Volumetric Real-Time Particle-Based Representation of Large Unstructured Tetrahedral Polygon Meshes

Philip Voglreiter, Markus Steinberger, Dieter Schmalstieg, and Bernhard Kainz

Institute for Computer Graphics and Vision,
Graz University of Technology, Inffeldgasse 16
A-8010 Graz, Austria
{voglreiter,steinberger,schmalstieg,kainz}@icg.tugraz.at
http://www.icg.tugraz.at

Abstract. In this paper we propose a particle-based volume rendering approach for unstructured, three-dimensional, tetrahedral polygon meshes. We stochastically generate millions of particles per second and project them on the screen in real-time. In contrast to previous rendering techniques of tetrahedral volume meshes, our method does not need a prior depth sorting of geometry. Instead, the rendered image is generated by choosing particles closest to the camera. Furthermore, we use spatial superimposing. Each pixel is constructed from multiple subpixels. This approach not only increases projection accuracy, but allows also a combination of subpixels into one superpixel that creates the well-known translucency effect of volume rendering. We show that our method is fast enough for the visualization of unstructured three-dimensional grids with hard real-time constraints and that it scales well for a high number of particles.

Keywords: mesh representations, volume rendering, GPU accelerated, particle-based

1 Introduction

Volume rendering is used in many disciplines and is strongly tied to visual representation of medical datasets. Irregular datasets – or unstructured grids –, are mainly used for simulations, for example for finite element analysis [2]. Rendering methods for such grids are an ongoing field of research.

Modern medical applications demand fast visualization techniques. Generation of images with interactive frame rates is essential for applications requiring visualization techniques which cater to hard real-time constraints. Furthermore, medical applications often need to provide a wide field of different techniques to visualize different modalities concurrently. One could think of image segmentation or simulations performed in parallel with rendering.

The recent developments in general computations on Graphics Processing Units (GPUs) offer the ability to solve a wide variety of parallelizable tasks

efficiently. In this paper, we introduce a novel way of stochastic Particle-based volume rendering (PBVR) exploiting these capabilities. In contrast to many other object space volume rendering approaches, basic PBVR does not require depth sorting of any kind. Instead, we treat projected particles in a way that is similar to z-buffering. Contrary to the highly sophisticated particle generation methods (Metropolis [6]) used by former approaches, we introduce a method of particle generation with little computational effort and online control of the number of generated particles. This ability is crucial for applications with hard real-time constraints and allows to alter visual effects such as density during runtime. Because the number of particles strongly influences the computational complexity, our proposed online control can also be used to steer the use of resources and thereby allocate resources for concurrent tasks. The complexity of the proposed method depends strongly on the number of particles needed. The amount of particles we need to render a given volume is strongly tied to the portion of the screen it covers while mesh complexity only shows a very minor impact. Also, the distance from the viewing camera influences the required number of particles. The screen resolution itself only plays a minor role when considering the computational complexity.

Contribution: We describe a fast method for parallel particle generation on the fly and simultaneous rendering for the visualization of large unstructured tetrahedral polygon meshes. A minimal preprocessing effort allows also to switch between volumes in real-time. We also introduce an improved method for particle superimposing by addressing perceptual issues.

2 Previous work

2.1 Unstructured mesh representation

In [1], Avila *et al.* propose an approach for direct volume rendering and define an irregular dataset rendering pipeline based on the widely used plane-sweep technique. Shirley *et al.* [12] describe a method for projecting tetrahedrons onto the image plane. The tetrahedrons need to be sorted before projection. Sorting is known to be in $O(n\ log\ n)$, inducing a super-linear increase in computational effort. Approaches like projected tetrahedrons have already been implemented on the GPU [5]. The authors exploit the capabilities of shaders and CUDA to perform depth sorting of the tetrahedrons. As an alternative approach, Challinger [4] describes a method for ray casting of unstructured grids. Ray casting generates images of a higher quality but shows an $O(n^3)$ complexity. However, ray casting offers ways to benefit from modern GPU capabilities as was shown in [15]. Still, the complexity constrains the efficiency of the algorithm. Point splatting [13] is very similar to particle-based approaches. The efficient point splatting approach described by the authors has a low memory consumption, but point splatting inherently produces artifacts in the rendering process.

