EUROGRAPHICS 2004 / M.-P. Cani and M. Slater
(Guest Editors)

Volume 23 (*2004*), *Number 3*

Hardware-Accelerated Rendering of Photo Hulls

Ming Li, Marcus Magnor and Hans-Peter Seidel

MPI Informatik, Saarbrücken, Germany

Abstract

This paper presents an efficient hardware-accelerated method for novel view synthesis from a set of images or videos. Our method is based on the photo hull *representation, which is the maximal photo-consistent shape. We avoid the explicit reconstruction of photo hulls by adopting a view-dependent plane-sweeping strategy. From the target viewpoint slicing planes are rendered with reference views projected onto them. Graphics hardware is exploited to verify the photo-consistency of each rasterized fragment. Visibilities with respect to reference views are properly modeled, and only photo-consistent fragments are kept and colored in the target view. We present experiments with real images and animation sequences. Thanks to the more accurate shape of the photo hull representation, our method generates more realistic rendering results than methods based on visual hulls. Currently, we achieve rendering frame rates of 2-3 fps. Compared to a pure software implementation, the performance of our hardware-accelerated method is approximately 7 times faster.*

Categories and Subject Descriptors (according to ACM CCS): CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism.

1. Introduction

Over the last decade, novel view synthesis from multiple real images has become an interdisciplinary research field between computer graphics and computer vision. Since real images are used as input, this technique lends itself to preserving fine detail and complex lighting effects existing in the original images, and hence is capable of generating photo-realistic virtual views.

Traditionally, the view synthesis problem is treated as a two-step process. First, explicit 3D models of real objects are reconstructed from the input images. Then a novel view is synthesized by projecting the 3D models onto the virtual image plane. Years of extensive investigations have led to a number of 3D reconstruction methods [DTM96, BBH03, Sze93, KS00]. The *shape-from-silhouette* approaches are based on the visual hull [Lau94], which is the maximal shape that reproduces the silhouette of an object from a given set of input images. Successful systems have been developed [MKKJ96, MBM01, TT02]. Unfortunately, the rendering quality of novel views is inherently limited since the visual hull representation is only an approximation to the true 3D geometry.

The *shape-from-photo-consistency* approach exploits color information in the input images to perform more accurate 3D reconstruction [KS00]. The result is often referred to as the *photo hull*. Although generally it provides better rendering results, its slow reconstruction performance prevents this approach from being applied in interactive or real-time view synthesis systems.

This paper addresses the performance issue and presents an efficient hardware-accelerated method to render photo hulls. Unlike the traditional way, we do not explicitly reconstruct complete photo hulls of 3D objects. Instead, reconstruction and rendering are combined into a single step. This is achieved by adopting a view-dependent plane-sweeping strategy to render a stack of planes directly in the target view. During rendering, the photo-consistency check is performed on the fly using graphics hardware, and photo hulls are implicitly reconstructed in form of depth maps. At the same time, colors collected from reference views are utilized to synthesize novel views of the photo hull. Visibility is also taken into account when performing photo-consistency checks and compositing colors from different reference views. Our method runs on off-the-shelf graphics hardware at interactive frame rates. Compared to a pure software implementation, the performance of our hardware-

accelerated method is approximately 7 times faster. To our knowledge, our method is the first to perform the entire reconstruction and rendering of photo hulls on graphics hardware with visibility fully taken into account.

The remainder of the paper is structured as follows. Sec. 2 reviews related work on visual hulls and photo hulls. Sec. 3 explains the hardware-accelerated photo hull rendering algorithm in detail. Sec. 4 presents our experiment results. Future research is discussed in the last section.

2. Related work

2.1. Visual hulls

The visual hull [Lau94] is the reconstructed result of the *shape-from-silhouette* approach. Usually, the reconstruction is accomplished by back-projecting the silhouettes of an object from different viewpoints into 3D space and intersecting the extruded generalized cones.

