Removing Highlight Spots in Visual Hull Rendering

Jie Feng, Liang Chen and Bingfeng Zhou
Institute of Computer Science & Technology, Peking University, Beijing, China

Abstract

Highlight spots could often cause artifacts in image-based visual hull (IBVH) rendering. In this paper, we propose a method that can remove highlight spots from reference images by utilizing the features of visual hull. First, we find the counterparts of a highlight sub-image in other images, with the aid of the correspondence between reference images in visual hull rendering. Then the illumination and color detail of the pixels in the sub-image could be retrieved from their corresponding pixels by color blending. Thus, highlight spots are removed seamlessly without affecting other regions of the scene. Using this improved rendering method, we can obtain more realistic and precise result, and re-lighting of the visual hull also becomes possible.

Introduction

Visual hull is an efficient technique for image-based rendering. Its main idea comes from 3D reconstruction methods of "Shape-from-Silhouette". It takes a group of photos as input, and produces a convex hull of the target object. Same as other image-based methods, visual hull also suffers from the highlight spots on the images, because they could often cause undesired artifacts in rendering, especially when the illumination in the virtual environment is changed.

In literatures, some image editing methods have been proposed to remove highlight spots from photos. Some of them need more than one photo at a single position to complete this task. For instance, the method of Agrawal et al [1] requires a pair of flash and ambient images, taken at the same position, and reduces highlights by comparing their image gradients. This kind of method is not practical in visual hull rendering, because it will largely increase the difficulty of acquiring and storing source images.

Some other methods could work on a single image, such as [2] and [3]. The former introduces illumination-based constrains into image inpainting, and the latter changes local illumination by solving Poisson equations. However, they only report good results when highlight spots lie in areas with simple or uniformly textured background. On another aspect, single-image-based methods cannot guarantee the consistency of corresponding areas in different source images, which is very important in visual hull rendering.

In fact, visual hull method itself provides much convenience for highlight removal. The source images, also called reference images, of a visual hull often have much overlaps. These overlaps will offer sufficient information to remove highlights. That is because the counterpart of a highlighted area in another image is often out of highlight, due to the relative movement of the object and the light.

Utilizing the calibration information of the cameras, the correspondence of pixels in different images could be found during the constructing of the visual hull. Thus, highlighted areas could be resampled from other reference images, and a new reference image without highlight spots could be generated by certain pixel blending strategy. With these new images, we could obtain more realistic and precise rendering result, and re-lighting of the visual hull is also possible.

In this paper, we employ an Image-based Visual Hull (IBVH) rendering method [4], which is described in the second section. The approach to interactively removing highlight spots in IBVH rendering is detailed in the third section, and the experiment results follow in the last.

Visual Hull Rendering Methods

The research on visual hull reconstruction dates back to 1970s [5]. Its main idea comes from "Shape-from-Silhouette". A reference image is separated into foreground and background. The foreground mask, i.e. silhouette, along with the calibration information of the camera, defines a back-projected cone in 3D space that contains the target object. Thus, the intersection of all silhouette cones forms a convex hull, which is the visual hull of the object.

There are mainly two sorts of visual hull construction methods: voxel-based and boundary-based methods. [6]

Voxel-based methods [7, 8] usually start from a working volume, which contains the target object and is quantized into voxels. The voxels are one by one put through tests: those lie inside all silhouette cones are preserved; the others are cleared. Therefore, the remaining voxels form the visual hull of the object. These methods are able to reconstruct very complex objects, such as trees. However, they usually suffer from quantization artifacts and are very expensive in both space and time costs.

In boundary-based methods [4, 9], silhouette cones are represented as boundary elements, such as surfaces or lines. The visual hull is constructed by computing the intersection of these elements, and the result could be composed by a group of surfaces patches, line segments, or points. Such methods usually consume little memory and run faster than voxel-based methods, and quantization artifacts are avoided. If necessary, boundary-represented visual hull could also be triangulated into meshes to produce an explicit 3D model. That makes it more useful in applications. However, these methods cannot reconstruct concave or very complex objects.

Image-based Visual Hull Rendering Method

In this paper, we adopt a method called Image-based Visual Hull (IBVH). It is a boundary-based method, originally proposed by Matusik et al [4].

