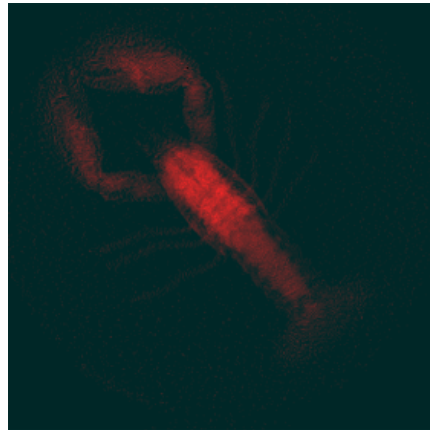


Monte Carlo Volume Rendering

Alvin Law

Elizabeth Blythe

Department of Computer Science
Purdue University



Abstract

This paper discusses the technique of Monte Carlo volume rendering (MCVR) for large volumetric data sets. MCVR transforms a uniform distribution of numbers in the range $[0, 1]$ and then uses a probability density function to importance sample the original dataset. The random points are then projected onto the image plane using a perspective projection. The intensity of a pixel is defined as the proportion of the number of sample points projected into the pixel over the total number of samples taken. As the number of samples increases, the image should converge to the desired solution. In this manner we are importance sampling the data points in a given dataset. This process dramatically reduces the processing time compared to traditional volume rendering techniques, such as ray casting, when working with large datasets such as those which result from an MRI. The results are X-ray images which clearly define key features of the datasets.

1. Introduction

With new developments in technology, the public has become accustomed to high quality images. The quality of images is especially important in the medical community where accuracy is extremely important. Through the use of modern 3D scanning technologies like the CT and MRI the detail available is drastically increased over traditional imaging techniques. However, with the increase in detail comes a dramatic growth in the amount of data being taken and, therefore, in the difficulty of processing

the resulting images quickly and efficiently. Traditional volume rendering techniques such as ray casting and splatting produce high quality images, but the computational cost required for these images is extremely high and often requires use of special graphics hardware in order to render the image at all.

In large datasets the contribution of a single low-intensity voxel is minimal. There may be many low-intensity voxels of this type – potentially in the scale of millions – and with traditional volume rendering techniques processing these large datasets takes a large amount of the total rendering time. One can observe that these voxels will contribute minimally to the overall image. It is therefore logical to spend less computational power on these voxels and use that saved time towards more important voxels. Capitalizing on the time saved through a preference towards important voxels is exactly what Monte Carlo volume rendering (MCVR) attempts to do.

In MCVR an X-ray image can be created by mapping uniformly random numbers in the range of $[0,1]$ to datapoints of the original dataset according to a probability density function. These importance-sampled points are then projected onto the image plane. The intensity of each pixel is estimated based on the number of samples projected onto the given pixel divided by the total number of samples. In this way, important sections from the dataset are sampled more often than those samples which do not contribute significantly to the final image. This saves computation time and allows for quick image rendering even for very large datasets with millions of voxels.

The main drawback to this method is that, as in traditional X-rays, depth information is not readily available. Depth information can be provided through applying shading and depth-cueing to an X-ray image. Applying shading will increase the processing time slightly but may be useful enough to outweigh the time increase. In order to shade, the image gradients need to be trilinearly interpolated from the neighboring samples points. The gradients are then used to determine a different intensity for the pixels which results in a shaded image.

We have implemented Monte Carlo volume rendering by rendering an X-ray image. It was found that, as suggested, a high quality image of the dataset is formed. The rendering of the image is very fast even for up to 10 million samples and beyond. We have also attempted to implement shading. However, the resulting images are not as revealing as expected.

2. Background and Related Work

This project was based on the paper *Monte Carlo Volume Rendering* written by Balázs Csébfalvi and László Szirmay-Kalos from the Department of Control Engineering and Information Technology at the Technical University in Budapest. In their paper, the authors were interested in improving the efficiency of 3D volume rendering on large datasets so that special graphics hardware would not be required in order to view high quality images. They also implemented Monte Carlo volume rendering and experienced results as expected with respect to computation time and detail of the final rendered image.

In addition, the paper briefly mentions a variance to the Monte Carlo integration concept called Quasi-Monte Carlo integration. This type of volume rendering is similar

to the original Monte Carlo rendering, but uses a more sophisticated method of generating the original point cloud. In Quasi-Monte Carlo Integration (QMCVR), a deterministic sequence is used as opposed to a random sampling. This modification allows for a slightly faster convergence to the desired image. However, because a more complicated method for generating sample points is used, the preprocessing cost for QMCVR is significantly higher than for MCVR. Csébfalvi and Szirmay-Kalos suggest that for practical applications a hybrid approach should be applied as a trade-off between MCVR and QMCVR preprocessing.