2.2 Particle-based volume rendering (PBVR)

In [10], Sakamoto *et al.* describe a general approach of PBVR based on the Metropolis Method [6], a well-known Monte-Carlo algorithm [8] for random number generation. In [11], the authors go deeper into detail and consider rendering tetrahedral grids by voxelizing them. Voxelizing a tetrahedral grid can be a rather time-consuming task. The required double interpolation can result in a loss of information compared to methods using the unstructured grid data directly. For a more detailed outline of the algorithm, please refer also to a preliminary non-peer-reviewed version of this work found at [14].

3 PBVR adaptions

The main idea of PBVR is to construct a dense field of light-emitting, opaque particles inside a volumetric dataset. These particles are used to perform object-based rendering by simulating the light emission of particles. Mutual occlusion induced by completely opaque particles plays a major role during rendering. Sakamoto *et. al.* [10] describe the basic model in more detail.

The following sections will explain the proposed method step by step. Each of the subsections is to be seen as a prerequisite to the following steps. Generally, PBVR can be subdivided into two major activities. First, a proper particle distribution inside the volume needs to be established. We describe this procedure in Section 3.1. Second, in Section 3.2, we outline how to project the particles onto the image plane. We consecutively show how to generate the well-known translucent effect of volume rendering in Section 3.3.

3.1 Particle Generation

In this paper we introduce a randomized process to generate particles. It is desirable to achieve a uniform distribution over the whole volume to avoid visually perceivable artifacts. We split the volume into tetrahedral cells and perform particle generation per cell, which enables parallelization. We will show how to retain a global distribution of mean values by concatenating local uniform distributions.

Particle Distribution over Cells: We consider a maximum number of particles p_{max} for the whole volume. To accomplish uniformity in distribution, we need to determine the number of particles p_{cell} each cell may emit. We calculate this number by using the proportion of the cell volume V_{cell} to the total volume of the grid V_{grid}. Therefore, the number of particles per cell is

$$p_{cell} = V_{cell}/V_{grid} \cdot p_{max}. \tag{1}$$

A proof for Equation 1 is provided in Appendix A. By using this formula we generate particles in a cell equaling the average number of particles considering

a global distribution. Assuming that the distribution in a cell is uniform, the concatenation of several cells again comprises a new random distribution. The number of particles per cell in this new distribution simulates the average, or expected, number of particles of a real uniform random distribution throughout the whole volume.

Particle Position: We use barycentric coordinates to generate particle positions. Barycentric coordinates describe a point relative to the vertices of a polygon. In case of tetrahedrons, the barycentric notation of a point is

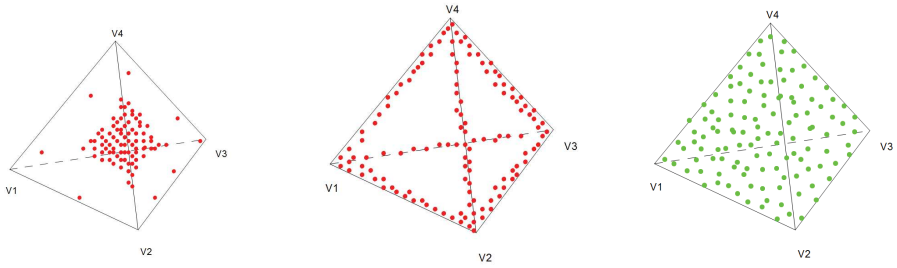
$$P = \alpha \cdot V1 + \beta \cdot V2 + \gamma \cdot V3 + (1 - (\alpha + \beta + \gamma)) \cdot V4 \tag{2}$$

where V1 through V4 denote the corner points of the tetrahedron and α, β, γ resemble the barycentric parameters. Constraining the parameters to be in the interval $(0, 1)$ and sum up to 1 restrains positions to the interior.

