Binary Volumetric Octree Representation for Image-Based Rendering

Alexander Zhirkov

The MSU Computer Graphics & Media Lab
Moscow, Russia

Abstract

A Binary Volumetric Octree (BVO) is a volume array with binary opacity voxels, represented as octree. The BVO structure allows to stores not the complete object volume, but only the surface voxels. The new approach to Image Based Rendering (IBR), using this representation, makes it possible to use single data structure for fast, occlusion-compatible hierarchical warping, splatting-based rendering, and easy level-of-detail selection. This representation allows approximation of multiple depth images with guaranteed solution of the main problem of IBR methods - gaps filling (with minor conditions imposed on the originals depth images). The rendering process can be either completely CPU-based, or use hardware assisted texture mapping.

Keywords: *Image Based Rendering, Binary Volumetric Octree, 3D mipmap.*

1. INTRODUCTION

Increasing requirements for scene quality and complexity lead to necessity of processing of enormous number of polygons. Therefore, last years the IBR is growing fast. The main idea of IBR is to use available photos of the scene to produce the desired scene view. There exist various technologies of IBR that differ in method of construction of new rendering view based on given photos. One of the methods is to use Relief Textures (RT) [1], where each image with depth is put on texture-mapped polygon. The texture is warped according to view angle (pre-warping step) and then textured polygon is projected to the output buffer (post-warping). This method works well if the object is well approximated by depth function (good example is a relief wall). In more general case it is necessary to rewarp information from one polygon to another, which results in much more complex algorithm. One approach to solution of this problem is to use a single structure, uniting multiple depth maps. This can be done with the aid of Layered Depth Image (LDI) [2]. Like RT, this structure can use back-to-front warping and a special fast warping transform. Disadvantage of this technique is restriction on the allowed camera locations. A possible way out is to use overlapping LDIs, or six LDI with common camera center [3], or even special centers of LDI projection around the object [4]. Introduction of the hierarchical approach to IBR such as LDI Tree [5], QSplats [6], Hierarchical Image-based Rendering [7] allows to easily determining Level Of Detail (LOD).

The last years saw a lot of interest to joint usage of volumetric textures and polygonal rendering. The main disadvantage of such structures in increasing size of storage, growing like n^3. One of solutions to this problem is a set of bounding boxes with small 3D textures in each [8].

Using single BVO structure for both geometry and color data increasing memory requirements with growing like n^2, but preserve volumetric structure. In contrast to Binary Space-Partitioning Trees, BVO stores only "on-the-plane" voxels, and does not contain information for returning to polygonal surface model. This give possibility of approximation not only polygonal, but also IBR and Point-based representations (see section 2).

The main features of the suggested method of IBR are:

- Fast CPU-based warping. The coordinate transformation for one voxel takes about two integer addition, indexing and division (in perspective projection case). Rendering process is performed in occlusion-compatible order, and no z-buffer is needed (section 3.1).
- Volumetric octree representation contains 3D mipmaps (or LODs), that allow to select sizes and number of elements in proportion with output buffer pixel size (section 3.2).
- Using fixed splat size with subpixel output buffer provides anti-aliasing in both coordinate and time axis (section 3.2).
- Polygonal model are not used in warping stage. Texture mapping is used only for compatibility with other models in the scene (section 3.3).
- Compact representation. The warping process operates with octree in binary form, were voxel coordinates are stored implicitly. Therefore, coordinates of a filled voxel occupy, on average, less than 3 bits.

2. APPROXIMATION BY BVO

2.1 Approximation of Points

Let n be the octree height. With similarity transformation put all the points into the cube with edge length equal to 2^n. We'll store node of octree only if corresponding cube contains at least one point (Figure1). Set the node color equal to arithmetical mean of colors of points contained in corresponding cube.

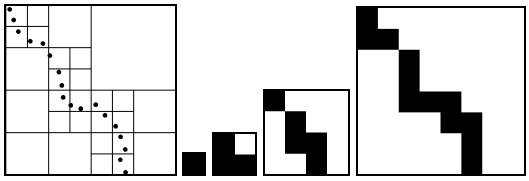


Figure 1. BVO approximation of points with n=0,1,2,3 (2D projection view)

2.2 Approximation of LDI and Multiple Depth Images

Each of these structures can be represented as a set of 3D colored points in unified orthogonal coordinate system and converted to BVO as in 2.1.

2.3 Approximation of Polygonal Model

Put all vertexes into a bounding cube. Represent all polygons as set of triangles. We'll recursively split each of the triangles into four similar triangles, and continue doing so while there exist a triangle with side whose length is greater than one. Treating all the obtained vertices as color points we create binary volumetric octree like in 2.1.

2.4 Continuity preservation condition

Condition of continuous visualization of voxels is 'continuity' of nearby voxels visualization. Two voxels are called neighbours if their edges have common point. Hence, every voxel (except extremes) has 26 neighbours. Let us examine how this condition is mapped on continuity of source representation.

2.4.1 Multiple depth images

The source of depth image can be of different nature: real photos with laser distance map, ray-casting of volumetric model, ray-tracing of polygonal model, z-buffer with image, etc. We'll examine only the case of visualization of Lambert opacity surface fragment from several cameras in orthogonal projection.