Research on the visual hull reconstruction dates back to Baumgart's PhD thesis [Bau74] in the 1970s. Since then, a variety of methods have been proposed. Most earlier methods are based on volumetric representations [Pot87, Sze93, MKKJ96]. Recently, new algorithms and increased computational power have led to the emergence of several real-time visual hull-based systems. View-independent polyhedral visual hulls reconstructed in real-time are demonstrated by Matusik et al. [MBM01]. Other researchers propose a number of methods to avoid explicit reconstruction and to render visual hulls directly from reference views. Lok [Lok01] renders a set of planes to generate novel views of visual hulls. Space occupancy is determined by exploiting graphics hardware. The image-based visual hull technique [MBR*00] creates view-dependent visual hulls by making use of epipolar geometry to speed up the intersection calculations of generalized cones. Li et al. [LMS03] rasterize generalized cones with projective texture mapping and achieve real-time rendering frame rates. Visual hull reconstruction is accomplished by modulating the projected alpha values and applying the alpha test.

The visual hull encloses the actual 3D object. The accuracy of the reconstruction increases as more input views are used. However, the inherent limitation of this method is that it fails to recover those concave regions that are not apparent in the silhouettes even with an infinite number of views, as is illustrated for 2D in Figure 1.

2.2. Photo hulls

The limitation of shape-from-silhouette methods can be overcome by exploiting additional information present in the reference images. The *shape-from-photo-consistency* approach takes advantage of color information in the input images to reconstruct the *photo hull* [SD97, KS00], which is

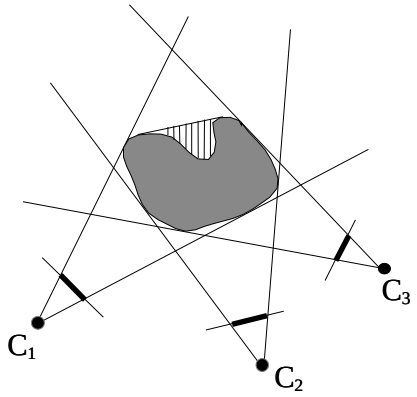

Figure 1: *Limitation of the visual hull approach. A visual hull created from three reference views in 2D. The three thick lines indicate silhouettes. The hatched concave area can never be reconstructed.*

the maximal shape that can consistently reproduce all reference color images. Compared to the visual hull, the photo hull is a more accurate geometric representation, and it leads to better photo-realism when synthesizing novel views.

The principle of the *shape-from-photo-consistency* approach consists of verifying the photo-consistency of each voxel in discretized 3D space. All inconsistent voxels are carved away and the remaining voxels form the photo hull. Usually, one assumes that the object to be reconstructed has approximately Lambertian surfaces. Under this assumption, a point on the surface of the object is photo-consistent if its corresponding projections in all visible reference views have similar intensities within a pre-defined threshold.

The original method for reconstructing photo hulls is presented by Seitz et al. [SD97] and refined by Kutulakos et al. [KS00]. Eisert et al. [ESG99] take a multi-hypothesis approach to verify the photo-consistency of voxels. Poulin et al. [PSD*03] reconstruct a point representation of photo hulls. However, all these algorithms are too time-consuming for real-time applications. Recently, hardware acceleration has been incorporated into the photo hull reconstruction algorithms. Prock et al. [PD98] and Sainz et al. [SBS02] make use of texture mapping capabilities of graphics boards to project input images onto voxels. This helps to obtain corresponding voxel color quickly. Culbertson et al. [CMS99] render voxels using graphics hardware to determine occlusion information with respect to input images. Although graphics hardware is employed in these methods, most computations still needs to be accomplished in software. Furthermore, these methods require reading back frame buffers to the main memory, which is a slow operation even for modern graphics boards.

Novel views of photo hulls can also be created without explicit reconstruction. Slabaugh et al. [SSH03] step along each viewing ray of the target view and find a sample that

	hardware-accelerated	photo-consistency check	considering visibility in photo-consistency check
IBVH	×	×	N/A
HAVH,OMR	√	×	N/A
CBSR	√	√	×
IBPH	×	√	√
HAPH	√	√	√

Table 1: *Comparison of our method with other relevant work.* ***IBVH****: Image-Based Visual Hulls [MBR*00]. **HAVH**: Hardware-Accelerated Visual Hull Reconstruction and Rendering [LMS03]. **OMR**: Online Model Reconstruction for Interactive Virtual Environments [Lok01]. **CBSR**: Real-Time Consensus-Based Scene Reconstruction [YWB02]. **IBPH**:Image-Based Photo Hulls [SSH03]. **HAPH**, Our method: Hardware-Accelerated Rendering of Photo Hulls*

is photo-consistent with reference views for each ray. Yang et al. [YWB02] render a stack of planes and exploit graphics hardware to perform the photo-consistency check using a simple criterion. Although fast performance is achieved, this method requires all reference views to be very close to each other because visibility is ignored during the consistency check. This imposes severe restrictions on the freedom of choosing novel viewpoints.

