DEPTH IMAGE-BASED REPRESENTATIONS FOR STATIC AND ANIMATED 3D OBJECTS

Y. Bayakovski, L. Levkovich-Maslyuk**, A. Ignatenko*, A. Konushin*, D. Timasov*, A. Zhirkov*, Mahnjin Han***, In Kyu Park****

* Dept. of Computational Mathematics and Cybernetics,
Moscow State University,
Moscow, 119899 RUSSIA
{yurib, ignatenko, azh}@graphics.cs.msu.su

** The Keldysh Institute of Applied Mathematics,
Russian Academy of Sciences,
Moscow, 125047 RUSSIA
levkovl@spp.keldysh.ru

*** Multimedia Lab.,
Samsung Advanced Institute of Technology,
Yongin, 449-712 KOREA.
{manjini, saitpik}@sait.samsung.co.kr

ABSTRACT

We describe a novel depth image-based representation (DIBR) that has been adopted into MPEG-4 Animation Framework eXtension (AFX). The idea of this approach is to build a compact representation of a 3D object or scene without storing the geometry information in traditional polygonal form. The main formats of the DIBR family are SimpleTexture (an image together with depth array), PointTexture (a view of a scene from a single input camera but with multiple pixels along each line of sight), and OctreeImage (octree data structure together with a set of images and their viewport parameters). The designed node specifications and rendering algorithms are addressed. The experimental results show the efficacy and fidelity of the proposed approach.

1. INTRODUCTION

This paper deals with depth image-based representations (DIBR) of still and animated 3D objects. Instead of a complex polygonal mesh, which is hard to construct and handle for realistic models, image-based methods represent a 3D object (scene) as a set of reference images completely covering its visible surface. This data is usually accompanied by some kind of information about the object geometry. For the methods we consider here, each reference image comes with a corresponding depth map, an array of distances from the pixels in the image plane to the object surface. One of the advantages of such a representation is that reference images can provide high quality visualization of the object without direct usage of its complex polygonal model. In addition, rendering time is proportional to the number of pixels in the reference and output images, but in general, not to the geometric complexity as in the case of polygonal models.

Most popular approaches in the field of IBR were suggested in [4][5]. *Relief Textures* (*RT*), introduced in [4] are single images with depth maps. *Layered Depth Images* (*LDI*) [5] use multi-valued depth maps and images, corresponding to single projection of 3D object. Colors and distances are stored for all intersections of each ray of projection with the object. This allows representing all parts of the surface, including those invisible from the single viewpoint.

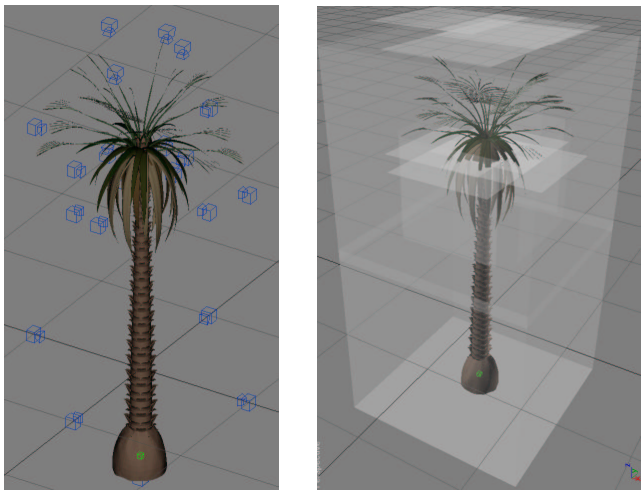
In this paper, new representations are proposed so as to combine advantages of different ideas suggested in the literature, providing a user with flexible tools best suited for a particular task. Our SimpleTexture format under DepthImage provides universal approach to painless usage of many reference images, adaptively positioned so as to capture complex shape of still and animated objects. PointTexture format under DepthImage combines the information of both visible and invisible parts of the object in a single data structure that can be efficiently compressed in either lossless or lossy mode. OctreeImage format is even more efficient in terms of storage, providing also progressive mip-mapping capabilities. The proposed representation techniques have been adopted into MPEG-4 Animation Framework eXtension (AFX) [3] as a part of MPEG-4 [1][2].