Uniform Patternless Generation: There are several ways to randomly generate barycentric coordinates showing statistically correct behavior, but still introducing disturbing visual patterns. Figure 1 depicts the results of incorrect approaches as well as the distribution we want to achieve. A method for correct random generation of points in tetrahedra was described by Rocchini [9]. The authors describe a method which folds a cube into a tetrahedron. First, we randomly generate α, β, γ. Next we perform a check on violation of the barycentric constraints. If a set of generated random variables is outside the given bounds, we map the particle back inside the tetrahedron as described in the article. Additionally, we use the barycentric parameters to linearly interpolate the corner points and gain the scalar values corresponding to the particles.

Particle Emission Probability: Simply projecting all generated particles would lead to an equal density throughout the whole volume. We need to thin out the particle field to accommodate for cell translucency. As we still want to avoid patterns within the rendered images, we do this stochastically per particle by applying the rejection method [8]. First, we determine the opacity a particle would anticipate. We do this in a fashion similar to conservatie volume rendering approaches, but please be aware that this is not the final opacity value of a pixel, but rather a computational variable. Next, we generate a stochastic variable x within the interval $(0, 1)$ on the real line. Only if x is smaller than the opacity of the particle, it is accepted and emitted. This adapts the number of particles per cell according to the respective cell opacities. The method allows for smooth transitions between vertices with distinct opacities.

Empty Space Skipping: Often, different approaches of volume rendering need to process regions which, according to the chosen transfer function, do not need to be displayed at all. In our approach, we can easily filter the dataset on the



(a) Particles concentrating at cell center

(b) Particles concentrating at cell border

(c) Uniformly distributed particles

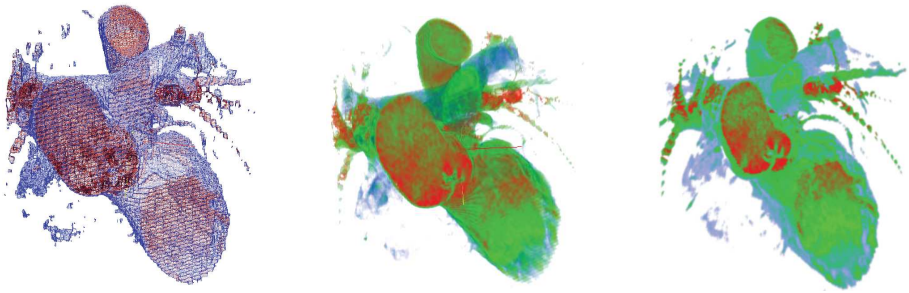
Fig. 1: Figures showing the results of different barycentric parameter generation approaches. The figures 1(a) and 1(b) both depict approaches resulting in disturbing patterns. This visual error accumulates over all cells and introduces streaks and clusters in the final image. Figure 1(c) shows the distribution we achieve when using the algorithm describe in Section 3.1.

fly. Depending on the selected transfer function, many cells anticipate a very low average opacity. This would lead to generation of particles with a rather low emission chance. To increase computational performance, we simply skip cells with a low average opacity. This prevents particle generation in those portions of the volume which would appear almost or totally translucent. This method shows an increase in performance directly proportional to the amount of cells we may neglect. This strategy is especially relevant for grids containing large connected empty regions, such as tetrahedralized CT scans. Furthermore, this approach is extremely adaptive to changes of the transfer functions.

3.2 Particle projection and Image Generation

Particle projection involves two steps. First, the screen space location of the particle needs to be determined, which requires the virtual camera parameters and the volume's transformation. Second, a color value needs to be assigned to each particle.

Projection from Object Space to Image Space: By using the modelview-projection matrix of the viewing camera we determine the image-space position of each emitted particle. Furthermore, we calculate its distance to the camera. If two particles hit the same fragment on the image plane, we choose to display the particle closer to the camera. Thereby, we effectively avoid depth sorting of particles hitting the screen as we only compare the depth values of subpixels. This approach is similar to common z-buffering.



