A Framework for Depth Image-Based Modeling and Rendering

Alexey Ignatenko, Anton Konushin
Department of Computational Mathematics and Cybernetics
Moscow State University, Moscow, Russia
ignatenko@graphics.cs.msu.su, ktosh@zmail.ru

Abstract

We present an image-based system for automatic modeling and interactive rendering of 3D objects. We describe our contribution to image-based modeling and interactive multi-resolution rendering algorithms. Our representation is based on images with depths, which allow it to be compact and flexible, suitable for static and animated scenes, simplify streaming network transmission. The representation has been proposed and accepted into MPEG-4 AFX (Animated Framework eXtension).

Keywords: Image-based rendering, Image-based modeling, MPEG-4, Depth Images, Splatting.

1. INTRODUCTION

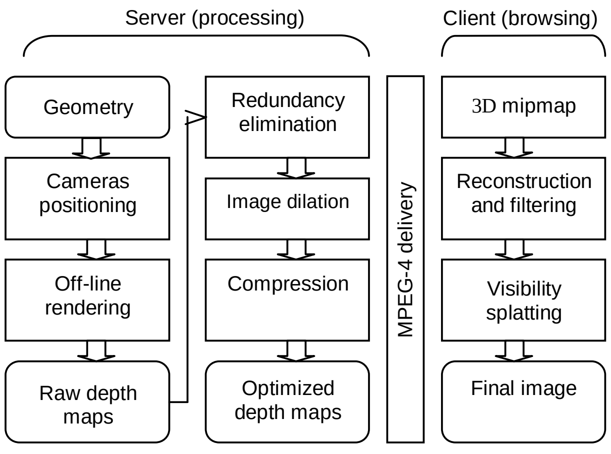
This paper describes a representation and a set of algorithms for image-based modeling and interactive rendering of high-quality 3D objects with precomputed illumination.

Recent multimedia applications demand visualization of 3D models with high quality in network environment, on different client devices, including mobile. Polygonal meshes are not well suited for this task due to redundancy of connectivity information in some cases and complex level-of-detail, compression, progressive transmission algorithms. Image-based graphics could present a solution for these problems. We have developed a framework that utilizes novel image-based and point-based techniques to obtain high-resolution 3D object representation based on images with depth. The framework is capable to process the data, make editing operations if necessary, deliver the model through the network, and render it in real-time.

The concept of using images to decrease rendering complexity and capture precomputed illumination is well known in graphics. Image-based rendering techniques proved their viability in different tasks. Images were used in image caching algorithms ([1]) as a replacement for rendering of distant geometry. In order to extend image-based concept to handling objects situated near the observer, several approaches were made on extending images with corresponding depth information [2]. Image-based rendering methods with image warping approaches were used for interactive browsing ([3], [4]). LDIs ([3]) (Layered Depth Images) were introduced in order to solve reconstruction problem in case of multiple reference images by pre-warping all initial samples into multi-valued depth map corresponding to single projection of 3D object. LDI allows keeping all parts of the surface, including invisible from the reference viewpoints.

Light-field methods ([14]) can produce high-quality rendering, but require a lot of reference images, which restricts the use of such techniques to non-animated objects with limited sizes.

The recent trend is to use point-based primitives instead of images. Point primitive has lower computational cost than triangle, therefore the use of points especially effective in complex scenes ([5], [6], [7], [8]). An important advantage is that points are more convenient for graphics pipeline and supported by hardware on most platforms.



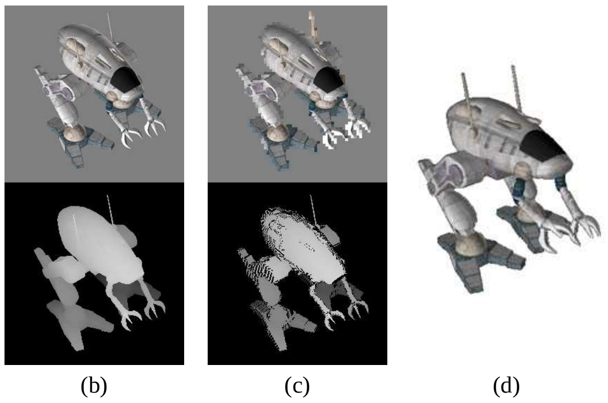


Figure 1: (a) Overview of the system: modeling, preprocessing, and rendering stages. (b) Example of initial color and depth map. (c) After optimization (d) Rendering results.