The paper is organized as follows. In Section 2 we describe the depth image-based representations (DIBR), developed for MPEG-4 AFX. Formal specifications of MPEG-4 DIBR nodes are given in Section 3. In Section 4 we describe the rendering method briefly. In Section 5 we demonstrate the rendering results of still and animated 3D objects in DIBR formats. Finally, we conclude in Section 6.

2. DEPTH IMAGE-BASED REPRESENTATION

Taking into account the ideas outlined in the previous section, as well as some of our own developments [7], we suggested the following set of image-based formats for use in MPEG-4 AFX: DepthImage with SimpleTexture or PointTexture and OctreeImage. Note that SimpleTexture and OctreeImage have animated versions.

Fig. 1. Example of Simple Textures. Six pairs of images and depth maps are used to render the model as shown in the center.



(a) (b)

Fig. 2. Generalized Box Texture (GBT). (a) Camera locations for 'Palm' model. (b) Reference image planes for the same model (21 SimpleTextures are used).

2.1 Static Data Representations

2.1.1 DepthImage

SimpleTexture is a single image with depth, which is similar to 'sprite with depth' and PointTexture is equivalent to LDI [4].

With SimpleTexture and PointTexture as building blocks, we can construct a variety of representations using MPEG-4 constructs. Formal specification will be given in Section 3, and here we describe the format geometrically.

DepthImage structure defines either SimpleTexture or PointTexture together with position in space and some other information. A set of DepthImages can be unified under a single structure called Transform node, and this allows us to construct a variety of useful representations. Most commonly used are the two of them that do not have a specific MPEG-4 name, but in our practice we called them Box Texture (BT), and Generalized Box Texture (GBT). BT is a union of six SimpleTextures corresponding to a bounding cube of an object or a scene, while GBT is an arbitrary union of any number of SimpleTextures that together provide a consistent 3D representation. Example of BT is given in Fig. 1, where reference images, depth maps and the resulting 3D object are shown. BT can be rendered with the aid of incremental warping algorithm [4], but we use different approach applicable to GBT as well (details of this rendering method are given in Section 6). An example of GBT representation is shown in Fig. 2, where 21 Simple-Textures are used to represent a complex object, the palm tree. It should be noted that unification mechanism allows, for instance, to use several LDIs with different cameras to represent the same object, or parts of the same object.

Hence, data structures like image-based objects, cells of LDI tree, cells of surfels-based tree structure [8]-[10], are all particular cases of this format, which obviously offers much greater flexibility in adapting location and resolution of SimpleTextures and PointTextures to the structure of the scene.

2.1.2 OctreeImage

The last of our formats is OctreeImage. This representation consists of the two main components – Binary Volumetric Octree (BVO), that represents geometry, and a set of reference images. Geometric information in BVO form is a set of binary (occupied/empty) regularly spaced voxels combined in larger cells in usual octree manner. This representation can be easily obtained from DepthImage data through the intermediate 'point cloud' form, since each pixel with depth defines a unique point in 3D space. Conversion of the point cloud to BVO is illustrated in Fig. 3. An analogous process allows to convert polygonal model to BVO (see [7] for details). Only the orthographic DepthImage models can be used for building OctreeImage (SimpleTextures and PointTextures can be built for both perspective and orthographic projections). Color is processed as follows. Each point in the cloud inherits its color from the corresponding SimpleTexture or PointTexture. Number of tree levels of BVO should be chosen so that voxel size would roughly correspond to a pixel on the rendered image. Then the color of each voxel is set to average color of all the points within it. Note that coarser levels of detail (mip-mapping) can be obtained from this model by simply discarding several finest subdivision lev-

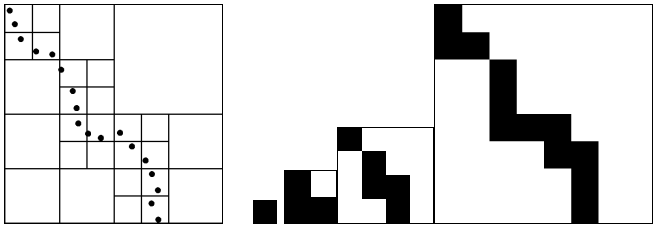


Fig. 3. Octree representation and mip-mapping for a 'point cloud' (illustrated in 2D).

els, while setting colors of octree cells at coarser level to the average color of their subcells ('children').