(a) Wire-frame surface, $\frac{1}{2}$ mesh-resolution ($>$ 30fps)

(b) Projected tetrahedrons, volumetric ($<$ 0.1fps)

(c) PBVR, volumetric (16.4fps)

Fig. 2: Comparison between a wire-frame surface representation, projected tetrahedrons and PBVR for a tetrahedralized MAGIX [7] dataset. The scalars show the contrast agent enrichment of one time step of the 4D MAGIX CT scan (left ventricle, blue = low, red = high). Projected tetrahedrons takes several seconds for a full visualization. Our method remains interactive while appearing visually comparable to (b).

3.3 Spatial Superimposing

2D Superimposing: For projection we subdivide each pixel into several subpixels. The subpixel level l describes a subdivision into $l \times l$ distinct subpixels. Thereby we increase the information available for each pixel on the screen as we increase the number of contributing particles originating from different depth levels. We compose the final pixel values by calculating the average of all contributing subpixels.

3D Superimposing: To increase the visual quality of the proposed algorithm we currently use superimposing over three dimensions. We store the projected particles in multiple 2D-superimposing layers to capture more contributing particles. Using this setup, we maintain several particles per subpixel rather than only storing the closest one. Unfortunately, we need to sort the particles in this refinement step for proper insertion. The number of particles per subpixel, i.e. the number of depth layers, may be adapted to the computational needs. This strategy results in a noticeable increase in visual quality, but has a severe impact on frame rates and memory consumption. Thus, we use this method for progressive refinement during periods with no direct interaction only.

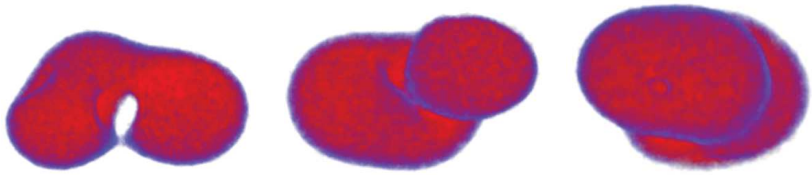
Particle Depth Enhancement: Simple averaging of subpixels may create undesired visual effects, similar to front face culling in triangle mesh rendering. This effect may hamper depth perception while rotating the volumes. Equalized treatment of particles, regardless of their depth, alleviates the impact of particles close to the camera. Consecutively, particles further away show a strong impact

on the final pixel which makes them appear too close to the camera. To support a better depth perception, we use the already present depth information of displayed particles. In detail, we analyze the depth of each pixel's subpixels z_{curr} and record their minimum z_{min} and maximum depth z_{max}. Then, we calculate the depth range and a depth ratio $\zeta = (z_{max} - z_{curr})/(z_{max} - z_{min})$, considering the gap to the maximum value. Using ζ as factor for the color values of subpixels, we achieve a linear depth evaluation of particles. We linearly blend the particles into the background considering the calculated parameter ζ. Those particles which are close to the camera sustain their influence on the color value while particles farther away partially lose their influence.

Translucency: Translucency is influenced by two parameters. Firstly, the number of generated particles influences how opaque the volume appears by altering the number of occupied subpixels. Secondly, the subpixel level increases or decreases the level of transparency in a similar way. By allowing online control of those parameters we also enable interactive adaption of frame rates while only slightly altering the translucency of the generated image.

4 Results

We evaluated our technique for several volumes including the MAGIX dataset [7] with 5 million cells, and a simulated radio frequency ablation (RFA) [3] data set with 55 thousand cells. The depicted results and frame rates were recorded on an NVIDIA GeForce GTX470 using CUDA 4.1, rendered at a resolution of 1200x800 pixels. Figure 2 shows a comparison of projected tetrahedrons [5] and our algorithm. The scalar values depict the flow inside the tissue. While the implementation of projected tetrahedrons we used is unable to cope with the sheer complexity of the dataset, our method stays responsive and even interactive during rendering. Please note that we rendered all of the shown pictures with 2D-superimposing to provide insight into the graphical capabilities of the basic method.



