REAL-TIME MICROFACET BILLBOARDING FOR FREE-VIEWPOINT VIDEO RENDERING

Bastian Goldlücke and Marcus Magnor

Max-Planck-Institut für Informatik
Graphics - Optics - Vision
Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany
Email: bg@mpii.de

ABSTRACT

We present a hardware-accelerated method for video-based rendering relying on an approximate model of scene geometry. Our goal is to render high-quality views of the scene from arbitrary viewpoints in real-time, using as input synchronized video-streams from only a small number of calibrated cameras distributed around the scene. Despite the fact that only a very coarse geometry reconstruction is possible, our rendering approach based on textured billboards preserves the details present in the source images. By exploiting multi-texturing and blending facilities of modern graphics cards, we achieve real-time frame-rates on current off-the-shelf hardware. One of the possible applications of our algorithm is to use it in an inexpensive system to display 3D-videos, which the user can watch from an interactively chosen viewpoint, or ultimately even live 3D-television.

1. INTRODUCTION

With respect to the availability of affordable recording hardware, robust computer vision algorithms, and sophisticated rendering techniques, more and more research activity is devoted to investigate three-dimensional television [1, 2, 3]. The ultimate goal is free-viewpoint TV, where the user is able to watch a scene recorded by several cameras simultaneously from an arbitrary viewpoint chosen interactively.

Naturally, rendering techniques employed in 3D-TV are closely related to the field of image-based rendering (IBR). In IBR, the *light field* of a scene, i.e. all rays of light emerging from any point in space into any given direction, is sampled using several conventional photographs. Given enough samples from different viewpoints, one can reconstruct arbitrary views of the scene from outside the convex hull of the recording positions [4]. One major problem is that a very large number of samples is necessary to attain convincing rendering results [5]. A way to reduce this number is to employ computer vision algorithms to reconstruct 3D scene structure. Hybrid model/image-based rendering methods based on the visual hull [6], per-image depth maps [7] or even a complete 3D scene geometry model [8] achieve realistic rendering results from only a relatively small number of images.

Another challenge requiring considerable hardware effort is to record dynamic light fields and reconstruct 3D models at interactive frame rates. Multiple synchronized video cameras are needed to capture the scene from different viewpoints. An off-line approach based on volumetric reconstruction is presented in [9] where a dynamic voxel model is used to render the scene from novel viewpoints.

This paper focuses on the rendering part and presents a hardware-accelerated algorithm to efficiently render novel views of a person moving through a scene from arbitrary viewpoints. It is akin to the *Microfacet Billboarding* approach of Yamazaki et al [10], but in contrast to it, only the streamed data from several synchronized, calibrated cameras together with a voxel model of the visual hull is used as an input to the algorithm. This model can be computed interactively using a recording system built at our institute, which we briefly introduce in the next section. Sect. 3 is devoted to our novel rendering algorithm, whose results are presented in Sect. 4. We conclude with some plans for future work in Sect. 5.

2. VIDEO ACQUISITION AND ONLINE VISUAL HULL RECONSTRUCTION

We use the image-based visual hull as an approximate geometric model of the foreground objects in the scene. The system we have currently available acquires six synchronized video streams via pairs of cameras connected to three client PCs, Fig. 1. The clients separate the foreground from the known static background of the scene employing a color correlation scheme based on statistically obtained threshold values. These foreground silhouettes are then utilized to construct a discrete version of the partial image-based visual hulls on the client computer. It is stored as a cubic binary voxel volume, where a voxel value of 1 indicates that the corresponding scene point might be occupied. The par-

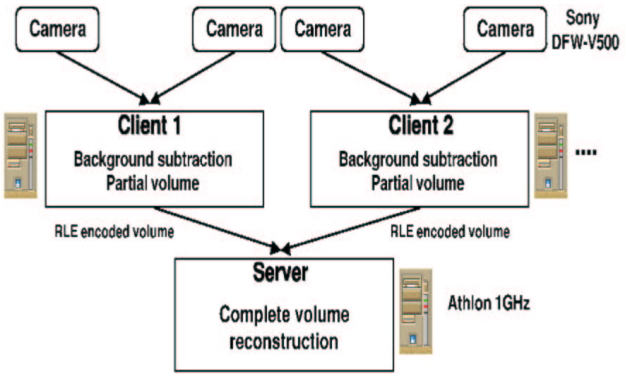

Fig. 1. *Architecture of the acquisition system*. The image-based visual hull is reconstructed online using several clients to compute partial volumes, which are combined on a server into the final voxel model of the scene.

tial voxel model is then RLE-encoded and transmitted to a server PC, which intersects the partial models to obtain the final voxel model of the visual hull.