Object representation as an octree with colors attributed to nodes is very convenient for rendering with the aid of splats, because splat size is easily computed from voxel size. But this is a memory-inefficient format. For storage/transmission it is transformed to much more compact form, as follows:

Octree structure is first converted to breadth-first traversal linkless form, where each node is represented by a single byte whose bits indicate if the corresponding subcube is further subdivided. Color information is stored as a set of reference images that are obtained by projecting the colored voxel representation onto the predefined image planes. This type of separation of color and geometry makes OctreeImage suitable for animation purposes.

2.1.3 Animated Versions of the Formats

Animated versions were defined for two of the DIBR formats: DepthImage containing only SimpleTextures, and OctreeImage. Data volume is one of the crucial issues with 3D animation. We have chosen these particular formats because video streams can be naturally incorporated in the animated versions, providing substantial data reduction (somewhat analogous ideas were developed in [6]).

For DepthImage, animation is performed by replacing reference images by MovieTextures. Lossy video compression does not seriously affect appearance of the resulting 3D objects.

3. MPEG-4 NODE SPECIFICATION

The DIBR formats are described in detail in MPEG-4 AFX node specification [3]. The definition of DIBR nodes is shown in Fig. 4. DepthImage contains fields determining position and orientation of either SimpleTexture or PointTexture. Scene-dependent information is stored in special fields of the DIBR data structures, allowing the correct interaction of DIBR objects with the rest of the scene. Note that the DepthImage node defines a single DIBR texture. When multiple DepthImage nodes are related to each other, they are processed as a group, and thus, should be placed under the same Transform node. The diTexture field specifies the texture with depth

```
DepthImage {
    field  SFVec3f     position      0 0 10
    field  SFRotation  orientation   0 0 1 0
    field  SFVec2f     fieldOfView   0.785398 0.785398
    field  SFFloat     nearPlane     10
    field  SFFloat     farPlane      100
    field  SFBool      orthographic  TRUE
    field  SFNode      diTexture     NULL
}

SimpleTexture {
    field  SFNode      texture       NULL
    field  SFNode      depth         NULL
}

PointTexture {
    field   SFInt32    width         256
    field   SFInt32    height        256
    field   SFInt32    depthNbBits   7
    field   MFInt32    depth         []
    field   MFColor    color         []
}

OctreeImage {
    field   SFInt32    octreeresolution   256
    field   MFInt32    octree             ""
    field   MFNode     octreeimages       []
}
```

Fig. 4. The specification of the DIBR nodes.

(Simple Texture or PointTexture), which shall be mapped into the region defined in the DepthImage node.

The OctreeImage node defines an octree structure and the projected textures. The octreeresolution field specifies maximum number of octree leaves along a side of the enclosing cube. The octree field specifies a set of octree internal nodes. Each internal node is represented by a byte. 1 in *i*th bit of this byte means that the children nodes exist for the *i*th child of that internal node, while 0 means that it does not. The order of the octree internal nodes shall be the order of breadth first traversal of the octree. The octreeimages field specifies a set of DepthImage nodes with SimpleTexture for diTexture field.

4. RENDERING

Rendering methods for DIBR formats are not part of AFX, but it is necessary to briefly explain the ideas used to achieve simplicity, speed and quality of DIBR objects rendering. Our rendering methods are based on splats, small flat color patches used as 'rendering primitives'. Splats are positioned in space according to the depth information for all the diTextures describing the object. Splat sizes are computed so that their projections constitute the object view without rendering-induced 'holes'. Then the obtained set of splats is rendered with the aid of standard OpenGL functions. For the OctreeImage, colors of the octree nodes (voxels) are reconstructed from the

reference images by rather complicated but fast algorithm. Then voxel sizes are computed, and the same OpenGL-based procedure completes the rendering.

5. EXPERIMENTAL RESULTS

DIBR formats were implemented and tested on various 3D models, both artificially constructed and obtained by scanning the physical objects. Quality of rendering, simplicity and convenience of the formats made them attractive for usage in the MPEG-4 AFX.

In Table I, we present storage requirements, which is needed for several 3D static and animated objects in DIBR formats: 'Dragon' is an animated model from the 3DS-MAX demo package; 'Morton512' is a DIBR versions of the same polygonal model (number in the model name indicates dimension of the reference image). 'Palm' is a complex model of a tree, illustrating the great flexibility of DepthImage format; 21 SimpleTextures where used to provide high-quality representation.