3. Method Details

X-Ray Volume Rendering

In order to generate an X-ray image of our dataset using MCVR we first need to create a point cloud of M random samples. This is done by mapping M uniformly distributed random numbers r_1, r_2, \dots, r_M in the range $[0, 1]$ to a probability distribution function (PDF) that characterizes the dataset. The PDF used by Csébfalvi and Szirmay-Kalos is defined as follows:

$$F(\mathbf{v}_i) = \frac{1}{\sum_n f(\mathbf{v}_n)} \sum_{j=1}^i f(\mathbf{v}_j).$$

where \mathbf{v}_j represents the j^{th} data point in the original dataset. A random sample \mathbf{x}_k can then be generated in two steps:

- (1) Datapoint \mathbf{v}_k will be chosen if the following inequality is satisfied:

$$r_k > F(\mathbf{v}_{i-1}) \text{ and } r_k \leq F(\mathbf{v}_i)$$

- (2) For each chosen sample apply a normalized reconstruction kernel, $h(\mathbf{x}_k)$, which is used to help blend the random samples and reduce aliasing. The reconstruction kernel must satisfy the following three properties:
 - (a) Separable as a direct produce of 1D kernels
 - (b) Non-negative over its domain
 - (c) The integral exists over all intervals

Our kernel chose a random direction, \mathbf{t}_k , scaled to half a voxel width. The k^{th} random sample is then formed using the following equation:

$$\mathbf{x}_k = \mathbf{v}_k + \mathbf{t}_k$$

After all of the random samples have been created, a perspective projection is used to project each sample onto the image plane. The projection will result in multiple sample points being mapped to the same pixel on the image plane. We keep track of the number of sample points projected to each pixel and use this information to scale the intensity of each pixel. Because of this scaling, a pixel which has fewer random samples

mapped to it is darker than a pixel which has many random samples mapped to it. This is important because data points from the original dataset which are not significant are sampled less often, and so will result in a dimmer pixel intensity in the final image.

Improvements

While the method described above produces good images using the Monte Carlo volume rendering technique, a couple improvements beyond what is described will improve either the processing time or the final image itself.

- (1) Csébfalvi and Szirmay-Kalos describe a way to significantly decrease the processing time by sorting the random numbers, r_1, r_2, \dots, r_M in ascending order. The sorting can be done using quicksort in $O(M \log M)$ average time. The decrease in processing time of the random samples greatly outweighs quicksort's cost. In our testing we discovered that rendering an image using this technique without sorting the random numbers r_1, r_2, \dots, r_M takes time in the scale of hours for an appropriate number of samples whereas simply sorting these random numbers reduces the sample generation time to mere seconds.
- (2) The final images contain noise around the dataset which can easily be reduced. This noise can be reduced by giving further weight towards important isovalues in the original data. Samples which had an isovalue above a threshold were weighted more heavily and therefore were sampled more often. This causes samples which are not significant to be sampled even less often which reduces the background noise in the resulting image.

Shading and Depth-Cueing

As mentioned above, MCVR produces accurate X-ray images, but depth information is lost in the final image. To account for depth Csébfalvi and Szirmay-Kalos suggest the use shading and depth-cueing in the X-ray images. This can be done by calculating the gradients at each point of the original dataset and trilinearly interpolating the gradients to calculate gradients at each random sample point x_k . These gradients will act as normals for our shading purposes. We will then need to calculate a depth-cueing function, $d(x_k)$, and a shading factor, $s(x_k)$. The depth-cueing function, $d(x_k)$, is simply as follows:

$$d(x_k) = 1/c$$

where c is equal to the distance from the light source to the sample point x_k . The shading factor used, $s(x_k)$, was a Lambertian bidirectional reflectance distribution function (BRDF). $s(x_k)$ is defined as follows:

$$s(x_k) = L \cdot N \cdot I \cdot C$$

where L is a vector from the light source to the point x_k , N is the normal at point x_k , I is the light intensity, and C is the color of the surface.

