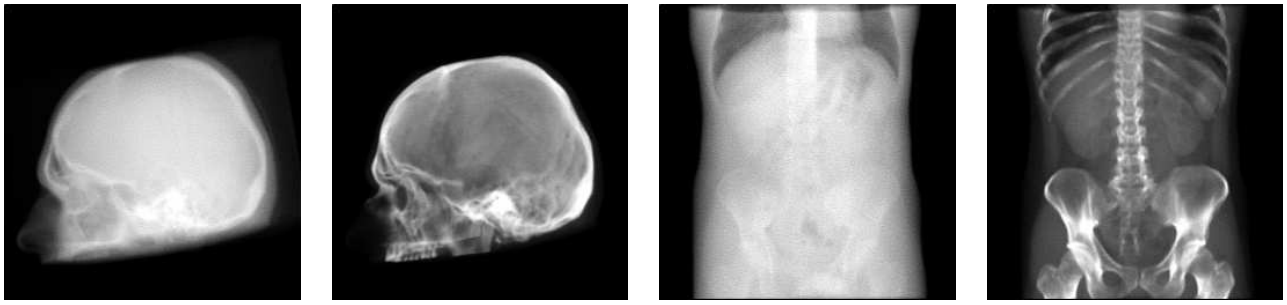
Interactive Transfer Function Control for Monte Carlo Volume Rendering

Balázs Csébfalvi*

Department of Control Engineering and Information Technology
Budapest University of Technology and Economics



(a) (b) (c) (d)

Figure 1: Monte Carlo volume rendering with (b, d) and without (a, c) transfer function parameter tuning.

ABSTRACT

Although Monte Carlo Volume Rendering (MCVR) is an efficient point-based technique for generating simulated X-ray images from large CT data, its practical application in medical imaging systems is limited by the relatively expensive preprocessing. The quality of images is strongly influenced by the transfer function, which maps a data value onto a sampling probability. An appropriate transfer function concentrates the point samples onto the region of interest. Since it is data dependent, a fine parameter tuning is necessary. However, the costly preprocessing has to be repeated whenever the transfer function parameters are modified. In this paper a new preprocessing algorithm is proposed for MCVR, which allows for an interactive transfer function control in the rendering phase, providing a visual feedback in a couple of seconds. In order to rapidly recompute point samples according to the modified transfer function, an efficient hybrid sampling strategy is applied, which combines the advantages of the probabilistic Monte Carlo sampling and the deterministic quasi-Monte Carlo sampling.

CR Categories: G.3 [Probability and Statistics]: Probabilistic Algorithms; I.3.3 [Computer Graphics]: Picture/Image Generation—Display Algorithms; I.3.7 [Computer Graphics]: Three-dimensional Graphics and Realism

Keywords: Monte Carlo volume rendering, importance sampling, progressive refinement.

1 INTRODUCTION

Since radiologist are well trained in interpreting gray-scale X-ray like images, in 3D medical imaging systems an X-ray volume-rendering tool is usually included. A simulated X-ray image can be generated by simply integrating the density function along the viewing rays. This approach often results in images of low contrast. Nevertheless, image quality can be improved by mapping the original data values onto intensities by an appropriately chosen transfer function, and integrating the reconstructed intensity function rather than the original density function. Figure 1 shows X-ray images generated by MCVR with (b, d) and without (a, c) transfer function parameter tuning.

Although several methods have been proposed for fast X-ray rendering, most of them do not support interactive transfer function control. Usually high frame rates are achieved by a computationally expensive preprocessing, which transforms the intensity function into the frequency domain by calculating Fourier or wavelet coefficients. Such a preprocessing assumes fixed intensity values at voxel locations. Therefore, whenever the transfer function parameters are modified, the costly preprocessing has to be repeated.

Our recently published Monte Carlo volume-rendering method also suffers from this limitation. Therefore, in this paper, a new preprocessing strategy optimized for interactive transfer function control is proposed for MCVR. It will be shown that 4M point samples, which guarantee a reasonably good image quality, can be recomputed according to the new transfer function parameters in a couple of seconds regardless of the original volume resolution.