The system was designed and built for online human motion capture [11]. It is capable of online performance and computes the visual hulls at a rate of 15 frames per second. The input to our rendering algorithm in each frame consists of the current camera images and the RLE-encoded voxel model of the visual hull. Currently we also precompute the visibility, i.e. whether a voxel is visible from a given camera or not.

3. HARDWARE ACCELERATED RENDERING

For each occupied voxel we render a single geometric primitive called a billboard, which is a rectangle parallel to the image plane having the same center as the voxel. Since the coordinates of the billboard's corners in 3D-space are known, we can compute their locations in the camera views and use this information to texture the billboards, Fig. 2. The cameras are immobile, so this projection Π can be precomputed and evaluated very efficiently using hardware-accelerated dependent texturing and customizable vertex transformations. In the remainder of the section we describe the texture setup in more detail.

For the sake of simplicity of notation we assume that the voxel volume lies in the unit cube $[0,1]^3$. The formulas can easily be adapted to arbitraty positions by inserting additional transformations at the appropriate locations. In a preprocessing step, the unit cube is divided into s^3 smaller cubes. The 3D textures T_i^{Π} which discretize the mappings Π_i will be of size $s \times s \times s$ and are defined in the centers of the cubes. For each camera i and each cube, we compute $\Pi_i(P)$ for its center P, encode the resulting homogenous 2D texture coordinate into the alpha and red channels of a texture and store them at the location P of the current

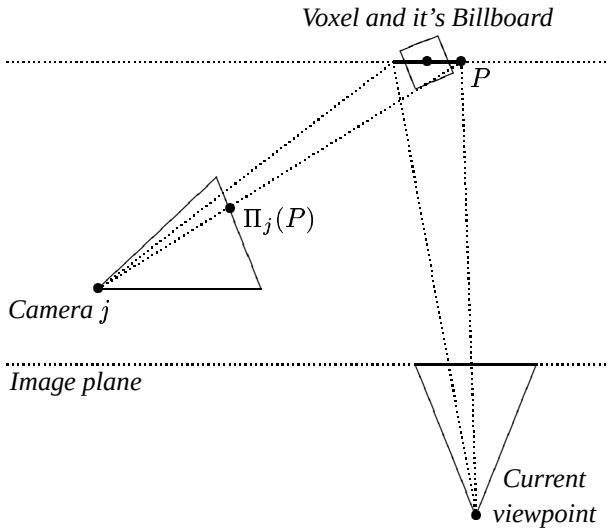

Fig. 2. *Illustration of billboard texturing*. The billboard associated to a voxel lies parallel to the image plane. It's corner P has the texture coordinate $\Pi_j(P)$ in the image of camera j. The projection Π_j can be precomputed.

3D texture T_i^{Π}. This initial setup needs to be performed only once.

For each frame in the sequence, the images I_i currently viewed by the cameras are retrieved from hard disk and loaded as texture images T_i^I onto the graphics card. Since we want to use for texturing only those pixels belonging to the silhouettes which were used to construct the voxel volume, we assign to all other pixels an alpha value of zero denoting full transparency, while an alpha value of one is assigned to all silhouette pixels. In order to smooth the blending in areas where the texture is in transition from one camera image to the next, an alpha value of 0.5 is assigned to pixels belonging to the boundary of the silhouettes.

During the last step the occupied voxel volume is rendered. This is the only part of the algorithm which depends on the current viewing direction. For each voxel of diameter d centered on V, we select two cameras j and k according to the following criteria:

- Any pixel of the voxel is visible in I_j and I_k.
- The arcs α_j and α_k between the viewing direction and the optical axes of the cameras j and k, respectively, are minimal.