Table 1 summarizes the most relevant work and compares them with our proposed new method. For more comprehensive surveys on the reconstruction of visual hulls and photo hulls, please be referred to [Dye01, SCMS01].

3. Hardware-accelerated photo hull rendering

3.1. Overview

Our method renders the view-dependent photo hull from a set of images or videos taken from different viewpoints. First, we compute an effective bounding volume of the object under consideration. This volume is then discretized into a set of slicing planes parallel to the image plane of the target view. The planes are processed in a front-to-back order. While each plane is rendered, we determine an active set of reference images and project these images onto the plane. The photo-consistency of each rasterized fragment is checked by exploiting the programmability of graphics hardware. Visibility masks associated with the active reference views are maintained on the graphics board and play a critical role during the photo-consistency check. For display, the color of a photo-consistent fragment is evaluated as the weighted average of the corresponding pixel values in the reference images. Photo-inconsistent fragments are discarded. By knowing about the photo-consistency of each fragment of the current plane, we are able to update the visibility masks that are used to process the next slicing plane. The outline of our rendering algorithm is given in pseudo-code in Figure 2. The rest of this section will explain each step in detail.

```
Generate slicing planes
foreach slicing plane H
  Render H in the target view
    with hardware-accelerated photo-consistency check
  Update visibility masks for active reference views
end foreach
```

Figure 2: *Outline of our hardware-accelerated photo hull rendering algorithm.*

3.2. Slicing plane generation

We are interested in synthesizing a novel view of some particular object. If the approximate location of the object is unknown, a great number of planes has to be rasterized along the target viewing direction, each of them covering the whole output window. This kind of space discretization quickly exhausts the fill rate capability of graphics hardware. Therefore, it is highly desirable to generate slicing planes within a pre-determined bounding volume of the object.

By specifying a fixed bounding box, one can compute a set of planes by stepping through this box for the target view direction. However, this solution is not optimal since a fixed bounding box is typically too conservative. We provide a better solution by computing an *effective bounding volume* using the visual hull which bounds the actual object more tightly than the fixed bounding box. The effective volume is dynamically adjusted each time a view is synthesized. This volume remains conservative since the visual hull is a superset of the photo hull. This property guarantees that no geometry will be missed when rendering the photo hull.

To determine the effective bounding volume, a depth map of the visual hull is created for the novel view. From this depth map, we compute both the visual hull's bounding rectangle on the image plane and its visible depth range in the viewing direction. Then the effective bounding volume is constructed by using the bounding rectangle and the depth

range. Note that the depth range does not span the entire visual hull. It covers only the visible parts of the visual hull. Thus, we avoid discretizing space of occluded parts that contribute nothing to the final rendering result.

To create the depth map of the visual hull, we extract silhouette contours of the foreground object and employ a fast hardware-accelerated method [LMS03] to render the visual hull in a small off-screen window (e.g. 80 x 60 pixels). The depth buffer is read back to main memory in floating point format, and a rectangular region enclosing the visual hull is calculated. We expand this bounding rectangle by one pixel in order to tolerate rounding errors introduced by rendering a down-sampled visual hull. The expanded rectangle is then scaled by the size ratio between the actual target view and the small off-screen window. Meanwhile, the minimum and maximum depth values are determined from the depth map. Since we adopt the OpenGL graphics API, the depth values must be converted from window space to eye space using the following equation:

$$Z_e = \frac{Z_n \bullet Z_f}{Z_f - Z_w \bullet (Z_f - Z_n)},$$

where Z_w and Z_e are the coordinates in window space and eye space, respectively. Z_n and Z_f define the near and far clipping planes of the target viewing volume. Once we know the depth range in eye space, we discretize the continuous depth within this range. For each discretized depth value, its corresponding slicing plane can be directly derived given the bounding rectangle on the image plane and the target viewing parameters. This is illustrated in Figure 3. To further tighten the bounding volume, we divide the depth range into several subranges and compute a smaller bounding rectangle for each of them.