Our framework obtains object appearance and geometry information using several renderings of the object from different viewpoints in off-line rendering package (e.g. Discreet 3DS MAX). We prepare and save the information in several depth images using z-buffer. We place several cameras around the possible view directions of the model and produce high-quality renderings from each camera. Resulting data is a set of color images with depth - so-called DIBR (Depth-Image Based Representation). This raw data is processed to prepare object description suitable for real-time rendering. We eliminate redundant samples with cleanup procedure; apply special processing to color images in order to allow lossy compression. Then we compress color maps with JPEG or wavelet-based algorithm. Transmission to client is done using MPEG-4 mechanisms. Then model can be visualized in interactive mode using point-based algorithms and OpenGL-compliant hardware. Figure 1 shows the process as a whole.

This paper is organized as follows. In the next section we describe model representation, developed for MPEG-4 AFX. The following sections describe various stages of the process. In Section 3 we describe modeling, preprocessing, and compression of the data. Rendering algorithms are described in Section 4. In Section 5 we give experimental results for objects in DIBR format. Conclusion is given in Section 5.

2. REPRESENTATION

We represent each object in the scene by a set of reference images, which cover visible surface of the object. Each reference image is accompanied by additional information about geometry (depth image) and camera parameters. Common camera parameters are used: position, orientation, field of view, near and far clipping planes. In case of orthographic camera width and height of camera view field are stored. Each pixel in gray-scale depth image represents a projection of some part of object surface. Pixels with intensity 0 are considered transparent - i.e. representing holes in the object. Reference images can be stored in different formats, including ones that can be progressively transmitted over network (e.g. with wavelet compression).

Each depth image contains its part of original model. Several depth images combined together represent a 3d object. There are no restrictions on sizes, position and orientation of reference images.

Static version of our representation is directly obtained from range data or off-line renderings and can be visualized either immediately or after applying optimization techniques.

Also we introduce animated version that consists of two streams: color stream and depth stream. Animated object therefore is represented as a set of synchronized color and depth streams. This representation is useful for 3d movie-like applications [9]. Animated image-based representation has numerous advantages over traditional key-frame animation, especially on complex scenes: no limitations on animation type, complexity-independent rendering, and small size due to highly effective MPEG-4 video compression algorithms.

Flexibility and simplicity of our representation, support for animated data, allowed it to be accepted into MPEG-4 AFX ([9], [10], [11]) as a part of Depth-Image Based Representation proposal [12].

3. DATA AQUISITION AND MODELING

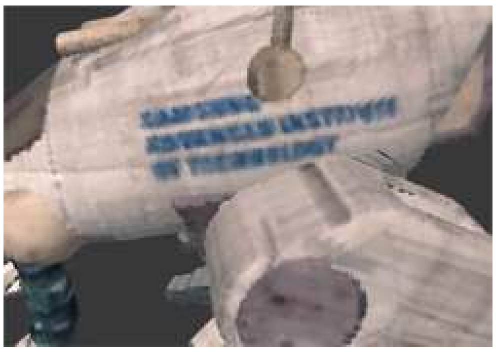
To create a model we have to obtain several images with depth containing different views of the object. The images of real world objects can be acquired by range scanning hardware or produced as an output of shape reconstruction techniques.

Synthetic objects can also be efficiently modeled with image-based representations. The images of the model can be created using off-line 3d modeling and rendering systems.