The voxel will be textured with the images of these two cameras, which are blended depending on the similarity of the camera's optical axes to the current viewing direction. The blending weights ω_j and ω_k for the cameras are set to

$$\omega_{j,k} := 1 - \frac{\alpha_{j,k}}{\alpha_j + \alpha_k}.$$

That way, a camera's image is reproduced exactly if the

State	Value
General	
BlendEquation	FUNC_ADD
BlendFunc	SRC_ALPHA, ONE_MINUS_SRC_ALPHA
Texture unit 1	
SHADER_OPERATION	TEXTURE_3D
TEXTURE_ENV_MODE	NONE
Texture unit 2	
SHADER_OPERATION	DEPENDENT_AR_ _TEXTURE_2D_NV
TEXTURE_ENV_MODE	COMBINE
COMBINE_RGB	REPLACE
SOURCE0_RGB	TEXTURE
OPERAND0_RGB	SRC_COLOR
COMBINE_ALPHA	REPLACE
SOURCE0_ALPHA	PREVIOUS
OPERAND0_ALPHA	SRC_ALPHA
SOURCE1_ALPHA	TEXTURE
OPERAND1_ALPHA	SRC_ALPHA

Fig. 3. OpenGL and texture stage state during billboard rendering.

viewing direction coincides with its optical axis, and transitions are reasonably smooth when the selection of the two cameras changes due to a change in viewing direction – although not perfectly so, since everywhere smooth transitions require knowledgle about all the boundary lines between different camera selections, which is too much computational effort to acquire.

Rendering the voxel requires two passes and at least two hardware texture units. In the first pass, the texture T_j^{Π} is set up in texture unit 1 with dependent texturing enabled. Thus unit 1 computes texture coordinates for texture unit 2, to which we assign the camera image T_j^I. Blending is enabled to ensure that background pixels are transparent, and voxels have to be drawn in back-to-front order to correctly blend them one upon the other. The geometry transferred for the voxel is a single billboard centered on V parallel to the current image plane and of size $\sqrt{3}d \times \sqrt{3}d$ in order to cover the projection of the voxel with the projection of the billboard in the worst case. With texture coordinates set equal to the vertex positions, the graphics hardware now takes care of the rest.

Similarly, in the second pass, T_k^{Π} in unit 1 outputs texture coordinates for T_k^I in unit 2. We now have to blend correcly between the contribution of the first camera and this one, so the blending weight ω_k of the second image relative to the first one is stored in the alpha channel of the current color, which is modulated by the output of texture unit 2. The billboard is then rendered again. The correct setup for the texture shaders and environments in an example implementation using nVidia's OpenGL extension NV_texture_shader is summarized in Fig. 3. This setup can also be used for the first pass with the primary color's alpha value set to one, so no time-consuming state changes are required.

One also has to take great care to minimize the number of changes in texture images to optimize caching. In order to achieve this, a slight modification of the above scheme can be applied: The multiple textures for the source images are pasted above each other into one large single texture, where the v coordinate is translated depending on the current camera. The same is done with the textures for the mappings. That way texture images have to be selected just once before rendering all of the geometry.

4. RESULTS

In our method, besides the negligible amount of geometry, data transfer from memory to the graphics card is limited to six very efficient texture loads per frame, one for each camera image. This amounts to 1.8MB of raw data in the case of 320 x 240 RGBA source images. It might be reduced by first selecting the cameras which are able to contribute to the scene depending on the current viewpoint. However, it does not appear too much in view of the AGP 4x peak bandwidth, which lies in regions of 1GB per second. Instead, retrieving the data from hard drive fast enough is the bottleneck and requires a sophisticated compression scheme.