(a) axial (b) left (c) right

Fig. 3: Simulated RFA (55k cells) with $p_{max} = 12$million, $l = 5$ @ 29fps

Figure 3 shows an RFA simulation rendered at 29 fps. The canals inside the volume show the impact of vessels on the heat distribution (heat sinks) during

the simulation. The transfer function communicates the cell death probability where red denotes almost sure death and blue a low probability of destruction. The raw performance of our algorithm at rendering the complex MAGIX dataset is depicted in Figure 4. For reference, the image shown in Figure 2 was rendered on subpixel level $l = 3$ with 60 million particles. Even when configuring the particle number and subpixel level to be far higher than necessary, our algorithm runs with interactive frame rates.

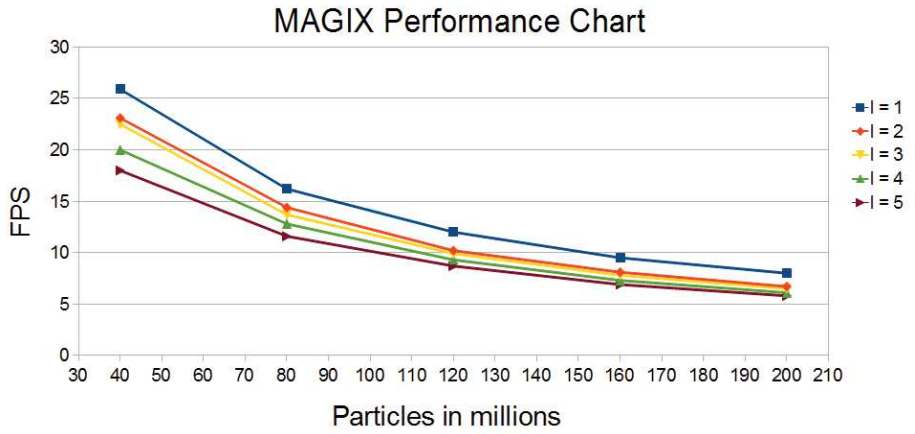


Fig. 4: Performance of the MAGIX dataset with different subpixel levels ($l = [1, 2, 3, 4, 5]$).

5 Conclusion

We have shown that our method is able to render millions of particles per second in real-time, mainly due to our fast and flexible particle generation process. We minimize GPU global memory access and increase computation occupancy. Furthermore, our approach offers a fast alternative for the real-time visualization of large unstructured tetrahedral polygon meshes. Future tasks are improved 3D-superimposing strategies to increase the image quality without a noticeable loss of render speed. Our approach will scale well for use on GPU clusters. Our method is open-source and a first version is freely available for download from our website[1]. A public link will be available by the time of the workshop.

6 Acknowledgments

This work was funded by the Austrian Science Fund (FWF): P23329.

