Volume Ray Casting quality estimation in terms of Peak Signal-to-Noise Ratio

Nikolay Gavrilov Vadim Turlapov

Lobachevsky state university of Nizhny Novgorod
Gagarina 23
Russia 603950, Nizhni Novgorod

{gavrilov86, vadim.turlapov}@gmail.com

ABSTRACT

In spite of a large number of techniques aimed for improvement of Direct Volume Rendering (DVR) quality and performance proposed in the literature, there is a lack of approaches for numerical quality estimation of the images obtained by visualization of medical and scientific volumetric datasets. In this paper we propose a method to estimate sampling artifacts in DVR. Using the proposed estimation method we compare different Ray Casting algorithms to expose optimal ones in quality-performance criteria. We also propose method which combines two techniques for sampling artifacts reduction: Preintegrated DVR and Virtual Samplings method. We show that this combination overcomes both basic methods when using local shading or/and tricubic filtering in RC algorithm.

Keywords

Volume Rendering, Ray Casting, Scientific Visualization.

1. INTRODUCTION

Nowadays there are a lot of GPU-based approaches allowing for interactive rendering [EHK*06]. In addition to different acceleration structures to improve rendering performance by empty space skipping, there are many approaches to improve rendering quality without significant performance [EHMDM08], [KHW*09]. Still there is not any method proposed in the literature to estimate the quality of the output generated by RC algorithm, there are no criteria by which we could compare different quality improvement techniques. Mostly researchers simply put images of competing algorithms side by side, appointing the human visual system to be the judge [MHB*00].

In this paper we propose a noise-based method to estimate sampling artifacts of RC. In addition we propose new RC rendering techniques and compare them with the preintegrated rendering method [EHMDM08] in terms of quality and performance.

2. RAY CASTING ARTIFACTS

Methods involving uniform sampling with post-classification invariably miss thin features along the ray path, omitting the desired surface features thus casing *sampling artifacts* [KHW*09]. The regularity of such artifacts has a wood-like appearance, which can be converted to noise by stochastic jittering of ray starting positions or other shifting strategies [Sch05]. For big datasets (of size greater than 512^3) or transfer functions that cause superficial thin slices the optimization of RC algorithm is needed to keep an acceptable rendering quality and performance.

There is also another important type of DVR artifacts which are introduced by the interpolation method – *filtering artifacts*, caused by trilinear interpolation. Unfortunately they cannot be randomized like sampling artifacts, but they can be significantly reduced by using B-splines interpolation [RtHRS08].

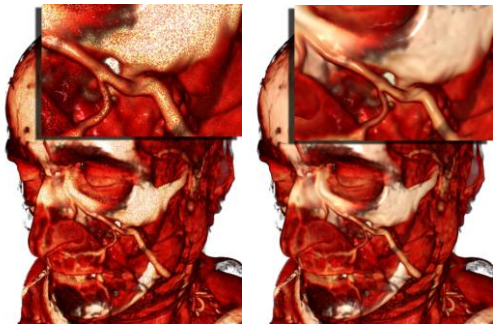


Figure 1. RC quality for sampling rate 1(left) and 8 (right)

Figure 2. Trilinear (left) & tricubic (right) interpolation modes side-by-side comparison.

3. QUALITY MEASUREMENT

As we use stochastic jittering of starting positions of the rays, sampling artifacts can be captured by the

image noise measurement. We can obtain a set of rendered images of the same dataset from the same viewpoint with the same TF and other visualization settings, but we can still change the jittering pattern changing the seed for random values generation.

We can consider the image pixels as a set of independent random values to measure the noise of each single pixel by evaluating the dispersion of its color C (we use YPbPr color space for the calculations below). We consider dispersion of C as an error in pixel (i, j), or as a level of its sampling artifact:

$$D(x,y) = \frac{1}{T}\sum_{i=1}^{T} ||C_i(x,y) - \frac{1}{T}\sum_{j=1}^{T} C_j(x,y)||^2$$

where T is a number of images in the series, $C_i(x,y)$ – color of (x, y) pixel from i^{th} image. It is also possible to use a ground truth image $\tilde{C}$ (image, obtained with a tremendously high sampling rate) instead of the average one:

$$D(x,y) = \frac{1}{T}\sum_{i=1}^{T} ||C_i(x,y) - \tilde{C}(x,y)||^2$$

To measure the noise level we use PSNR (Peak Signal-to-Noise Ratio) logarithmic decibel scale which is mostly used to measure the error introduced by images compression. In the volume rendering domain PSNR is mostly utilized to measure the error of 3d dataset compression [GS04], but not for the rendering quality estimation. We consider PSNR as a quality of the rendering algorithm, it is calculated as follows:

$$PSNR = 10log_{10}\left(\frac{MAX_I^2}{MSE}\right) \qquad MSE = \frac{1}{N}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} D(i,j)$$