In this algorithm, silhouettes are represented as a bench of 3D rays emitted from a desired view, and the resulting visual hull is made up of a group of line segments. It is remarkable that the computing is limited to the image space of the reference images,

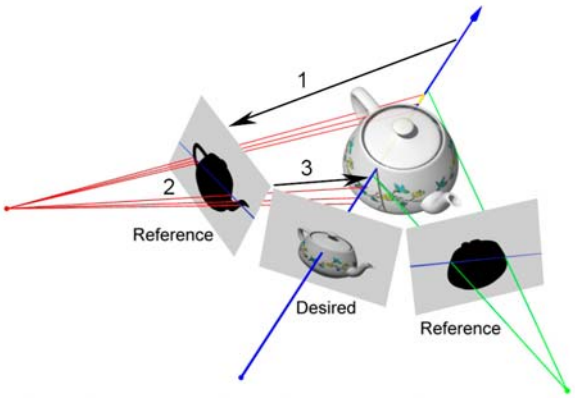

***Figure 1**. The three steps of IBVH computing [4]: 1) A desired viewing ray is projected to a reference image. 2) The epipolar line intersects the silhouette. 3) The resulting 2D intervals are projected back to 3D and get corresponding 3D segments.*

and the result is view-dependent, which makes the method quite efficient.

As illustrated in Fig. 1, the kernel algorithm of IBVH method includes three main steps:

First, for every pixel of the desired view, a viewing ray is calculated, which is emitted from the virtual camera center and passes through current pixel. By using the calibration information of cameras and epipolar geometry theory, the projection of this viewing ray on a reference image, i.e. the epipolar line could be computed.

Second, the epipolar line intersects the 2D silhouette on the reference image, and results in a group of 2D intervals. To reduce the computation cost and increase speed, the original algorithm sorts silhouette edges in so-called *bin* structures by their slopes. In our implementation, we adopt a different strategy in building *bins*: instead of slopes, the edges are sorted by their direction angles, so that all edges could be properly sorted, even when the epipole (the intersection point of all epipolar lines) falls in the object area.

Third, the 2D intervals on the reference image are projected back to 3D space and get corresponding 3D segments on the viewing ray. Then the intersection of all the 3D segments from all the reference images indicates the visual hull boundary at current pixel of desired view. Projecting the nearest endpoint of final 3D segments onto the reference images, the color of current pixel could be synthesized.

Removing Highlight Spots on Visual Hull

The IBVH rendering method provides great convenience in finding pixel correspondence between reference images. Utilizing this correspondence, we could rapidly remove highlight spots by resampling the target image, using the information from its counterparts on other reference images.

To reduce computing cost and minimize the error introduced by pixel mapping and resampling, we first select sub-images that contain highlight spots interactively. The resampling performance would be applied only to these selected highlight sub-images, and the rest part of the image would remain unchanged.

Pixel Correspondence between Reference Images

The approach to finding pixel correspondence between different reference images is similar to that of rendering a desired new view in IBVH, as described in the last section. Their fundamental differences lie in: 1) When rendering a desired view, the target image is a new, virtual and synthesized one, but here it is exactly one of the reference images; 2) The calculations are no longer performed over the whole image, but limited to the selected highlight sub-images.

Fig. 2 gives an illustration of how pixel correspondences are found. Given a target image I_0, for every pixel p_0 in a highlight sub-image, we could also compute a 3D viewing ray r. For another reference image I_k, the epipolar line of p_0 (denoted as l_e) is calculated by using the fundamental matrix between I_0 and I_k. If line l_e intersects the silhouette of I_k at point p_a^k and p_b^k, then there are two 3D rays, r_a^k and r_b^k that are emitted from camera C_k and pass through p_a^k and p_b^k, respectively. Note that r, r_a^k and r_b^k are all in the same plane. Therefore, they would intersect and result in a 3D segment (v_a^k, v_b^k).

Applying the same calculation to other reference images, we get a group of 3D segments $\{(v_a^k, v_b^k) \mid k=1\ldots n\}$ (n is the number of images in use). The intersection of all these segments, denoted as (v_a, v_b), is considered as the intersection of r and the visual hull of the object, and the nearer endpoint v_a is the corresponding 3D point of pixel p_0. Utilizing the calibration information of camera C_k, v_a can be projected onto image I_k, and get pixel p_k. So far, we have found the corresponding pixel of p_0 on every reference images.