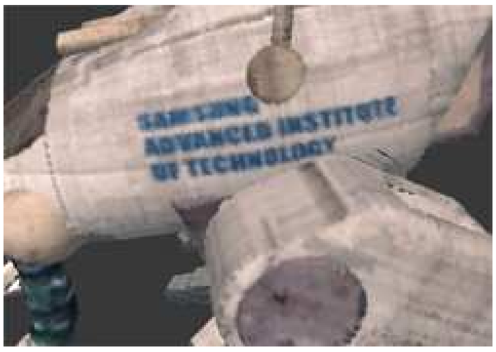
To create the DIBR model we should place a set of reference cameras around the model in 3D rendering system. For each reference camera the image together with Z-buffer snapshot is produced. The number of cameras and their orientation depend on the shape of the object and on field of application. Our system allows automatic and manual positioning of cameras. In manual mode user can freely tune number and positions of cameras. Resulting object will be a union of all samples visible from all cameras. The process of automatic camera positioning arranges pre-defined number of cameras (usually 6, 12, or 24) evenly in space of possible viewing directions. In order to increase model quality and capture some internal details, that are not visible from cameras in a standard camera configuration, more cameras can be added manually. The versatility of DIBR allows placing several cameras with different resolution and orientation to capture separate object parts with different sampling rate.

In practice this stage usually requires little user effort, because approximation of the model can be obtained with automatic camera positioning, and then refined in one-two steps, if necessary. User is free to select desired resolution of images, therefore to control quality of the model. Usually 6 to 8 cameras represent the model of an object with 256x256 (low quality) to 1024x1024 (high quality) images.

Because each reference camera observes only a part of object surface, several depth maps are required to form a complete model. In this case samples reconstructed from the different depth maps may almost coincide in space. These samples capture the same point on the surface of the object. Such redundant samples cause several problems: increased size of the model, rendering artifacts due to reconstruction rounding errors and color inconsistency of samples from different depth maps. To overcome

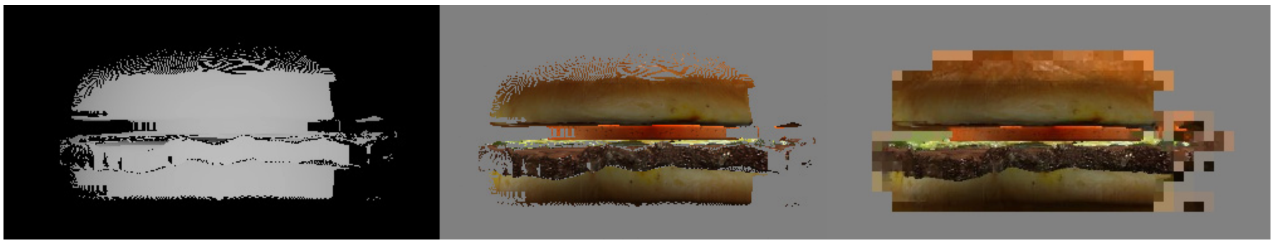


(a)



(b)

Figure 2: Example of depth map cleaning. (a) – 'Robot' model rendered before clean. (b) – after clean. Note improved text sharpness.



(a) (b) (c)

Figure 3: Dilation of image color for JPEG compression. (a) Initial depth image after cleaning (holes correspond to removed redundant pixels). (b) Color image correspond to the depth map - the holes will produce visible artifacts after JPEG compression. (c) Dilated image - holes were filled by nearby colors, color changes become smoother.

these problems we use novel algorithms described in the following sections.

3.1 Redundancy elimination

The task of redundancy elimination stage is to minimize the number of object surface samples duplicated in different depth maps. The sample is considered redundant if there is another sample or set of samples capturing the same part of surface with higher quality. Here we describe our algorithm for detecting such samples. It is applicable for depth maps created with orthographic cameras.

For each pixel of each depth map we restore the corresponding 3D sample position and compute a set of additional attributes: index of source depth map, position (x,y) in this map, sample size, normal vector, and so-called 'rating'. Rating characterizes the sampling quality at a given point in a given depth map. For the pixel in the depth map its rating is inverse proportional to corresponding surface sample area:

$$r = \frac{(N_p, N_s)}{A_p}$$

where N_p is a normal to camera viewing plane, N_s is a surface normal at the given sample, and A_p is a the maximum projected area of the splat. $(\cdot)$ is vector dot-product).