D(i,j) is dispersion (or mean-square error) in *(i,j)* pixel, *MSE* – mean-square error of the whole image of size *m* x *n*, MAX_I – maximum possible of length pixel in the color space, equal to 1 in our case, *N* – number of non-background pixels, which is often less than n*m, there are too many background pixels with a null dispersion (see Fig. 4). To avoid the quality overstatement these pixels should be ignored.

We also use PSNR to estimate quality of each single pixel in order to build the quality map of the image which shows us areas of higher and lower rendering quality. The pixels with PSNR > 40dB have no noticeable noise – for these pixels the mean error is less than 1%. Areas where PSNR is less than 30dB contain rather noticeable noise. Figure 3 shows quality maps for different sampling rates.

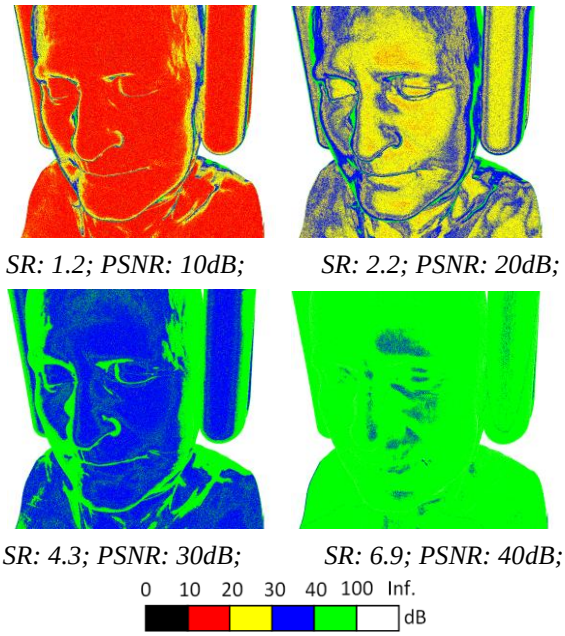


Figure 3. Quality maps and corresponding PSNRs for DVR with different sampling rates (SR).

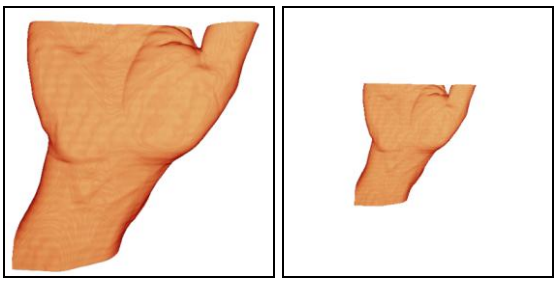


Figure 4. Here we do not ignore background pixels in PSNR calculation.

4. EXPERIMENTS

There are a lot of conditions besides the sampling rate that have influence on RC quality and performance: dataset, viewpoint, transfer function, screen resolution, sizes of bricks we use to decompose our dataset, shading model, filtering method, GPU we use, etc. Fortunately, some of these parameters do not affect rendering quality. Screen resolution affects only the precision of PSNR estimation. The dependence between number of processed pixels and frame rendering time is almost linear, as well as dependence between RC sampling rate and rendering time.

We used 10 different volumetric datasets of sizes 512x512x420 to 512x512x5382 12-bit. We make datasets decomposition into bricks of size 256^3 which appears to be optimal for the GPU we used (GeForce GTX 580 3GB). We used 6 different TFs, some of them impose strict visualization conditions causing presence of thin slices in space.

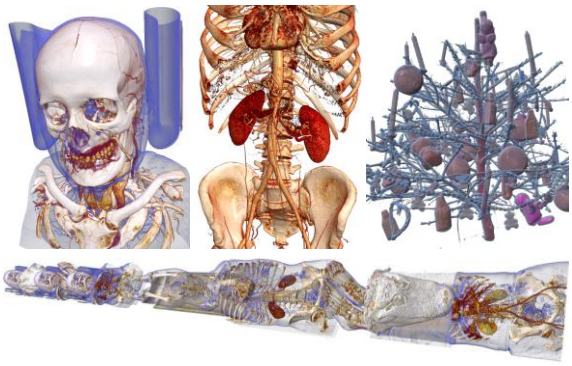


Figure 5. Examples of datasets used in experiments.