Based on the values obtained from the depth-cueing function and the shading function, the intensity I at pixel k can be calculated with the following formula:

$$I = \sum (s(x_k) * d(x_k)) \text{ from } i=0 \text{ to } k$$

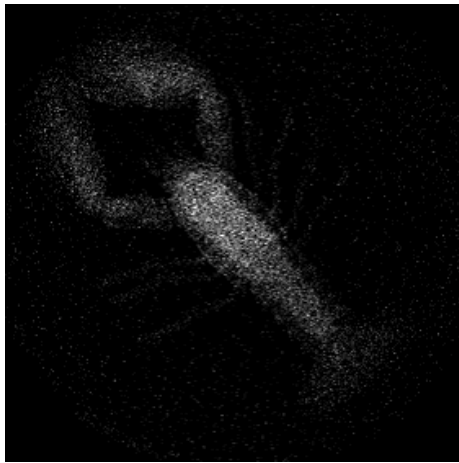
This intensity is then appropriately scaled based on the number of random samples points which have been projected to the pixel in a similar method to when generating the X-ray image.

4. Results

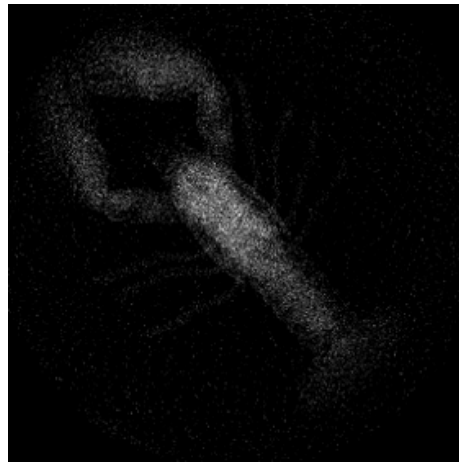
We have successfully implemented the MCVR technique as described to create X-ray quality images. The technique was applied to both the lobster dataset and the MRI dataset for a head. We have also attempted to implement shading and depth-cueing with limited results. Resulting standard X-ray images are shown in Figures (a) – (f) on the next page.

The resulting images for the X-ray images are comparable to those created by Csébfalvi and Szirmay-Kalos. In the lobster dataset, there is originally a “halo” around the lobster image which we have reduced using the PDF improvement of further weighting important datapoints. We also noticed that the lobster’s legs do not stand out as well as they should. This is because of the random sampling and the thin width of the legs. The legs of the lobster are relatively skinny compared to that of the rest of the body. The naïve reconstruction kernel used further smoothes away the shape of the thin legs. An intelligent reconstruction kernel would prevent the t_k translation vectors from smoothing out the legs.

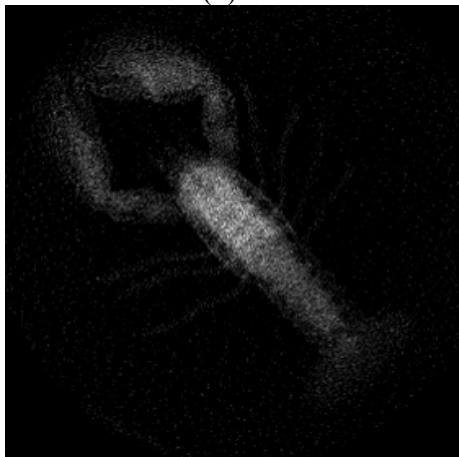
In Figure (a) below it is obvious that the rendered image is very grainy. This is an artifact of Monte Carlo techniques where a relatively small number of sample points are used. It can be seen as we increase the sample size M that the image of the lobster becomes much clearer and important details stand out more. Figure (f) shows a lobster with 1 million samples. At this point the ratio of samples, M , to pixels is sufficiently high, and the rendered image is near convergence and is indistinguishable from traditional volume rendering techniques.



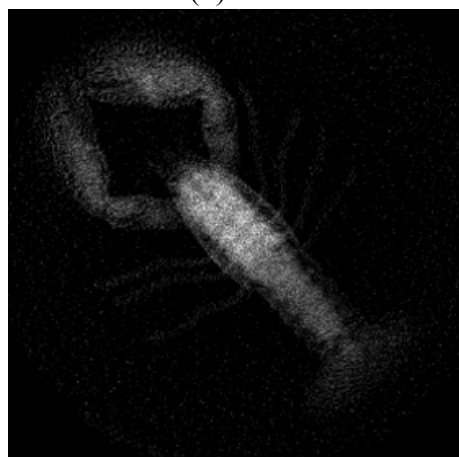
(a)



(b)



(c)



(d)



(e)



(f)

Figures (a)-(f) compare 400x400 images using different sample numbers.
(a) – 31,250 samples, (b) – 62,500 samples, (c) – 125,000 samples,
(d) – 250,000 samples, (e) – 500,000 samples, and (f) – 1,000,000 samples

Because images created by MCVR lack depth and shading characteristics, it is not useful to us to be able to produce these images if they take rendering time proportional to a traditional technique such as ray casting. Table 1 summarizes the computation times required to render the images in Figure (a)-(f) above. These renderings were done on a 1.7Ghz Pentium Centrino workstation with 512MB RAM. As can be seen, the image rendering time is independent of the size of the original dataset. The primary cost of the MCVR algorithm resides in the sampling of the dataset. This cost grows linearly with respect to M.