In Section 2 the previous work related to X-ray volume rendering is discussed. In Section 3 and in Section 4 the Monte Carlo and quasi-Monte Carlo volume-rendering methods are reviewed respectively. Hybrid sampling is introduced in Section 5, and in Section 6 it is described how to exploit it for fast recomputation of point samples. In Section 7 the implementation results are reported, and finally in Section 8 the contribution of this paper is summarized.

2 PREVIOUS WORK

The practical applicability of recent X-ray rendering techniques in medical imaging systems can be analyzed from different aspects. On one hand, as the size of the volumetric data sets acquired by modern CT or MRI scanning devices drastically increases, the complexity of the applied display algorithm is getting more and more important. On the other hand, the display algorithm is required to be flexible, so it has to support interactive control of the rendering

*e-mail: cseb@iit.bme.hu http://www.iit.bme.hu/~cseb/

IEEE Symposium on Volume Visualization and Graphics 2004
October 11-12, Austin, Texas, USA
0-7803-8781-3/04/$20.00 ©2004 IEEE

parameters. These requirements are usually contradictive in practice.

Classical volume-rendering methods can be classified according to the order of data processing. *Object-order* methods process the volume voxel by voxel projecting them onto the screen. In contrast, *image-order* methods generate an image pixel by pixel casting a ray from the view point through the center of each pixel, and resampling the volume along the viewing rays.

Image-order volume rendering can be accelerated by decomposing the input volume into a hierarchical structure of voxel blocks in order to skip the empty regions [3]. Nevertheless, the most important advantage of the image-order approach, is that the evaluation of the rays can be terminated if the accumulated opacity has already reached a predefined threshold. *Early ray termination* [8], however, cannot be exploited using alternative visualization models, like X-ray volume rendering or Maximum Intensity Projection (MIP).

Applying the object-order approach for X-ray rendering, the voxels are projected as splats onto the screen, approximating the projection of the reconstruction kernel [17]. The contributions of these splats are accumulated in a high-precision accumulation buffer. Splatting is suitable for rendering sparse data sets, and by using a hierarchical decomposition of the volume, it supports progressive refinement [7]. For example, combining splatting with the hierarchical wavelet transform [9, 4], the data can be visualized at different levels of detail.

A common drawback of the classical image-order and object-order methods is that a volume of size N^3 can be rendered in $O(N^3)$ time. This is obvious for the object-order methods, since they traverse all the voxels. An image-order technique provides a similar image quality if the image resolution is at least N^2 and the number of samples along the viewing rays is at least N. Otherwise the original data is downsampled, so a voxel missed by the viewing rays can cause an arbitrarily high error in the projection. Therefore the complexity of image-order methods is considered to be $O(N^3)$ as well.

Performing volume rendering in the frequency domain, the time complexity of X-ray projection can be reduced to $O(N^2 \log N)$ [10, 14]. In a preprocessing of time complexity $O(N^3 \log N)$ the discrete Fourier transform of the volume is calculated. In the rendering phase, this frequency domain representation is resampled along a plane, which is perpendicular to the viewing direction and passes through the origin. According to the *Fourier projection-slice theorem*, the inverse Fourier transform of this resampled slice is equivalent to the projection of the volume. Thus, assuming that the image resolution is N^2, a simulated X-ray image can be obtained in $O(N^2 \log N)$ time. Fourier Volume Rendering (FVR) can be combined with the wavelet transform in order to enable progressive refinement [9, 4, 15]. Nevertheless, the Fourier projection-slice theorem is valid only for parallel projection, therefore FVR cannot be used for rendering perspective views.

The time complexity of X-ray projection can be further reduced by using Monte Carlo Volume Rendering (MCVR) [2]. According to this approach, a normalized reconstruction of the intensity function is used as a probability density function to generate a point cloud of random samples. Projecting this point cloud onto the screen an X-ray image of the volume can be obtained. Using quasi-Monte Carlo integration the accuracy of this method can be deterministically controlled. (Q)MCVR benefits from the fact, that the intensity integrals are finally quantized, and therefore they do not need to be evaluated at an accuracy, which is significantly higher than the accuracy of the quantization. Due to the controlled precision progressive integration, the evaluation is terminated if the required error level has already been reached. The number of point samples, which is necessary to keep the average pixel error below the level of quantization error, depends only on the number of pixels regardless of the number of voxels. In this sense the time complexity as well as the storage complexity of (Q)MCVR is $O(N^2)$. Here it is assumed that the image resolution is proportional to N^2 but it is fixed in advance. Thus, whenever the image resolution is increased, the preprocessing of complexity $O(N^3)$ has to be repeated to achieve the same pixel accuracy. In contrast, using FVR, a slice can be resampled at a higher resolution without repeating the preprocessing of complexity $O(N^3 \log N)$.