Both visual hull rendering and depth buffer reading do not take much time because the off-screen window is very small. In our tests, this step amounts to only 2% of total rendering time. This cost greatly pays off later in our rendering algorithm.

3.3. Slicing plane rendering

Traditionally, view synthesis approaches based on photo hulls treat reconstruction and rendering separately, and the photo-consistency check is performed on voxels in object space. In our method, the photo hull reconstruction is embedded into the rendering process. The slicing planes are sent through the graphics pipeline, transformed by the target viewing parameters, and rasterized into fragments. Instead of examining voxels, we check photo-consistency for each fragment by running a fragment program [Ope] on the graphics card and decide whether to draw the fragment in the output frame buffer.

To perform the photo-consistency check for a fragment, we need to locate those pixels in the reference views which

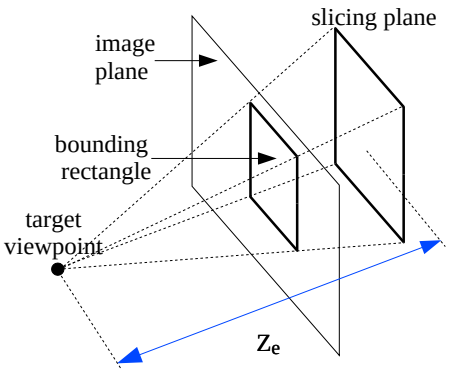

Figure 3: *Slicing plane generation. Given the bounding rectangle enclosing the visual hull, and a discrete depth value Z_e along the viewing direction, the slicing plane can be directly generated. This is done for each discrete depth value within the depth range of the visual hull.*

correspond to the visible 3D point. Those reference views behind the frontmost slicing plane can be simply excluded. The remaining views are called *active reference views*. We make use of projective texture mapping to sample the pixels in all active views. Texture matrices are set up to match the viewing parameters associated with reference images. To account for visibilities, for each active view a visibility map is kept on the graphics board to indicate which color pixels should participate in the photo-consistency check as well as the output color composition. Each visibility map coincides with its associated active view and shares the same set of texture coordinates. Figure 4 gives an example of the photo-consistency check as well as the involved visibility issue.

The measure that we adopt to check for the photo-consistency is the variance of corresponding visible pixel values in different reference views. Yang et al. [YWB02] approximate the variance using the *sum-of-squared-difference* (SSD). Unlike their method, we compute the true variance value, giving more reliable results. In the fragment program the variance σ^2 is computed as follows:

$$\sigma^2 = \left[\sum_{i=1}^{N}(R_i - \bar{R})^2 + \sum_{i=1}^{N}(G_i - \bar{G})^2 + \sum_{i=1}^{N}(B_i - \bar{B})^2\right] / N,$$

where N is the number of those active views in which the 3D point associated with the fragment is visible, (R_i, G_i, B_i) is the sampled pixel color from the ith view, and $(\bar{R}, \bar{G}, \bar{B})$ is the mean color of the corresponding pixels in all N views. Once we know the variance, the photo-consistency can be expressed as a threshold function:

$$\textit{photo-consistency} = \begin{cases} 1, & \sigma^2 < T \\ 0, & \text{otherwise} \end{cases},$$

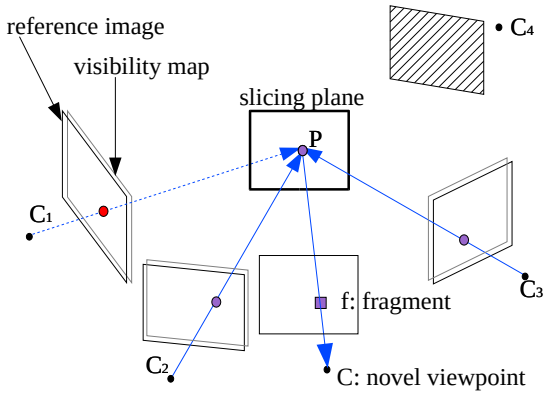

Figure 4: *Photo-consistency check of a fragment. C_1, C_2, C_3 and C_4 are reference viewpoints. The views corresponding to C_1, C_2, C_3 are active views. The dotted frame attached to each of them represents a visibility map. Each reference view is associated with a visibility map. From C_1, the point P is occluded according to its visibility map, and the red pixel in C_1 is not used for the photo-consistency check. The fragment f is photo-consistent because the corresponding pixels in C_2 and C_3 have the same color.*