Sufficient condition of continuity preservation of surface fragment. All three conditions must be satisfied:

1) The fragment is completely visible by one of the cameras
2) The angle between this camera view direction and surface normal at every point of the surface fragment must not exceed 45'
3) Pixel side length in depth image resolution corresponding to this camera does not exceed volume edge size.

Necessary condition: the fragment is visible from one of cameras viewpoints

2.4.2 Opacity polygonal model

From the algorithm 2.3 it follows that if two polygons are continuously connected, then corresponding volumes are connected through neigh-boring voxels. The opposite is not true: two unconnected triangles that are closer to each other than voxel edge length will be joined into neighbours voxels.

3. RENDERING

3.1 Recursive transform computation

Detailed description of hierarchical transformation computation and back-to-front display order algorithm based on octree coordinates decomposition was described in object-order volumetric rendering techniques [9,10]. This process makes modification because of specialized BVO representation.

Let n be octree height, T - transformation matrix 4x4, v - normal coordinates of voxel. Assume that voxel coordinates are stored (as noticed in introduction) in packed implicit form, and rewrite node coordinate as (1) and node transformation as (2).

$$v = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} (\sigma_x^n ... \sigma_x^2 \sigma_x^1) \\ (\sigma_y^n ... \sigma_y^2 \sigma_y^1) \\ (\sigma_z^n ... \sigma_z^2 \sigma_z^1) \\ (0...01) \end{pmatrix} = \sum_{i=1}^{n} 2^{i-1} \sigma^i \quad (1)$$

$$Tv = \sum_{i=1}^{n} T\, 2^{i-1} \sigma^i \quad (2)$$

Let *f* be any term in the sum (2). With fixed *T* it can be computed at frame preprocessing step in a table look-up fashion (3).

$$f_T(i, \sigma^i) = T 2^{i-1} \sigma^i \quad (3)$$

Rewrite objective function in convenient for the recursive computation form:

$$F_i = \sum_{j=i}^{n} f_T(i, \sigma^j) =$$
$$= f_T(i, \sigma^i) + F_{i+1} = F_i(\sigma^i, F_{i+1})$$
$$Tv = F_1(\sigma^1, F_2(\sigma^2 .. F_n(\sigma^n) ..)) \quad (4)$$

Compare number of operations for hierarchical warping transform computation (4) and for equivalent direct 3D warping in parallel projection case (Table 1). For these purpose we need to roughly calculate number of nodes in BVO. Let p be the number of opacity cubes on n-th 3D mipmap level. Because we use method of surface approximation, p is directly proportionate to the area of approximated surface, therefore $p \sim n^2$. Hence the total number of nodes is

$$p + \tfrac{1}{4} p + ... + \left(\tfrac{1}{4}\right)^n p = (1 - \left(\tfrac{1}{4}\right)^n) / (1 - \tfrac{1}{4}) < \tfrac{4}{3} p$$

For comparison let us assume that $p = kn^2$, where coefficient k depends on model and typically is in range 0.5<k<2.

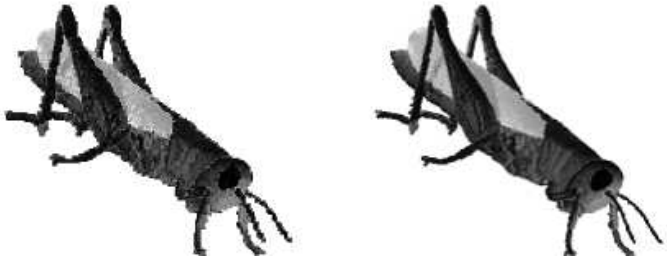
Table 1. Computation cost of normal coordinates transformation

	Multiplication	Addition	Indexing

these splats require the large amount of computation. Using

Model	3D Mi...	Number of O...	Frames per second	
			Sampling Mode 1	Sampling Mode 2
Angel	8	120183	27	18
	7	28615	97	67
Head	8	126997	26	17
Grasshopper	8	32305	54	33

Figure 3. Successive 3D mipmaps (8-th, 7-th, 6-th)



Mode 1: splat 2x2 Mode 2: splat 3x3, subpixel 0.5

Figure 4. Different sampling modes

simple splats like small single-color opacity boxes leads to aliasing, noticeable at model borders and time-aliasing, noticeable when the viewing position slightly changes. These artifacts can be suppressed by using subpixel level. Let *SPL* subpixel level. Let us analyze conditions on opacity box splat size:

$$splatsize \geq \lceil L(\varphi,\psi)VPP/SPL \rceil SPL$$

$$\sqrt{2} < L(\varphi,\psi) < \sqrt{3}$$

Here L is the length of voxel diagonal projection and depends on the angle at which the voxel is viewed. As discussed in the Introduction, the best quality/speed ratio is achieved when volume edge size $\approx$ pixel edge size i.e. $VPP \approx 1$. Imposing also a condition that the splat size must be identical for all volume resolutions, we have only two sampling methods satisfying these assumptions:

- *Sampling mode 1*: SPL=1, splat 2x2, $0.57 < VPP \leq 1.14$

Direct 3D warping	$16kn^2$	$12kn^2$	0
Hierarchical 3D warping	$128n$	$\lvert \frac{8}{3}kn^2 \rvert + 96n$	$\lvert \frac{8}{3}kn^2 \rvert$

3.2 Pre-Warping

The basic IBR problem occurs at resampling step and is called gap-filling problem. There exist two base methods for solving this problem: linear interpolation between points (used in RT) and splatting (most LDI based methods, QSplats). In our approach, we used splats because of its predetermined size. The size of splat in volumetric based rendering must be sufficient to cover the corresponding volume. This size depends on linear sizes of voxel and pixel. Let *VPP* = voxel edge/pixel size. The splats also differ in type of kernel. The complex high quality splats are used, in Volumetric Rendering most with Gaussian kernel [11], in Point Rendering systems they are ellipses [6] and in IBR methods mesh-splats are frequently used [12]. However,

- *Sampling mode 2*: SPL=2, splat 3x3, $0.44 < VPP \leq 0.88$

The difference in speed and quality of these rendering methods is shown in Table 2 and Figure 4.

3.3 Post-Warping

Unlike RT method, the Post-Warping procedure is needed mostly to provide compatibility of BVO representation with other IBR or Polygonal models in common scene.

For this purpose, a square texture is introduced into the scene, lying between the camera and BVO object. This texture plane must be parallel to the camera screen (Figure 2). The binary logarithm of the side of this textured square is equal to the maximum number of levels-of-detail ("3D mipmapping levels") used in pre-warping step.

For smooth change between LODs, we do not use standard 2D-mipmaps for trilinear filtering, but projections of 3D-mipmaps instead (see the visual difference between 2D and 3D mipmaps in Figure 5).

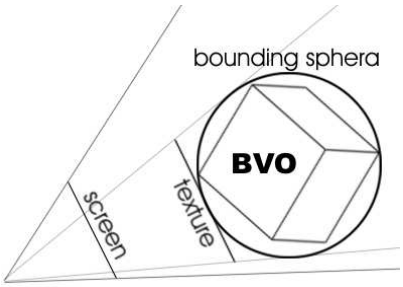


Figure 2. Integrating BVO model into a scene

4. RESULTS AND CONCLUSION

We would like to demonstrate speed and rendering quality of three models obtained from IBR-representations. The first two models were obtained from orthogonal LDI, and the last one – from six images with depth. Tests were run on Intel Celeron 500MHz. Table 2 shows frame rates with different 3D mipmap level and different *Sampling Modes* (see section 3.2). The tests are illustrated in Figure 3 and Figure 4.

Table 2. Software rendering times for different sampling modes

Apart from using for IBR, BVO structure can be also used in hardware-accelerated polygonal rendering. It can achieve the greatest efficiency by replacing polygonal models at long distances, where triangle sizes becomes comparable with pixel output resolution.

5. ACKNOWLEDGMENTS

Samsung Advanced Institute of Technology (SAIT) was supported this research under the "Advanced 3D Rendering Technology" project. I wish to express my gratitude to Yuri Bayakovski and Leonid Levkovich-Maslyuk for helpful discussions.

6. REFERENCES

[1] Manuel M. Oliveira, Gary Bishop, McAllister. Relief Textures Mapping. *SIGGRAPH 2000.*

[2] Steven Gortler, Li-wei He, Richard Szeliski. *SIGGRAPH '98*

[3] Manuel M. Oliveira, Gary Bishop. Image-Based Objects. *SIGGRAPH '99.*

[4] Rademacher, Paul and Gary. Multiple-Center-of-Projection Images. *SIGGRAPH'98.*

[5] Chun-Fa Chang, Gary Bishop, Anselmo Lastra. LDI Tree: A Hierarchical Representation or Image-Based Rendering. *SIGGRAPH'99.*

[6] Szymon Rusinkiewicz, Marc Levoy. QSplat: A Multiresolution Point Rendering System for Large Meshes. *SIGGRAPH 2000.*

[7] Hierarchical Image-Based Rendering using Texture Mapping Hardware. *Proceedings of the Eurographics Workshop on Rendering '99.*

[8] Rüdiger Westermann, Ove Sommer, Thomas Etrl. Decoupling Polygon Rendering from Geometry using Rasterization Hardware. *Rendering Techniques '99*

[9] A. Li and G. Crebbin. Octree Encoding of Objects from Range Images. *Pattern Recogniton 1994.*

[10] Frieder G., Gordon D. and Reinolds. Back-to-Front Display of Voxel-Based Objects. *IEEE Computer Graphics and Applications, 1985.*

[11] L. Westover. Footprint evaluation for volume rendering. *SIGGRAPH'90.*

[12] William R. Mark, Leonard McMillan, Gary Bishop. Post-Rendering 3D Warping. *Symposium on Interactive 3D Graphics1997.*

About the author

Alexander Zhirkov is a graduate student of Computational Mathematics and Cybernetics Dept., MSU.

E-mail: **zh@graphics.cs.msu.su**