Recently one of the most popular volume-rendering techniques is 3D texture mapping [1, 16]. Although the complexity of this technique is $O(N^3)$, due to the hardware supported implementation, it significantly decreases the constants in the "O notation". The adaptation of this method to X-ray volume rendering requires a highly accurate accumulation buffer, therefore it can be implemented only on expensive graphics cards supporting floating-point precision and pixel shading. The most important drawback of 3D texture mapping, however, is that the local texture memory, where the volume data has to be loaded into is limited. Although large data sets can be decomposed into blocks, swapping these blocks between the main memory and the local texture memory might drastically reduce the performance. Therefore 3D texture mapping is mainly used to interactively render data sets of moderate size.

Apart from the complexity, flexibility is also an important aspect in practical medical applications. Unfortunately, recent X-ray rendering techniques of complexity lower than $O(N^3)$, do not support interactive transfer function control. For example, using FVR, the Fourier transform of the intensity function has to be recalculated whenever the user changes the transfer function, which can nonlinearly map the original densities onto intensity values. (Q)MCVR has a similar limitation, since the point-cloud also has to be recomputed according to the modified transfer function, which directly determines the distribution of the samples.

Taking the above mentioned aspects into account, in this paper, the original (Q)MCVR method is improved in order to support interactive transfer function control. It will be shown that a fast recomputation of the point samples can be done in $O(N^2)$ time, so the costly preprocessing does not have to be repeated whenever the image resolution is increased or the transfer function is changed.

3 Monte Carlo Volume Rendering

Monte Carlo Volume Rendering (MCVR) is a point-based technique for producing simulated X-ray images from large CT data [2]. In a preprocessing step a point cloud of random samples is generated by using a normalized reconstruction of the intensity function as a probability density function. In the rendering phase, this point cloud is projected onto the image plane, and to each pixel an intensity value is assigned, which is proportional to the number of samples projected onto the corresponding pixel area.

MCVR gives an unbiased estimation of the projected intensity function. The image quality is characterized by the standard deviation of the estimation. It has been shown [2] that the average standard deviation of all the pixels can be reduced below the level of quantization error if the following condition is fulfilled:

$$M > (\frac{1}{WH} - \frac{1}{W^2H^2}) \cdot W^2H^2B^2L^2 \approx WHB^2L^2, \qquad (1)$$

where M is the number of random point samples, W and H are the width and height of the image respectively, B is the average luminance taken from interval [0,1], and L is the number of quantization levels. Thus M has to be proportional to the to the number of pixels, but it does not depend on the number of voxels. Therefore the time complexity as well as the storage complexity of MCVR is a linear function of the number of pixels. As a consequence, if the volume size is N^3 then an X-ray projection of resolution N^2 can be generated by MCVR in $O(N^2)$ time using $O(N^2)$ storage ($M = O(WH) = O(N^2)$).

4 Quasi-Monte Carlo Volume Rendering

The convergence can be accelerated by applying quasi-Monte Carlo (QMC) integration [6], which is formally equivalent to Monte Carlo (MC) integration but it transforms a low-discrepancy deterministic sequence into the required distribution rather than uniformly distributed pseudo-random numbers. For example, 3D Halton points [5] defined in the unit cube can be used as a low-discrepancy quasi-random sequence. The kth point in this 3D Halton sequence is calculated as $\mathbf{h}_k = [H_k^5, H_k^3, H_k^2]$, where H_k^b denotes the kth number in a 1D Halton sequence of base b (see the definition in the Appendix). Each sample $\mathbf{x}_k = [x_k, y_k, z_k]$ in the point cloud is generated by transforming the corresponding Halton point $\mathbf{h}_k$ according to the following equations:

$$H_k^2 = \int_0^{z_k} \int\int p(x,y,z)dxdydz, \tag{2}$$

$$H_k^3 = \int_0^{y_k} \int p(x,y|z_k)dxdy,$$

$$H_k^5 = \int_0^{x_k} p(x|y_k,z_k)dx,$$

where

$$p(x,y,z) = \frac{g([x,y,z])}{\int\int\int g([x,y,z])dxdydz},$$

$$p(x,y|z_k) = \frac{p(x,y,z_k)}{\int\int p(x,y,z_k)dxdy},$$

$$p(x|y_k,z_k) = \frac{p(x,y_k,z_k)}{\int p(x,y_k,z_k)dx},$$

and $g(\mathbf{x})$ is a continuous reconstruction of the discrete intensity function $g(\mathbf{x}_{i,j,k})$. Such a transformation of a 3D Halton sequence is analogous to the MC importance sampling, since it concentrates more samples into the regions of higher intensity.

Theoretically, Quasi-Monte Carlo Volume Rendering (QMCVR) ensures a deterministic $O(M^{-2/3})$ error bound, while MCVR has just a probabilistic $O(M^{-1/2})$ error bound [13, 11]. In practice MCVR needs approximately twice as many samples to reach the same error level as QMCVR does. On the other hand QMC preprocessing is significantly slower than MC preprocessing, since analytical integration is necessary to evaluate Equation 2.

5 Hybrid Sampling

Since the advantages of MC and QMC sample generation are complementary, their combination seems to be fruitful. Therefore, in this section, a hybrid resampling strategy is introduced.

Assume that the original voxel densities $f(\mathbf{x}_{i,j,k})$ defined on a regular grid are transformed by an appropriate transfer function $t(f(\mathbf{x}_{i,j,k})) = g(\mathbf{x}_{i,j,k})$. A continuous reconstruction of the transformed discrete density function $g(\mathbf{x}_{i,j,k})$ is obtained by a simple convolution:

$$g(\mathbf{x}) = \sum_{i,j,k} g(\mathbf{x}_{i,j,k}) \cdot h(\mathbf{x} - \mathbf{x}_{i,j,k}), \tag{3}$$

where $h(\mathbf{x})$ is the reconstruction kernel. According to the MCVR approach, a pixel intensity $I_{i,j}$ is estimated from M samples [12]:

$$I_{i,j} = \int_{V_{i,j}} g(\mathbf{x})d\mathbf{x} = \int_V g(\mathbf{x})v_{i,j}(\mathbf{x})d\mathbf{x} \tag{4}$$

$$\approx \frac{1}{M}\sum_{k=1}^{M} \frac{g(\mathbf{x}_k)v_{i,j}(\mathbf{x}_k)}{p(\mathbf{x}_k)},$$

where V denotes the entire volume, $V_{i,j}$ is a subdomain which is projected onto the area of pixel (i,j), $p(\mathbf{x}_k)$ is the probability density of sample $\mathbf{x}_k$, and the visibility function is defined as follows:

$$v_{i,j}(\mathbf{x}) = \begin{cases} 1 \text{ if } \mathbf{x} \in V_{i,j} \\ 0 \text{ otherwise.} \end{cases} \tag{5}$$

In order to reduce the variance of the estimation, the probability density function $p(\mathbf{x})$ has to be proportional at least to the view independent term of the integrand, that is the transformed density function $g(\mathbf{x})$. Therefore, using the original MCVR approach, point samples are generated in two steps. In the first step a voxel location is selected with a probability proportional to the transformed density of the corresponding voxel. In the second step, the normalized reconstruction kernel $h'(\mathbf{x}) = h(\mathbf{x}) / \int h(\mathbf{y})d\mathbf{y}$ is used as a local probability density function to generate a random translation vector, which is then added to the voxel location selected in the first step. This two-tiered approach guarantees that $p(\mathbf{x})$ is really proportional to $g(\mathbf{x})$.