where T is a user-defined threshold. When applying the photo-consistency check, we need to pay attention to the pixels close to the silhouette contour in the reference images. The reason is that such pixels often represent mixtures of the foreground and the background pixels. Therefore, they should not contribute to the decision on photo-consistency. In order to leave them out, we compute a sloping map for each reference view and encode it in the alpha channel. This map has the value 1 in the central part of the foreground object and drops gradually to 0 at its boundary. By setting a small threshold, we prevent the pixels near the boundary from being sampled.

Sloping maps are also employed to perform the silhouette-consistency check, which is useful for rejecting most inconsistent fragments efficiently. In a sloping map, the value 0 represents scene background, whereas non-zero values indicate the foreground object. A fragment is silhouette-consistent only if the alpha values sampled from all reference views are greater than zero. This check is performed before the photo-consistency check in the same fragment program. It amounts to the visual hull computation as described in [Lok01].

Finally, if a fragment passes both the silhouette-consistency and the photo-consistency check, a color is assigned to the fragment using the following formula:

$$(R,G,B) = \frac{\sum_{i=1}^{N} A_i * (R_i, G_i, B_i)}{\sum_{i=1}^{N} A_i},$$

where A_i is the alpha value of the ith sloping map. This weight eliminates the artifacts of having seams along silhouette boundaries on the photo hull. Parallel to processing the color channels, we set the alpha channel of the output frame buffer to 1 for the fragments passing the photo-consistency test. This information will be used for updating the visibility maps in the next step.

Figure 5: *Influence of visibility maps in photo hull rendering. Eight reference views are used. **Left**: Without considering visibilities with respect to reference views, the black color from the raised foot participates in the color-consistency check. Therefore, some regions on the thigh of the other leg fail to be reconstructed. **Right**: This artifacts are removed when visibility maps are employed.*

3.4. Visibility map updating

Initially, all visibility maps contain the value zero, which suggests that the first slicing plane is visible for all pixels in the active reference views. While we proceed to render each slicing plane from front to back, implicit geometric information of the object is produced in the form of alpha maps in the target view. This information must be reflected in the visibility maps in order to be able to check the photo-consistency and to composite final colors correctly. To show the importance of the visibility maps, we compare the rendering results with and without visibility maps in Figure 5. Note that we only need to update the visibility maps for the active reference views.

The algorithm for updating visibility maps is as follows:

1. Copy the alpha channel of the target view to a texture T_a. Since a bounding rectangle is determined in the subsection 3.2, the copying operation only needs to be carried out for the area of the bounding rectangle which is usually much smaller than the whole output window.
2. Render the current slicing plane for each active reference view using the texture T_a.
   The rendering is performed in an off-screen buffer on the graphics card and does not interfere with the frame buffer of the target view. The current slicing plane, together with the texture T_a, represents one layer of the photo hull in

object space. Therefore, by projecting this layer to a reference view, we are able to generate the occlusion information for the planes behind the current one. We divide off-screen buffer into rectangular regions and assign them to different reference views. For each active reference view, the viewport is set to its corresponding rectangular region. Then the slicing plane is rendered with the texture T_a. The visibility maps are produced in the alpha channel of the off-screen buffer and copied to separate textures in preparation for rendering the next plane.

3. Clear the alpha channel of the target view.
   The alpha channel of the target view corresponds to the current slicing plane. It should be erased before processing the next plane. We achieve this by clearing the output frame buffer with the color channels disabled for writing.

All these operations are executed on the graphics card. No copying operation from the frame buffer to main memory is required. Rendering the slicing planes textured by alpha maps is a trivial task for modern graphics hardware.