All samples located close enough to each other in 3D (i.e. those representing, effectively, the same 3D sample) are compared by their respective rating. One sample with the highest rating is kept and others are removed from their depth maps. To eliminate a redundant sample the corresponding pixels made black in the depth map. We also consider cases when sizes of samples are too different that one sample entirely covers another. In this case we remove large samples that contain smaller ones.

The result of this procedure is non-redundant representation, where each sample of the object surface is captured by only one map. Results show that cleanup procedure leads to reduction of model size and increase of rendering quality (see Figure 2). Rendering is also faster due to smaller number of samples need to be processed (see Table 1).

3.2 Compression

Before transmission to a client machine, color and depth images should be compressed in order to decrease download time. Lossy image compression methods offer much higher compression ratio compared to lossless algorithms without sacrificing much in image quality. But pixel colors are changed after lossy compression. These changes are barely noticeable for a human observer but break color consistency condition for DIBR models that result in noticeable rendering artifacts.

We propose the solution called color dilation. The main idea is to change the background color near object silhouette in color map, without modification of the depth map. Actual samples are identified by non-empty pixels of the depth map, so geometry will be the same after dilation. The new colors are calculated depending on the compression scheme (block-based for JPEG, continuous for wavelet) and on colors of the neighboring object silhouette pixels. During the compression border pixels are mixed with colors of updated background. Due to color dilation difference of pixel colors before and after compression is much lower (see Figure 1).

4. RENDERING

Our visualization software uses OpenGL to render the scene at interactive rates. We deal with an animation frame-by-frame. Before rendering a frame, we select a desired quality of the depth maps. We choose the quality of a depth map proportionally to the visible size of its bounding box. Image is resampled and 3D positions of all samples are restored. Then data is projected onto the viewing plane and the final image is reconstructed. Resampling of the reference depth image allows to maintain desired frame rate.

We implemented two reconstruction methods based on splatting [13] idea. First reconstruction method uses GL_POINTS as rendering primitive. This approach is fast, but the quality depends on precision of point size calculation. Point size calculation is implemented using hardware shaders. This allowed us to precisely calculate size of each point depending on distance to the viewer and camera parameters. We approximate required parameters by estimating projected sizes of several reference points.

Further improvement of quality is achieved by the second scheme, using more complex splatting. Instead of uniformly colored disc splats, it uses discs with Gaussian intensity distribution. In order to archive correct blending of nearby points, we use two-pass rendering [7]. This method allows achieving higher quality, but it is near two-times slower due to multi-pass rendering of each frame.

An advantage of our approach is simple level of detail control by resizing of each image to some needed extent. In our implementation this is done at client size after downloading of the image in full size. Using MPEG-4 backchannel this resizing can be done by server, thus greatly saving bandwidth.

	Hamburger	**Robot**	**T-Rex**
Number of textures	6	6	12
Number of samples	237 412	411 723	452 462
Uncompressed raw size	7,5mb	7,5mb	15mb
Compressed raw size (png)	1,09mb	1,66mb	2,68mb
Texture dims	512	512	512
Raw rendering speed	29 fps	17 fps	16 fps
Preprocessing time (m:s)	00:06	00:07	00:31
Optimized number of samples	191 988	293 889	312 780
Dilated compressed size (png, jpg)	279k	436k	735k
Optimized rendering speed (point, gauss)	35(13)	23(11)	23(12)

Table 1: Storage requirements, preprocessing and rendering time for DIBR models.