In order to incorporate advantages of MC and QMC sampling, the first step of MC sampling is slightly modified. Instead of transforming uniformly distributed pseudo-random numbers, a deterministic low-discrepancy Halton sequence [5] is transformed into the required discrete distribution. A voxel location $\mathbf{v}_{i(k)}$ is selected if $H_k^2 > G(\mathbf{v}_{i-1})$ and $H_k^2 \leq G(\mathbf{v}_i)$, where H_k^2 is the kth element of a Halton sequence of base 2, and function $G(\mathbf{v}_i)$ is defined as:

$$G(\mathbf{v}_i) = \frac{1}{\sum_n g(\mathbf{v}_n)} \sum_{j=1}^{i} g(\mathbf{v}_j). \tag{6}$$

According to our practical experience (see Section 7), such a hybrid sampling provides almost the same convergence speed as QMC sampling does (for some data sets hybrid sampling even overtakes QMC sampling) and its preprocessing cost is similar to that of the pure MCVR. Nevertheless, hybrid preprocessing of large data sets can still take several minutes.

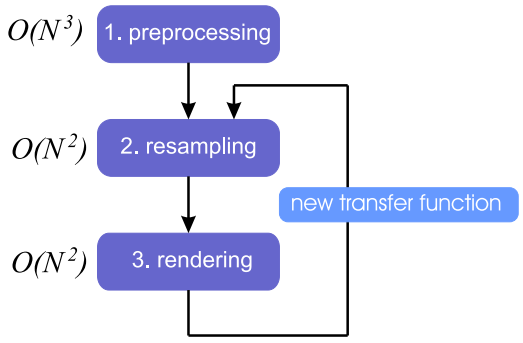


Figure 2: MCVR pipeline optimized for interactive transfer function control.

6 Fast Sample Generation

In this section, MCVR is optimized for an efficient recomputation of sample points. The original MCVR pipeline is modified in the following way (see Figure 2). In the first step, which is a preprocessing, the voxels are sorted by their density values and stored in an appropriate data structure. In the second step, this data structure is used to rapidly recompute the sample points according to the current transfer function parameters. In the third step, the point cloud is rendered, and based on the visual feedback the user decides whether the transfer function needs to be further modified.

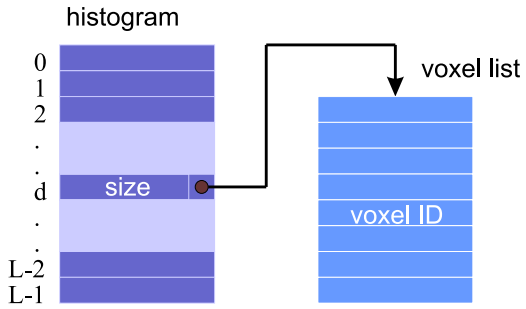


Figure 3: Bucket sorting of the voxel IDs according to the corresponding density values.

In the first step, the voxels are sorted based on the histogram of the data set. In a medical CT data the density values are usually stored in 8 or 12 bits, therefore only 256 or 4096 different density levels can be distinguished in a volume. Let us denote the number of density levels by L. During the preprocessing, an array of L voxel lists is allocated, which is referred to as **histogram[L]**. An array element **histogram[d]** contains a reference to a voxel array, which stores 4-byte IDs of voxels having the same density value d (see Figure 3). The ID of a voxel is determined from the corresponding voxel indices i, j, and k as: ID $= k \cdot X \cdot Y + j \cdot X + i$, where X, Y, and Z denote the volume dimensions.

In the second step, based on the current transfer function parameters, it can be easily determined how many samples to take from the voxel arrays of different densities. Using the following algorithm the point samples can be rapidly recomputed:

```
FastSampleGeneration(int m, double TF[]) {
  double sum = 0.0;
  for(int l = 0; l < L; l++)
    sum += TF[l] * histogram[l].size;

  long M = ((long)1 << m) - 1;
  double integral = 0.0; long iMin = 0;
  for(l = 0; l < L; l++) {
    long size = histogram[l].size;
    integral += TF[l] * size;
    long iMax = integral / sum * M;
    for(long i = iMin; i < iMax; i++) {
      long j = size * (i - iMin) / (iMax - iMin);
      long voxelID = histogram[l][j];
      Vector vector = voxelLocation(voxelID);
      vector += randomTranslation();
      pointList[Halton(i, m)] = vector;
    }
    iMin = iMax
  }
}
```