4. Experimental results

In order to achieve scalability to process multiple live videos, our photo hull rendering algorithm has been integrated into a distributed system. The size of the input images is 320x240 pixels. Several client computers connect to a rendering server via a standard TCP/IP network. Each node has an Athlon 1.1GHz CPU and is responsible for silhouette contour extraction and sloping map computation. The server is a P4 1.7GHz processor machine equipped with a GeForce FX 5800 Ultra graphics card. The resolution of the rendered novel view is set to 320x240 pixels. The bounding rectangle enclosing the photo hull occupies about one third of the entire viewport. The space is discretized into 60 slicing planes orthogonal to the target viewing direction.

We have carried out our experiments using three datasets. One consists of the real image data of a toy puppy. Eight reference cameras are placed in front of the puppy and span about 120° along an arc. In Figure 6, the comparison between the visual hull and the photo hull shows that the photo hull-based method is capable of recovering general concave regions and generating more realistic novel views.

The second dataset is a synthetic animation sequence of a girl performing Kungfu. Eight cameras are evenly distributed and along the circumference around the girl. Figure 7(a) illustrates the configuration and Figure 7(b) shows a novel view of the photo hull. We generate slicing planes using the fixed bounding volume and the effective bounding volume, respectively. The rendering speed of the latter is approximately 2.5 times higher.

As the third dataset, real video sequences are recorded from eight surrounding Firewire cameras to make a parody of the bullet time scene in the movie "The Matrix". A snapshot rendered from these videos is presented in Figure 7(c).

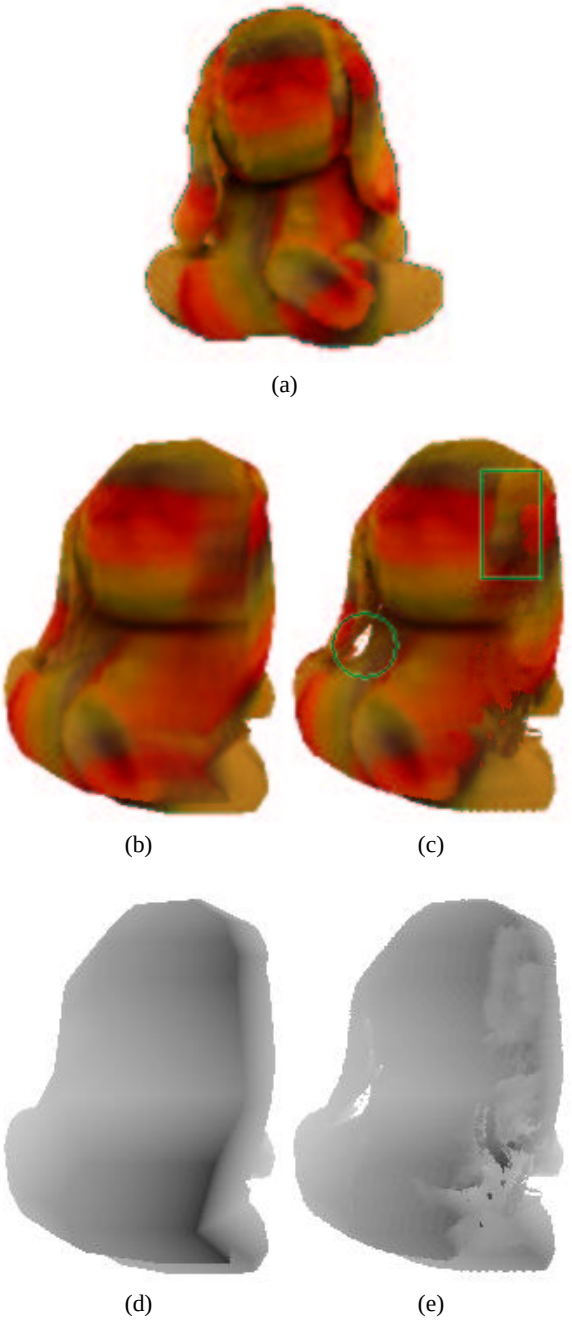

Figure 6: *Comparison of the rendering results of the visual hull and the photo hull.* ***(a)*** *One of the eight input images.* ***(b)*** *A novel view of the visual hull of the puppy.* ***(c)*** *A novel view of the photo hull generated with our hardware-accelerated method. In the circle, the empty space between the left ear and the body are correctly carved away. Also note that the boxed region is less blurry than the corresponding region in (b).* ***(d)*** *and* ***(e)*** *are the depth maps for (b) and (c), respectively. Note the fine details revealed in (e).*

An accompanying video demonstrates that the user can interactively change his/her viewpoint and examine the person in action.