After all data has been transferred, our implementation is able to render a $64 \times 64 \times 64$ voxel volume from an arbitrary viewpoint at 30 fps. The contributions from the two nearest cameras are blended for each voxel. The program runs on a 1.8GHz Pentium IV Xeon with a GeForce 4 Ti 4600 graphics card. If no dependent texturing is available on a system, the mapping from 3D-space to the camera images can also be computed for each vertex by the CPU or a vertex program, decreasing performance slightly. The only real drawback of the algorithm is the limited resolution of the texture color channels: Since image coordinates are currently converted into 8-bit values, the discretized range of Π consists of only 256×256 distinct points. Higher image resolution gives more detail, nevertheless, because coordinates in-between arise from bilinear interpolation on the graphics card. Furthermore, future hardware available this year will definitely support textures with higher dynamic range, probably up to floating-point accuracy, thus nullifying the problem.

Fig. 5 displays a rendered image of the person from a novel viewpoint directly in between two of the source cameras having an angular distance of about 60 degrees. Note that despite the low resolution of $64 \times 64 \times 64$ of the voxel

Fig. 4. *Source images*. These are four of the six silhouettes used to construct the visual hull and texture the billboards. The cameras are spaced roughly 60 degrees apart.

Fig. 5. *Rendered view.* The novel viewpoint lies directly in between two of the source cameras in Fig. 4.

model, much finer details are visible. There are also few noticable transitions between voxels textured by different cameras, and the overall sharpness compared in view of sharpness and resolution of the source images in Fig. 4 is good.

5. CONCLUSIONS

We have presented a fast algorithm for image-based rendering using a voxel representation of the visual hull as an approximation to scene geometry. It achieves real-time frame rates on current off-the-shelf hardware and preserves intricate details in the source images while requiring only relatively coarse geometry data. Therefore it is an ideal technique for applications in free-viewpoint television, which is our ultimate goal.

To further advance into this direction, we plan to investigate novel compression methods tailored to the kind of data needed by the renderer. We also plan to accelerate the visibility computation to interactive rates and further improve the performance of the geometry reconstruction exploiting recent progress in the development of graphics hardware.

6. REFERENCES

[1] M. Op de Beeck and A. Redert, "Three dimensional video for the home," *Proc. EUROIMAGE International Conference on Augmented, Virtual Environments and Three-Dimensional Imaging (ICAV3D'01)*, Mykonos, Greece, pp. 188–191, May 2001.

[2] B. Wilburn, M. Smulski, K. Lee, and M. Horowitz, "The light field video camera," *SPIE Proc. Media Processors 2002*, vol. 4674, Jan. 2002.

[3] S. Würmlin, E. Lamboray, O. Staadt, and M. Gross, "3d video recorder," in *Proceedings of Pacific Graphics*, 2002, pp. 10–22.

[4] M. Levoy and P. Hanrahan, "Light field rendering," *Proc. ACM Conference on Computer Graphics (SIGGRAPH'96)*, New Orleans, USA, pp. 31–42, Aug. 1996.

[5] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum, "Plenoptic sampling," *Proc. ACM Conference on Computer Graphics (SIGGRAPH-2000)*, New Orleans, USA, pp. 307–318, July 2000.

[6] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan, "Image-based visual hulls," in *Proceedings of ACM SIGGRAPH*, 2000, pp. 369–374.

[7] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, "The lumigraph," *Proc. ACM Conference on Computer Graphics (SIGGRAPH'96)*, New Orleans, USA, pp. 43–54, Aug. 1996.

[8] D. Wood, D. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. Salesin, and W. Stuetzle, "Surface light fields for 3D photography," *Proc. ACM Conference on Computer Graphics (SIGGRAPH-2000)*, New Orleans, USA, pp. 287–296, July 2000.

[9] S. Vedula, S. Baker, and T. Kanade, "Spatio-temporal view interpolation," Tech. Rep. CMU-RI-TR-01-35, Carnegie Mellon University, Sept. 2001.

[10] S. Yamazaki, R. Sagawa, H. Kawasaki, K. Ikeuchi, and M. Sakauchi, "Microfacet billboarding," in *Proceedings of the 13th Eurographics Workshop on Rendering*, 2002, pp. 175–186.

[11] C. Theobalt, M. Magnor, P. Schueler, and H.-P. Seidel, "Combining 2D feature tracking and volume reconstruction for online video-based human motion capture," in *Proceedings of Pacific Graphics 2002*, 2002, pp. 96–103.