000. www.iit.bme.hu/˜szirmay/diss.html.

24. R. Tamstorf and H. W. Jensen. Adaptive sampling and bias estimation in path tracing. In *Rendering Techniques '97*, pages 285–295, 1997.

25. R. F. Tobler, A. Wilkie, M. Feda, and W. Purgathofer. A hierarchical subdivision algorithm for stochastic radiosity methods. In *Rendering Techniques '97*, pages 193–203, 1996.

26. E. Veach and L. Guibas. Bidirectional estimators for light transport. In *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 419–428, 1995.

27. Harold R. Zatz. Galerkin radiosity: A higher-order solution method for global illumination. In *Computer Graphics, Annual Conference Series*, pages 213–220, 1993.