For the "Kungfu girl" and "Matrix parody" datasets, all reference views are used during rendering. Three of them are the active views for most target views (cf. Subsection 3.3). The rendering speed is about 2 fps. For the "puppy" dataset, the reference views are closer to each other and the number of active views ranges from 5-8. In this case, we need more time to update the visibility maps. Therefore, the rendering speed decreases to 1.7 fps.

The *image-based photo hull* technique (IBPH) [SSH03] can be regarded as a software implementation of our method. In order to compare the performance with it, we apply our method to a subset of the puppy dataset and keep the rendering configuration as close as possible to theirs. Five reference views are used, and the target viewpoint is specified such that all views are active. While the IBPH runs at 0.4 fps on a machine with dual 2GHz processors, we achieve frame rates of 2.7 fps, approximately 7 times faster. Notice that the IBPH work also reports some higher frame rates. However, they are obtained by computing down-sampled photo hulls in the target view (For instance, 6.8 fps for a grid size of 4x4 pixels). For more fair comparison, we do not use them here.

The number of reference images supported by our method is restricted by the maximum number of texture units available on the graphics hardware. In our experiment setup, the Geforce FX graphics card has 8 texture units. However, in the very near future, the new graphics hardware will have more texture units. Another factor influencing the performance of our method is that current graphics hardware does not have full support of branch instructions. Therefore, in our fragment program, final color evaluation has to be carried out not only on the photo-consistent fragments but also on rejected fragments. This inefficiency will disappear with the next generation of graphics cards featuring real branch instruction support, and the performance of the proposed rendering algorithm will be notably improved. The potential speed-up depends on the ratio of the accepted and rejected fragments. We expect our algorithm can easily go up to 2 or 3 times faster for a typical scene.

5. Conclusion and future work

In this paper we have presented a hardware-accelerated photo hull rendering method for directly synthesizing novel views from a set of images or videos. By exploiting color information, our method is able to recover concave parts of an object. We generate novel views of the object without explicit photo hull reconstruction by rendering slicing planes from the target viewpoint. Graphics hardware is extensively employed to constrain discretization space, to check the photo-consistency of fragments and to maintain the visibility maps. Thanks to hardware acceleration, the rendering algorithm runs at interactive frame rates, and is much faster than a software implementation.

Currently, the photo-consistency check in our method is based on a single sample from each active reference view. Thus, calibration errors and image noise can introduce instabilities to the checking process. Incorporating local neighborhood information will provide even more robust reconstruction and rendering results. The mipmapping technique as utilized in [YP03] could be adopted in this context.

Like most photo hull-based methods to date, our method assumes Lambertian surfaces, so the photo-consistency check is sensitive to glossy surfaces. Removing this restriction would lead to more general solutions. We are planning to investigate suitable approaches.

References

[Bau74] BAUMGART B. G.: *Geometric modeling for computer vision*. PhD thesis, Stanford University, Oct. 1974. 2

[BBH03] BROWN M. Z., BURSCHKA D., HAGER G. D.: Advances in computational stereo. *PAMI 25*, 8 (Aug. 2003), 993–1008. 1

[CMS99] CULBERTSON W. B., MALZBENDER T., SLABAUGH G. G.: Generalized voxel coloring. In *Workshop on Vision Algorithms* (Sept. 1999), pp. 100–115. 2

[DTM96] DEBEVEC P. E., TAYLOR C. J., MALIK J.: Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH 1996* (Aug. 1996), pp. 11–20. 1

[Dye01] DYER C. R.: Volumetric scene reconstruction from multiple views. In *Foundations of Image Understanding*. Kluwer, 2001, pp. 469–489. 3

[ESG99] EISERT P., STEINBACH E., GIROD B.: Multi-hypothesis, volumetric reconstruction of 3-D objects from multiple calibrated camera views. In *International Conference on Acoustics Speech and Signal Processing, ICASSP '99* (Mar. 1999), pp. 3509–3512. 2