First of all $\sum_n g(\mathbf{v}_n)$ is calculated in variable **sum** (see Equation 6) taking the current transfer function parameters into account. The transfer function is evaluated at each density level and passed as an array (**TF[L]**) to the sample generator routine. Term $\sum_{j=1}^{l} g(\mathbf{v}_j)$ is represented by variable **integral** assuming that voxels $\mathbf{v}_1$, $\mathbf{v}_2$, $\ldots$,$\mathbf{v}_n$ are sorted according to their density values. In the second for loop, for each density level l the number of samples to be taken from the voxel array of density l is determined, which is **iMax - iMin**. Here it is assumed that the M number of all the samples can be expressed as $2^m - 1$, where m is a positive integer. In this case the ordering of a Halton sequence of base 2 is $1/2^m$, $2/2^m$, $\ldots$, $(2^m-1)/2^m$. By transforming this ordered Halton sequence rather than the original one the implementation of the sample generation is significantly simplified. However, in the final **pointList**, the order of the point samples has to correspond to the original order of the Halton sequence. Therefore routine **Halton(i, m)** is called, which mirrors the lower m bits of index i resulting in the correct permutation. The right order of the samples is important because of the smooth progressive refinement in the rendering phase.

Having a voxel ID selected, the corresponding sample point has to be generated. First the voxel location is calculated from the voxel ID in the following way: $z =$ ID div $(X \cdot Y)$, $y =$ (ID mod $(X \cdot Y)$) div X, $x =$ ID mod X. Here it is assumed that the 3D indices of a voxel are identical to the voxel coordinates. If it is not the case, the final point cloud is appropriately scaled taking the real voxel coordinates into account. Routine **voxelLocation** returns the location of a deterministically selected voxel as a vector. To this vector a random translation is then added, which is generated by using the normalized reconstruction kernel as a local probability density function. Since in this algorithm random sampling is combined with a deterministic sampling, it results in a mixture of the MCVR and QMCVR methods, which is referred to as Hybrid Monte Carlo Volume Rendering (HMCVR) in the further discussion.

Note that the time complexity of this hybrid sampling is proportional to the M number of samples and does not depend on the number of voxels. In order to guarantee that the average pixel error is under the level of quantization error, M has to be proportional to the number of pixels, which is assumed to be N^2 [2]. In this sense the complexity of algorithm **FastSampleGeneration** is $O(N^2)$. In contrast, using the original MCVR or QMCVR methods, the point sampling is built into the preprocessing, which traverses all the voxels, therefore its complexity is $O(N^3)$. Furthermore, the time and storage complexity of MCVR and QMCVR is $O(N^2)$ with a certain assumption. It is assumed that the number of pixels is proportional to N^2 but it is not modified in the rendering phase. Thus for a high quality zooming a greater number of point samples has to be generated to guarantee the same pixel accuracy. It is also true for HMCVR, but in this case, the samples can be recomputed in $O(N^2)$ time rather than in $O(N^3)$ time. So the time complexity of HMCVR is $O(N^2)$ even if the image resolution is allowed to be changed in the rendering phase.