Quality-Performance dependence

When comparing efficiency of different quality improvement methods it is necessary to consider not only quality but the rendering performance as well. We change RC sampling rate, thus varying rendering quality and performance. As the result, we obtain quality-performance dependence for each rendering method. Depending on rendering conditions we have obtained a set of dependencies. There are 4 cases: use/not use local shading and trilinear/tricubic filtering. Mostly these two options define the optimal RC algorithm, while other visualization conditions do not have such influence on the efficiency of method. On Figures 6 and 7 we present typical dependencies we obtained in our experiments. Each line corresponds to a rendering method. We vary sampling rate from 1 to 8 samplings per voxel.

Rendering methods we used are: UDVR (Unoptimized DVR), PDVR (Preintegrated DVR [EHMDM08]), CVS (virtual samplings with cubic spline interpolation [LYS*10]), LVS (virtual samplings with linear interpolation), ASM (adaptive step method), PLVS and PCVS (are modifications of LVS and CVS: we make pre-integrated classification instead of post-classification)

In all cases experiments have showed inefficiency of UDVR method. When comparing PSNR values we mostly take in account [30dB, 40dB] range which corresponds to convenient rendering quality. When PSRN>40dB the artifacts are hardly perceptible by a human.

In many cases PDVR height efficiency is caused by its high performance, but when using shading its quality is dramatically limited. ASM is efficient only in cases with shading and trilinear filtering because we perform many samplings causing low performance. On the other hand, we calculate gradient on each ray step thus providing higher shading quality.

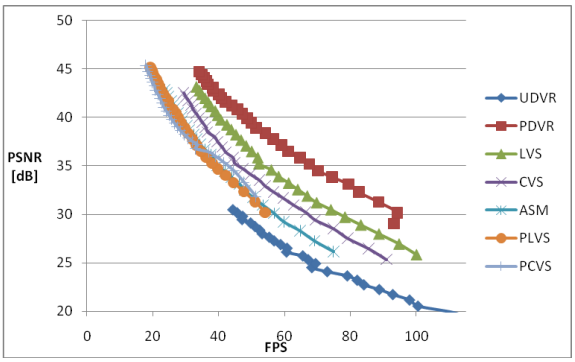


Figure 6. Quality-performance dependencies for different RC algorithms in case of not using tricubic filtering and without local shading.

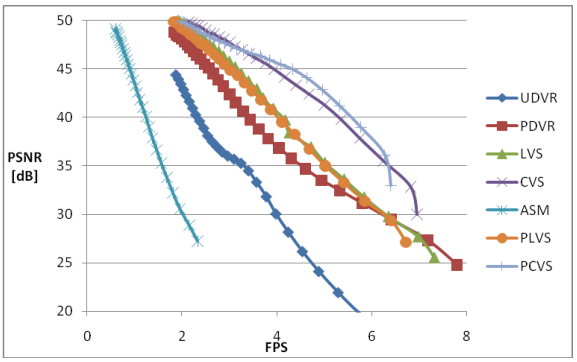
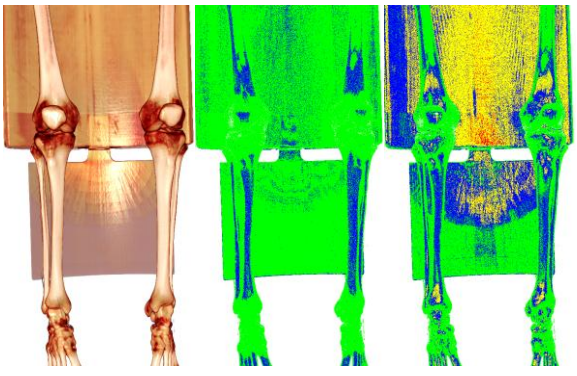


Figure 7. Quality-performance dependencies for different RC algorithms in case of using tricubic filtering and local shading.



DVR Output; PSNR: 36.4dB; PSNR: 22.1dB;

Figure 8. Quality maps for PDVR method without (middle) and with shading (right); sampling rate used here is 2 per voxel.

For LVS and CVS we use interpolated gradient, thus providing acceptable shading quality. But they have lower performance than PDVR has at the same sampling rates and they use post-classification method instead of preintegration, so that often they have lower efficiency.