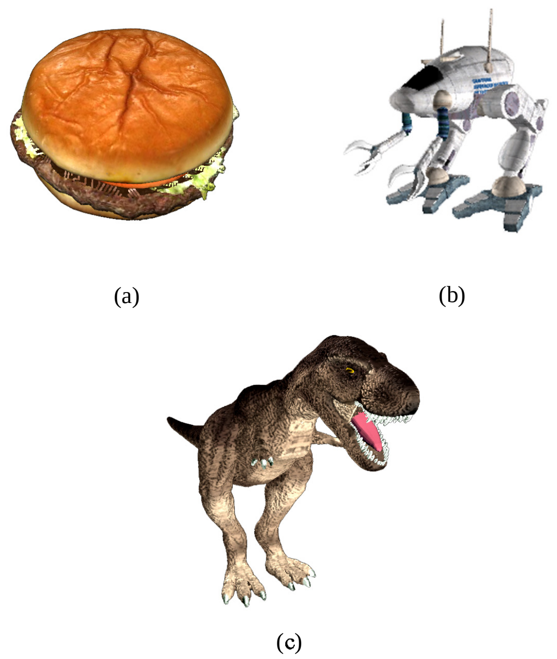


Figure 4: Interactive rendering of DIBR models. (a) Hamburger (b) Robot (c) T-Rex

5. RESULTS

We implemented and tested DIBR framework on various 3D models with different shapes and complexity. Sizes and processing time for several models in DIBR format is present in Table 1. The result of interactive rendering for different models is shown in Figure 4. Tests were performed on Intel Pentium IV 1500Mhz with NVidia GeForce3 TI 200 accelerator. Rendering was done into 1024x1024 window. The quality and speed of rendering is quite good for various rendering shapes.

6. CONCLUSION

In this paper we presented an image-based system, which main purpose is modeling and rendering of 3D models in image-based format, which has been accepted into MPEG-4 AFX. The representation is based on images with depth and has static and animated versions. We contribute a set of optimization techniques, which allow storing the model with high compression ratio and rendering it in real-time with high quality.

7. REFERENCES

[1]J. Shade, D. Lischinski, D. Salesin, T. DeRose, and J. Snyder. Hierarchical Image Caching for Accelerated Walkthroughs of Complex Environments. *SIGGRAPH '96 Proceedings*, 1996.

[2] L. McMillan. An Image-based Approach to Three-Dimensional Computer Graphics. *Univ. of North Carolina, Computer Science, Ph.D. Dissertation*, 1997.

[3]J. Shade, S. Gortler, L. Hey, R. Szeliski. Layered Depth Images. *SIGGRAPH '97 Proceedings*, 1997.

[4] G. Bishop, M.M. Oliveira. Relief Textures. *SIGGRAPH '2000 Proceedings*, 2000.

[5] Mark Levoy and Turner Whitted. The Use of Points as a Display Primitive'. *Technical Report 85-022, University of North Carolina*, 1985.

[6] J. P. Grossman and W. Dally. Point Sample Rendering. *Proc. Of Eurographics Workshop on Rendering*, 1998.

[7]L. Ren, H. Pfister, M. Zwicker. Object Space EWA Surface Splatting: A Hardware Accelerated Approach to High Quality Point Rendering. *Eurographics '02 Proceedings*, 2002.

[8] D. K. McAllister, L. Nyland, V. Popescu, A. Lastra, C. McCue. Real-Time Rendering of Real World Environments. *Rendering Techniques '99, Proc. of Eurographics Workshop on Rendering*, 1999.

[9] ISO/IEC JTC1/SC29/WG11 14496-1, Coding of Audio-Visual Objects: Systems.

[10] ISO/IEC JTC1/SC29/WG11 14496-2, Coding of Audio-Visual Objects: Visual.

[11] ISO/IEC JTC1/SC29/WG11 N5397: FDIS of ISO/IEC 14496 Part 16: Animation Framework eXtension (AFX), Awaji Island, December 2002.

[12] Y. Bayakovski, L. Levkovich-Maslyuk, A. Ignatenko, A Konushin, D. Timasov, A. Zhirkov, Mahnjin Han, In Kyu Park. Depth Image-based representation for static and animated 3D objects. *Proc. of Intl. Conf. on Image processing*, 2002.

[13] L. Westover, Footprint Evaluation for Volume Rendering. *SIGGRAPH '90 Proceedings*, 1990.

[14]Wei-Chao Chen, Jean-Yves Bouguet, Michael H. Chu, R. Grzeszczuk. Light Field Mapping: Efficient Representation and Hardware Rendering of Surface Light Fields. *SIGGRAPH '02 Proceedings*, 2002.