Number of Samples	Time for Sampling	Time for Rendering	Time for Complete Image
0.03125 million	0.053s	0.201s	0.254s
0.0625 million	0.100s	0.256s	0.356s
0.125 million	0.120s	0.210s	0.330s
0.25 million	0.220s	0.249s	0.469s
0.50 million	0.430s	0.288s	0.718s
1.00 million	0.802s	0.390s	1.192s

Table 1

Rendering an image using as many as 1 million sample points still takes approximately one second. This is a significantly shorter amount of time compared to standard ray

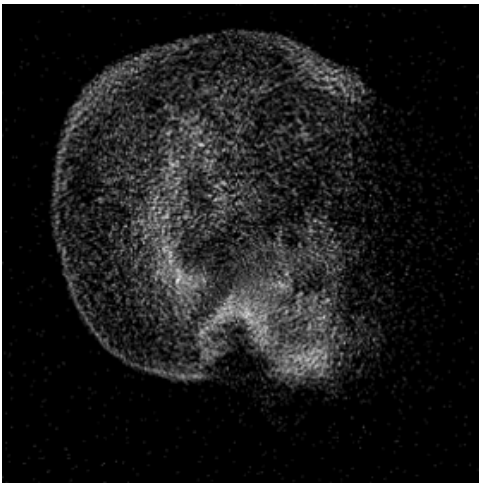


Figure 1: MCVR image using MRI dataset of a head. 1 million samples were used

casting techniques. In the smaller number of samples, the majority of the total time for using the Monte Carlo volume rendering technique comes from the actual rendering itself, and not from the sampling of the original dataset. It is only when we reach one-half million samples that the sampling technique even begins to make a difference, and even then it is not significant. Therefore, it is obvious that the MCVR technique is very efficient at sampling the data and rendering complete images.

The lobster dataset is not as large as some of the medical datasets which would use MCVR techniques. To observe the effects of MCVR on larger datasets, we have also rendered an image using MRI data for a head (see Figure 1 to the left). This image was created using only 1 million

samples. While this sampling size was sufficient for the lobster dataset, the MRI dataset is much larger and this sampling size was obviously not large enough to bring out all of the detail. In addition, we can see one of the drawbacks of the MCVR technique. Without coloring it is difficult to make out specific portions of the brain and impossible to differentiate between the bone, skin, and brain matter. Ray casting an image of this size would be able to provide such details at the cost of many minutes in rendering time. If such information is not prudent, then the MCVR produced image is sufficient and only took seconds to render.

We have also attempted to implement shading to provide depth information. While the resulting images were not as shaded as we would have liked, they do indicate a blending of the pixels which leaves us hopeful that further work in this area can produce the expected results. See Figure 2 to the right for an image we rendered using our shading technique. The shading technique has obviously increased the noise in our image even with the improvements to reduce the noise. As expected, the time required to process a shaded image is more costly. This is a direct result of computing the gradients at each of the original datapoints and then interpolating them in order to calculate the gradient at each of random sample point x_k . The image in Figure 2 required 2.297 seconds for the sampling which includes calculating the gradients and 3.51 seconds to produce a final rendered image. Storing these extra values requires additional memory. Due to memory restrictions, we were unable to create images with extremely large numbers of sample points as were possible with standard X-ray images.



Figure 2: MCVR image using shading and depth-cueing with 1 million samples

5. Discussion

The Monte Carlo volume rendering technique discussed in this paper successfully renders high quality images in a shorter amount of time than traditional volume rendering techniques such as ray casting. An analysis of advantages and disadvantages of both techniques provides insight regarding whether Monte Carlo rendering is a useful alternative to ray casting.

When ray casting, at least one ray must be cast from every pixel to accumulate volumetric data. This computation is potentially extremely expensive. Depending on the desired image features, this can be overkill. Images produced using ray casting are also view dependent. This is limiting to applications which require a change in the viewing angle. In this case, one would have to recast all of the rays again which is not conducive to user interaction. However, ray casting does have some strong points. For example, this technique makes it easy to highlight certain areas by assigning colors to known isovalue ranges. Therefore with ray casting we could color skin, bone, and other tissues different colors. See Figure 3 as an example of this.