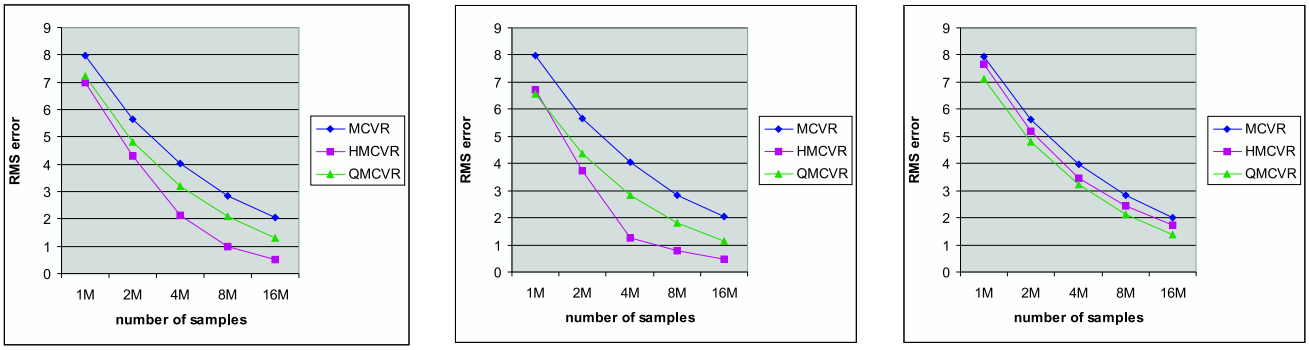
7 Implementation

HMCVR was implemented in C++ and tested on several medical data sets on a 2GHz AMD Athlon XP 2600 PC with 1GB of RAM. In Figure 4 the convergence of hybrid Monte Carlo sampling is compared to that of MC and QMC sampling. The reference images for the RMS error calculation were generated by analytically integrating the intensity function assuming trilinear reconstruction. Note that, for some data sets, hybrid sampling results in even faster convergence speed than QMC sampling does.

In order to get a visual feedback during interactive transfer function control, a preview of resolution 256×256 is rendered. For such an image resolution, a point cloud of 4M samples ensures a reasonably good image quality (see Figure 1). This point cloud can be interactively rendered at 5.32 frames/second on the test configuration using a pure software implementation. When the transfer function parameters are modified, the point cloud is recomputed in 2.28 seconds.

Having the appropriate transfer function parameters set, a high-quality simulated X-ray image is produced by taking a larger number of point samples according to the current transfer function. This high-quality rendering can be performed either on-line or off-line.

In case of off-line rendering, the viewing direction is fixed in advance. Therefore the point samples are immediately projected onto the image plane by the sample generator routine without storing them in a buffer. Preprocessing and off-line rendering (includes



(a) MRI angiography data. (b) CT scan of a human head. (c) CT scan of a human body.

Figure 4: RMS errors of MCVR, QMCVR, and HMCVR for different test data sets.

(a) MRI angiography data. (b) CT scan of a human head. (c) CT scan of a human body.

Figure 5: High-quality X-ray projections generated from 16M samples.

data set	MRI angio	CT head	CT body
number of voxels	8126464	10420224	7829520
preprocessing	3.36 sec	3.29 sec	3.87 sec
off-line rendering	4.61 sec	4.71 sec	5.04 sec

Table 1: Preprocessing and off-line rendering (includes resampling and projection) times in case of 16M point samples.

sample generation and projection) times in case of 16M point samples are shown for different data sets in Table 1.

For on-line rendering, however, the point samples have to be buffered in the main memory. In this case, interactivity can be ensured by using progressive refinement. Point clouds containing 1M, 4M, and 16M samples were rendered at 21.28, 5.32, and 1.39 frames/second respectively.

Figure 5 shows high-quality simulated X-ray images of different test data sets rendered by HMCVR. The transfer functions were fine tuned for each single volume. For example, the MRI angiography data set contains contrast agent in the blood vessels, therefore the transfer function emphasizes the higher density regions. Such a transfer function results in a similar visual appearance as that of a Maximum Intensity Projection (MIP).

8 Conclusion

In this paper Monte Carlo Volume Rendering (MCVR) has been optimized for interactive transfer function control. It has been shown, that after having the input volume of resolution N^3 preprocessed in $O(N^3)$ time, an X-ray projection can be recomputed according to the new transfer function parameters in $O(N^2)$ time. Apart from this practical improvement, this work has lead to an important theoretical contribution as well. The number of samples, which is necessary to reduce the average pixel error below the level of quantization error, is proportional to the number of pixels and does not depend on the number of voxels. Assuming that the image resolution has to be proportional to N^2, the time complexity of MCVR is $O(N^2)$. Nevertheless, using the original MCVR method, the image resolution is assumed to be fixed in advance. As a consequence, whenever the image resolution is increased, a greater number of samples has to be recomputed in $O(N^3)$ time. In this paper a new hybrid sampling algorithm has been proposed, which has time complexity of $O(N^2)$. Therefore, similarly to Fourier Volume Rendering (FVR), the computationally expensive preprocessing is performed only once, even if the user increases the image resolution in the rendering phase. However, FVR has preprocessing cost of $O(N^3 \log N)$ and rendering cost of $O(N^2 \log N)$. In contrast, using Hybrid Monte Carlo Volume Rendering (HMCVR), resampling and rendering can be performed in $O(N^2)$ time after a preprocessing of complexity $O(N^3)$.