[KS00] KUTULAKOS K., SEITZ S.: A theory of shape by space carving. *IJCV 38*, 3 (July 2000), 199–218. 1, 2

[Lau94] LAURENTINI A.: The visual hull concept for silhouette-based image understanding. *IEEE Trans. PAMI 16*, 2 (Feb. 1994), 150–162. 1, 2

[LMS03] LI M., MAGNOR M., SEIDEL H.-P.: Hardware-accelerated visual hull reconstruction and rendering. In *Graphics Interface 2003* (June 2003), pp. 65–71. 2, 3, 4

(a) (b) (c)

Figure 7: *Photo hull rendering resulting from eight surrounding input views.* ***(a)*** *The arrangement of the cameras* ***(b)*** *A novel view from the Kungfu girl synthetic sequence.* ***(c)*** *Novel view from the matrix parody real video sequence.*

[Lok01] LOK B.: Online model reconstruction for interactive virtual environments. In *2001 Symposium on Interactive 3D Graphics* (Mar. 2001), pp. 69–72. 2, 3, 5

[MBM01] MATUSIK W., BUELER C., MCMILLAN L.: Polyhedral visual hulls for real-time rendering. In *12th Eurographics Workshop on Rendering* (June 2001), pp. 115–125. 1, 2

[MBR*00] MATUSIK W., BUEHLER C., RASKAR R., GORTLER S. J., MCMILLAN L.: Image-based visual hulls. In *SIGGRAPH 2000* (July 2000), pp. 369–374. 2, 3

[MKKJ96] MOEZZI S., KATKERE A., KURAMURA D. Y., JAIN R.: Reality modeling and visualization from multiple video sequences. *IEEE Computer Graphics and Applications 16*, 6 (Nov. 1996), 58–63. 1, 2

[Ope] OPENGL ARCHITECTURAL REVIEW BOARD: ARB_fragment_program OpenGL extension. http://oss.sgi.com/projects/ogl-sample/registry/ARB/fragment_program.txt. 4

[PD98] PROCK A. C., DYER C. R.: Towards real-time voxel coloring. In *1998 Image Understanding Workshop* (Nov. 1998), pp. 315–321. 2

[Pot87] POTMESIL M.: Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics and Image Processing 40* (1987), 1–20. 2

[PSD*03] POULIN P., STAMMINGER M., DURANLEAU F., FRASSON M.-C., DRETTAKIS G.: Interactive point-based modeling of complex objects from images. In *Graphics Interface 2003* (June 2003), pp. 11–20. 2

[SBS02] SAINZ M., BAGHERZADEH N., SUSIN A.: Hardware accelerated voxel carving. In *1st Ibero-American Symposium in Computer Graphics* (July 2002), pp. 289–297. 2

[SCMS01] SLABAUGH G. G., CULBERTSON W. B., MALZBENDER T., SCHAFER R.: A survey of volumetric scene reconstruction methods from photographs. In *Volume Graphics 2001* (June 2001), pp. 81–100. 3

[SD97] SEITZ S. M., DYER C. R.: Photorealistic scene reconstruction by voxel coloring. In *CVPR 1997* (June 1997), pp. 1067–1073. 2

[SSH03] SLABAUGH G. G., SCHAFER R. W., HANS M. C.: Image-based photo hulls for fast and photo-realistic new view synthesis. *Real-Time Imaging 9*, 5 (Oct. 2003), 347–360. 2, 3, 7

[Sze93] SZELISKI R.: Rapid octree construction from image sequences. *CVGIP: Image Understanding 58*, 1 (July 1993), 23–32. 1, 2

[TT02] TAKASHI M., TAKESHI T.: Generation, visualization, and editing of 3d video. In *1st International Symposium on 3D Data Processing Visualization and Transmission* (June 2002), pp. 234–245. 1

[YP03] YANG R., POLLEFEYS M.: Multi-resolution real-time stereo on commodity graphics hardware. In *CVPR 2003* (June 2003), pp. 211–220. 7

[YWB02] YANG R., WELCH G., BISHOP G.: Real-time consensus-based scene reconstruction using commodity graphics hardware. In *Pacific Graphics 2002* (Oct. 2002), pp. 225–235. 3, 4