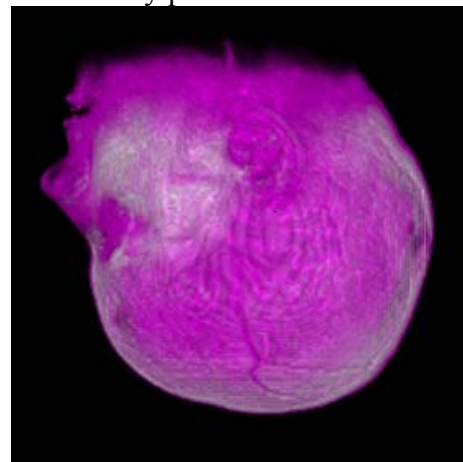


Figure 3: Image produced using ray casting highlight areas of the brain and skull

With MCVR we can generate the same quality images dramatically faster than with ray casting. Monte Carlo rendering also provides us with view independence making it more applicable to applications where speed is paramount. One could sample the original dataset with a very large number of samples x_1, x_2, \dots, x_M and re-project

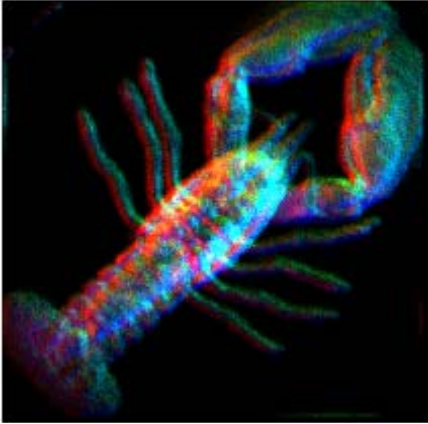


Figure 4: Color shaded lobster by Csébfalvi and Szirmay-Kalos

these samples from any viewpoint. While this benefit has not been implemented, it is feasible to add to the program. MCVR is the clear winner over ray casting when dealing with large volumes of data, but it does have some drawbacks which must be considered. Even when introducing shading into a Monte Carlo image correctly, depth-cues are not as easily understood as with ray casting techniques. In the paper by Csébfalvi and Szirmay-Kalos shading is implemented using different colors to represent different depths. See Figure 4 to the left.

This image is difficult to interpret and still does not show the depth detail that ray casting shows. In addition, it does not show different parts of the

lobster in specific colors whereas with ray casting it is possible to specify one color for the lobster's shell and another for the meat inside. Another apparent drawback to MCVR was discussed above regarding the lobster's legs. It seems as if datasets which have many important skinny areas may not render quality images using MCVR. Therefore, for these datasets it may be best to use another volume rendering technique.

Obviously there are drawbacks and advantages to each method. However, it seems as if the lower time processing in the Monte Carlo method is such a large advantage that it should be the preferred method when dealing with large datasets provided that specific features of the dataset do not need to be accentuated. While MCVR's lack of effective depth-cues also can be considered problematic, most datasets are interpreted by a user who is already familiar with the depth information of the dataset.

6. Conclusion

We have implemented the Monte Carlo volume rendering technique as described in the paper *Monte Carlo Volume Rendering* written by Csébfalvi and Szirmay-Kalos. This technique made use of a probability density function and reconstruction kernel to importance sample points from the original dataset. These random samples were perspectively projected onto the image plane and the intensities of the pixels were scaled based on the number of random samples projected to each specific pixel. The result of this method is an X-ray image which is comparable to traditional volume rendering techniques but can handle large amounts of data in significantly less time than classical ray casting methods. Additionally, we have attempted to provide depth information on the original X-ray images through the use of shading and depth-cueing. While our attempt produced an image which was clearly blended, the results did not provide the quality expected.

MCVR seems to be very useful in the rendering of images which require large amounts of data. If the purpose of the final image does not require knowledge of depth

information, there is no loss of data through this technique. In addition, where traditional methods are computationally slow and may even require special graphics hardware, this technique is dramatically faster without the use of special hardware. This is advantageous in the medical community where image scans such as MRIs require significant amounts of data and images are expected quickly.

In the future we would like to improve upon our shading technique to provide better quality images. Using hybrid Quasi-Monte Carlo integration is also possible to approach convergence faster.

7. References

Csébfalvi, B. and Szirmay-Kalos, L. 2003 Monte Carlo Volume Rendering
<http://www.fsz.bme.hu/~cseb/Publications/Vis03/>