ACKNOWLEDGEMENTS

This work has been supported by OTKA 42735, IKTA 00159/2002, and the Postdoctoral Fellowship Program of the Hungarian Ministry of Higher Education.

REFERENCES

[1] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings of IEEE Symposium on Volume Visualization*, pages 91–98, 1994.

[2] B. Csébfalvi and L. Szirmay-Kalos. Monte Carlo volume rendering. In *Proceedings of IEEE Visualization 2003*, pages 449–456, 2003.

[3] J. Danskin and P. Hanrahan. Fast algorithms for volume ray tracing. In *Proceedings of Workshop on Volume Visualization*, pages 91–98, 1992.

[4] M. H. Gross, L. Lippert, R. Dittrich, and S. Häring. Two methods for wavelet-based volume rendering. *Computers and Graphics*, 21(2):237–252, 1997.

[5] A. Keller. The fast calculation of form factors using low discrepancy sequences. In *Proceedings of Spring Conference on Computer Graphics*, pages 195–204, 1996.

[6] A. Keller. *Quasi Monte Carlo Methods for Photorealistic Image Synthesis*. Ph.D. thesis, Shaker Verlag Aachen, 1998.

[7] D. Laur and P. Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. *Computer Graphics (Proceedings of SIGGRAPH '91)*, pages 285–288, 1991.

[8] M. Levoy. Volume rendering by adaptive refinement. *The Visual Computer*, 6(1):2–7, 1990.

[9] L. Lippert and M. H. Gross. Fast wavelet based volume rendering by accumulation of transparent texture maps. *Computer Graphics Forum (Proceedings of EUROGRAPHICS '95)*, pages 431–443, 1995.

[10] T. Malzbender. Fourier volume rendering. *ACM Transactions on Graphics*, 12(3):233–250, 1993.

[11] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C (Second Edition)*. Cambridge University Press, 1992.

[12] I. Sobol. *A Primer for the Monte Carlo Method*. CRC Press, 1994.

[13] L. Szirmay-Kalos and W. Purgathofer. Analysis of the quasi-Monte Carlo integration of the rendering equation. In *Proceedings of Winter School of Computer Graphics*, pages 281–288, 1999.

[14] T. Totsuka and M. Levoy. Frequency domain volume rendering. *Computer Graphics (Proceedings of SIGGRAPH '93)*, pages 271–278, 1993. http://www-graphics.stanford.edu/papers/fvr/.

[15] M. A. Westenberg and J. B. T. M. Roerdink. Frequency domain volume rendering by the wavelet X-ray transform. *IEEE Transactions on Image Processing*, 9(7):1249–1261, 2000.

[16] R. Westermann and T. Ertl. Efficiently using graphics hardware in volume rendering applications. *Computer Graphics (Proceedings of SIGGRAPH '98)*, pages 169–176, 1998.

[17] L. Westover. Footprint evaluation for volume rendering. *Computer Graphics (Proceedings of SIGGRAPH '90)*, pages 144–153, 1990.

APPENDIX

Definition of a Halton sequence of base b: To construct the kth sample, consider the digits of the base b representation of k in the reverse order (that is $k = a_0 + ba_1 + b^2a_2 + b^3a_3 + \ldots$, where each $a_j \in \{0,1,\ldots,b-1\}$) and define the following element of [0,1]:

$$H_k^b = \frac{a_0}{b} + \frac{a_1}{b^2} + \frac{a_2}{b^3} + \frac{a_3}{b^4} + \ldots. \tag{7}$$

Definition of an n-dimensional Halton sequence: Chose n relatively prime integers $b_1, b_2, \ldots, b_n$ (usually the first n primes, $b_1 = 2, b_2 = 3, b_3 = 5, \ldots$, are chosen). The kth point in the Halton sequence is defined as $\mathbf{h}_k = [H_k^{b_n}, H_k^{b_{n-1}}, \ldots, H_k^{b_2}, H_k^{b_1}]$.