In Fig. 5, the results of rendering are shown. The quality of rendering is quite good for various types of object shape. The rendering frame rate is given in the Table I. Tests were performed on INTEL Pentium III 500MHz machine with OpenGL accelerator.

6. CONCLUSION

Set of image-based formats for representation of 3D static and animated objects was developed. Effective rendering methods, providing visualization at interactive rates with good quality, were described. Great flexibility of the developed formats and their ability to easily adapt to complex object shapes was demonstrated on test examples. Additional useful properties, such as easy mip-mapping, are also provided. Several efficient compression methods were developed and tested. The formats were adopted into MPEG-4 AFX (Animation Framework eXtension) as an alternative to usual polygonal representations.

REFERENCES

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

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

[3] ISO/IEC JTC1/SC29/WG11 N4415: PDAM of ISO/IEC 14496-1 / AMD4, Pattaya, December 2001.

[4] M. Oliveira, G. Bishop, D. McAllister. "Relief textures mapping," *Proc. of SIGGRAPH'00*, pp. 359-368, July 2000.

[5] J. Shade, S. Gortler, L. He, R. Szeliski, "Layered depth images," *Proc. of SIGGRAPH'98*, pp. 231-242, July 1998.

[6] C. Bregler, "Video based animation techniques for human motion," *SIGGRAPH'00 Course 39 : Image-based modeling and rendering*, July 2000.

[7] A. Zhirkov. "Binary volumetric octree representation for image based rendering," *Proc. of GRAPHICON'01*, 2001.

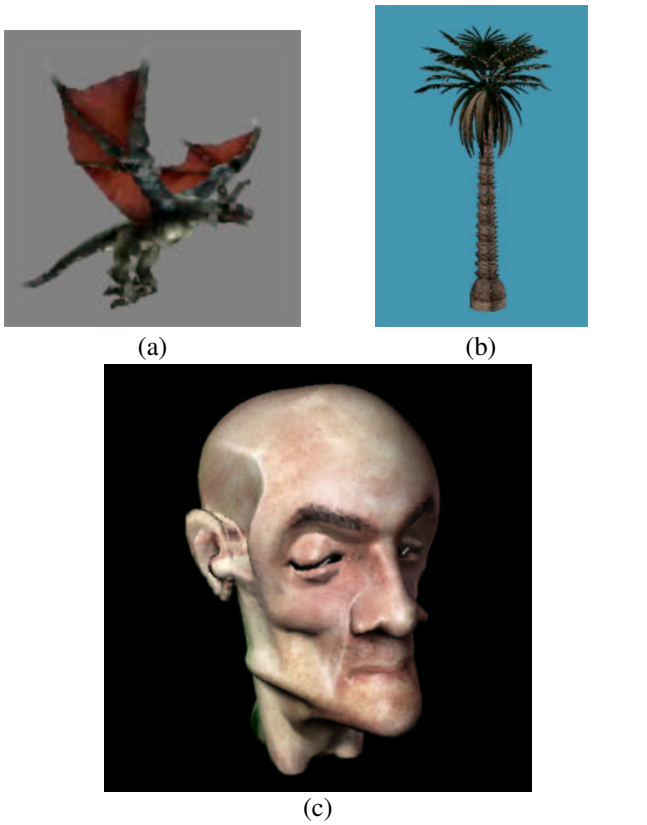
[8] M. Oliveira and G. Bishop, "Image-based objects," *Proc. of ACM Symposium on Interactive 3D Graphics*, pp. 191-198, April 1999.

[9] C. Chang, G. Bishop, and A. Lastra, "LDI Tree: A hierarchical representation for image-based rendering," *Proc. of SIGGRAPH'99*, pp. 291-298, August 1999.

[10] H. Pfister, M. Zwicker, J. Baar, and M. Gross, "Surfels: Surface elements as rendering primitives," *Proc. of SIGGRAPH'00*, pp. 335-342, July 2000.

Table I. Storage requirement and rendering time.

	Dragon	**Morton**	**Palm**
Format	OctreeImage	OctreeImage	DepthImage
Size	375 KB	173 KB	1.78 MB
Rendering Speed	24 fps	3.2 fps	1.1 fps
Number of Texture	6 (per frame)	6	21
Image Resolution	256 x 256	512 x 512	256 x 256



(a) (b)

(c)

Fig. 5. Screenshots of rendered objects. (a) Dragon. (b) Palm (c) Morton.