As for techniques PLVS and PCVS, they are most acceptable in cases of using tricubic filtering or

shading. Expensive tricubic filtering compensates the additional computations in these methods to interpolate data values, which are absent in PDVR. On the other hand they provide with better shading. Figure 8 shows low PSNR for the Preintegrated rendering with shading on, while PLVS (or PCVS) methods interpolate gradient to avoid such artifacts.

Optimizing RC algorithms

Before comparing efficiency of RC algorithms we have made their optimization by searching theirs optimal parameters. For instance, in all RC algorithms we have proposed in this paper (all except UDVR and PDVR) there is such parameter as number of ray step division which defines number of internal steps. Varying this parameter for an algorithm we change its quality and performance like we did this by varying the sampling rate. At some point the augmentation of this parameter does not significantly improve quality.

We can also consider a RC method as a set of different RC methods with different number of step divisions in order to build quality-performance dependencies. Figure 9 shows that optimal numbers of divisions in LVS method are 3 and 4. In general we obtain the same result for all other RC methods. However, in cases of TF that causes thin slices in object space we need more step divisions for non-preintegration methods, i.e. LVS, CVS and ASM. In that cases we need up to 10 divisions to avoid severe sampling artifacts. Surely in those cases approaches that use preintegration table work much better.

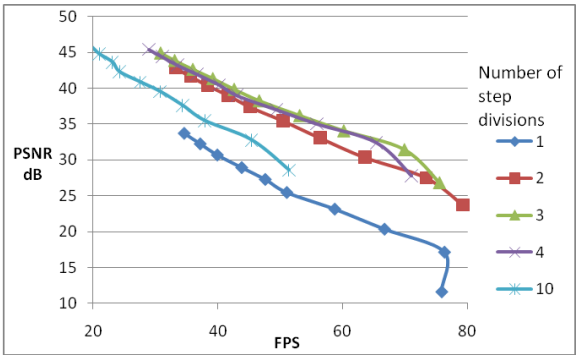


Figure 9. Quality-performance dependencies for LVS algorithm with different number of step divisions.

Shading & Filtering influence on PSNR

Unfortunately filtering artifacts cannot be measured as a noise like sampling artifacts. Still those regions on the image where trilinear filtering artifacts appear have lower PSNR in comparison to those on the image obtained with tricubic filtering. The overall PSNR is almost the same.

The local shading makes the image darker and this causes lower dispersion of intensities of pixels, i.e. higher overall PSNR. Still PDVR method shows better quality when the shading is off.

5. CONCLUSION

A method for Ray Casting quality numerical estimation was proposed. By evaluating noise we calculate PSNR for each single pixel and for the whole image as well. The usage of PSNR allowed us to measure RC noise in decibel scale, and like in images compression domain, the desired quality lies in [30dB, 40dB] range. Comparing PSNR produced by different RC algorithms at fixed fps and varying their parameters we can compute optimal ones for any particular class of visualization cases, e.g. reconstruction filter or shading options.

6. REFERENCES

[EHK*06] Engel K., Hadwiger M., Kniss J., Rezksalama C., Weiskopf D.: Real-time volume graphics. Eurographics Association (2006), 112–114.

[EHMDM08] El Hajjar J. F. et al. Second order pre-integrated volume rendering //Visualization Symposium, 2008. PacificVIS'08. IEEE Pacific. – IEEE, 2008. – C. 9-16.

[GS04] Guthe S., Strasser W.: Advanced techniques for high-quality multi-resolution volume rendering. Computers & Graphics 28, 1 (2004), 51–58.

[KHW*09] Knoll A., Hijazi Y., Westerteiger R., Schott M., Hansen C., Hagen H.: Volume ray casting with peak finding and differential sampling. Visualization and Computer Graphics, IEEE Transactions on 15, 6 (2009), 1571–1578.

[MHB*00] Meissner M., Huang J., Bartz D., Mueller K., Crawfis R.: A practical evaluation of popular volume rendering algorithms. In Proceedings of the 2000 IEEE symposium on Volume visualization (2000), ACM, pp. 81–90.

[LYS*10] Lee, B., Yun, J., Seo, J., Shim, B., Shin, Y. G., & Kim, B. (2010). Fast high-quality volume ray casting with virtual samplings. Visualization and Computer Graphics, IEEE Transactions on, 16(6), 1525-1532.

[RtHRS08] Ruijters D., Ter Haar Romeny B., Suetens P.: Efficient gpu-based texture interpolation using uniform b-splines. Journal of Graphics, GPU, and Game Tools 13, 4 (2008), 61–69.

[Sch05] SCHARSACH H.: Advanced gpu raycasting. Proceedings of CESCG 5 (2005), 67–76.