[1] `http://www.icg.tugraz.at/project/mvp/downloads-1/GPUPBVR.zip`

References

1. R. Avila, H. Taosong, H. Lichan, A. Kaufman, H. Pfister, C. Silva, L. Sobierajski, and S. Wang. VolVis: A diversified Volume Visualization System. In *Visualization, 1994., Visualization '94, Proceedings., IEEE Conference on*, pages 31 –38, CP3, Oct. 1994.
2. I. Babuska. Generalized Finite Element Methods : Main Ideas , Results , and Perspective. *Security*, 1(1):67–103, 2004.
3. T. Bien, G. Rose, and M. Skalej. FEM Modeling of Radio Frequency Ablation in the Spinal Column. In *Biomedical Engineering and Informatics (BMEI), 2010 3rd International Conference on*, volume 5, pages 1867 –1871, oct. 2010.
4. J. Challinger. Scalable parallel Volume Raycasting for nonrectilinear computational Grids. In *Proceedings of the 1993 symposium on Parallel rendering*, PRS '93, pages 81–88, New York, NY, USA, 1993. ACM.
5. A. Maximo, R. Marroquim, and R. Farias. Hardware-Assisted Projected Tetrahedra. *Computer Graphics Forum*, 29, Issue 3:903–912, 2010.
6. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.
7. OsiriX DICOM Viewer public sample image sets. Dicom Image Sets. `http://www.osirix-viewer.com/datasets/`, May 2012.
8. C. P. Robert and G. Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.
9. C. Rocchini and P. Cignoni. Generating random points in a tetrahedron. *J. Graph. Tools*, 5(4):9–12, Oct. 2000.
10. N. Sakamoto, J. Nonaka, K. Koyamada, and S. Tanaka. Volume Rendering using tiny Particles. In *Multimedia, 2006. ISM'06. Eighth IEEE International Symposium on*, pages 734 –737, Dec. 2006.
11. N. Sakamoto, J. Nonaka, K. Koyamada, and S. Tanaka. Particle-based Volume Rendering. In *Visualization, 2007. APVIS '07. 2007 6th International Asia-Pacific Symposium on*, pages 129 –132, Feb. 2007.
12. P. Shirley and A. Tuchman. A polygonal Approximation to direct Scalar Volume Rendering. In *Proceedings of the 1990 workshop on Volume visualization*, VVS '90, pages 63–70, New York, NY, USA, 1990. ACM.
13. F. Vega-Higuera, P. Hastreiter, R. Fahlbusch, and G. Greiner. High performance Volume Splatting for Visualization of neurovascular Data. In *Visualization, 2005. IEEE*, pages 271 – 278, Oct. 2005.
14. Voglreiter, P. and Kainz, B. Stochastic Particle Based Volume Rendering. `http://www.cescg.org/CESCG-2012/papers/Voglreiter-Stochastic_Particle-Based_Volume_Rendering.pdf`, Feb. 2012.
15. C. Zhang, P. Xi, and C. Zhang. CUDA-Based Volume Ray-Casting using cubic B-spline. In *Virtual Reality and Visualization (ICVRV), 2011 International Conference on*, pages 84 –88, Nov. 2011.

A Proof of Equation 1

We show the proof for a one-dimensional distribution. The three-dimensional proof can be easily obtained by using vectors and multidimensional probability distributions.

Theorem 1. *Let n be the number of uniformly random distributed points in the interval (a, b) with $a, b \in \mathbb{R}$ to be generated. The mean value of points generated within a sub-interval $(a', b') \subseteq (a, b)$ is equal to the fraction of the size of the sub-interval to the whole interval.*

Definition 1. *The cumulative distribution function of a uniform random distribution over the interval (a, b) with $a, b \in \mathbb{R}$ is given as*

$$F(x) = \begin{cases} 0 & \text{if } x \leq a \\ \frac{x-a}{b-a} & \text{if } a \leq x \leq b \\ 1 & \text{if } x \geq b \end{cases}$$

Definition 2. *A sub-interval $(a', b') \subseteq (a, b)$ is defined as*

$$a \leq a' < b' \leq b$$

Lemma 1. *The probability of a randomly distributed variable X to be within an interval (a', b') as a subset of the original distribution over (a, b) can be calculated as*

$$P(a' \leq X \leq b') = F(b') - F(a') \tag{3}$$

Thus, we can obtain the probability of a particle to be within the sub-interval (a', b'):

$$P(a' \leq X \leq b') = F(b') - F(a') = \frac{b'-a}{b-a} - \frac{a'-a}{b-a} = \frac{b'-a'}{b-a} \tag{4}$$

The average number of particles within the sub-interval thus is given as

$$n' = n * P(a' \leq X \leq b') = n * \frac{b'-a'}{b-a} = n * \frac{V'}{V} \tag{5}$$

where V denotes the whole interval (volume), and V' denotes the sub-interval (tetrahedral cell volume). □