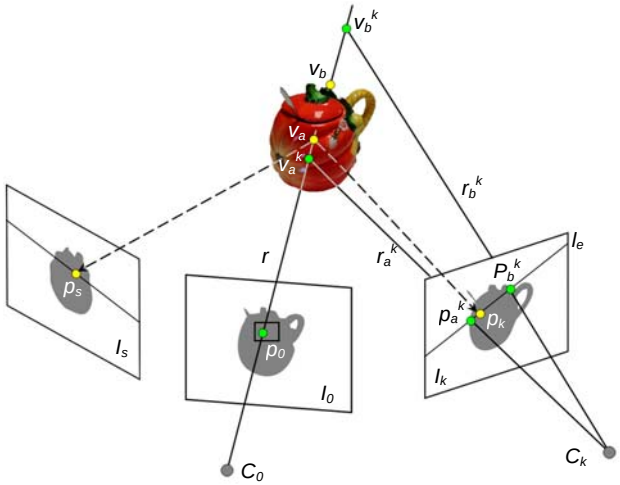

***Figure 2**. The procedure of looking for the corresponding pixels of p_0 on other reference images.*

Highlight Sub-image Resampling

When the counterparts of a highlight sub-image on other reference images are found, we must re-calculate the color of its pixels to reduce highlight effect. This is completed by blending the appearance colors of pixel p_0 and its corresponding pixels $\{p_k \mid k=1\ldots n\}$.

The fact that we could take advantage of is: usually, due to the relative movement of the object and the light, most part of the

***Figure 3**. The counterparts of a highlight sub-image. Left: The target image (the yellow rectangle is the highlight sub-image); Right: The counterparts of the sub-image on two other reference images (Note that they include little highlight in them, because the object rotates while the light does not).*

corresponding region of current sub-image is out of highlight area (Fig. 3). Based on this fact, we assume that the majority of the appearance colors of $\{p_k \mid k=0\ldots n\}$ are mainly distributed around the true diffuse color. Thus, when blending the colors, the pixel whose color deviates far from the others would be given small weight. On the contrary, those that are closer to the average color would be given larger weight.

In fact, highlight could be represented and detected well enough in gray-scaled images, so we need only calculate the gray level deviation of the pixels. That makes the calculation simpler. Let g_k be the gray level of pixel p_k, and the average gray level of $\{p_k \mid k=0\ldots n\}$ is

$$g_{avg} = \frac{1}{n}\sum_{k=0}^{n} g_k \,. \tag{1}$$

Then the weight of p_k could be defined by the reciprocal of the squared deviation of g_k:

$$w_k = c_k \frac{1}{|\, g_k - g_{avg} \,|^2}. \tag{2}$$

The coefficient c_k is a constant, which guarantees the sum of w_k to be 1. Finally, the colors of $\{p_k \mid k=0\ldots n\}$ are blended according to the following formula:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} \sum_{k=0}^{n} w_k R_k \\ \sum_{k=0}^{n} w_k G_k \\ \sum_{k=0}^{n} w_k B_k \end{pmatrix}. \tag{3}$$

Here, (R_k, G_k, B_k) is the RGB color of p_k, and (R, G, B) is the final color for pixel p_0.

Note that we assume n reference images (besides the target image) are used in resampling and blending a pixel. These n images are selected from all reference images according to the angles between the viewing ray of current pixel and the rays of its corresponding pixels. Only the images with the smallest angles, i.e. the closest n images are used to fix the target pixel.

Using a larger number of images will make the average color closer to the true color and help to filter out highlighted pixels. However, it does not always lead to better result, for blending more pixel colors would cause blurring effect, especially in the areas with complex texture. To get a globally better result, we adjust the value of n according to each sub-image's gray level deviation sum.

***Figure 4**. The result of removing highlight spots from reference images. The left are original images (yellow rectangles are highlight sub-images); the right are the resampled image, in which highlight spots have been removed seamlessly.*

Assume the current sub-image is composed of m pixels, and each pixel has a gray level of b_i. Then the average gray level is

$$b_{avg} = \frac{1}{m}\sum_{i=1}^{m} b_i \,, \tag{4}$$

and the deviation sum is

$$b_{dev} = \sum_{i=1}^{m} |\, b_i - b_{avg} \,|^2 \,. \tag{5}$$

Then a given threshold τ determines the value of n:

$$n = \begin{cases} N_{\max}, & b_{dev} > \tau \\ N_{\min}, & b_{dev} \le \tau \end{cases} \tag{6}$$

That means a sub-image with more complex texture will be given a smaller n to prevent blurring, and a sub-image with simpler texture will be given a larger n for smoother result.

Experimental Results

Applying the method described above, we can remove highlight spots from a group of cup photos and get satisfying rendering results.

The original data includes 23 images, taken around the object by a digital camera. There are obvious highlight spots in each image. They are selected as in the left column of Fig. 4 (each rectangle is a highlight sub-image). Using the correspondence in the visual hull, the counterparts of a sub-image can be found in other images (as shown in Fig. 3). Then, each pixel in the sub-image is blended with its corresponding pixels, and result in an image without highlight spots (right column of Fig. 4).

When deciding the number of images in use, we let $N_{\max}$=9 and $N_{\min}$=4, i.e. using 9 closest images in resampling a simple-textured sub-image, and 4 in complex sub-images.

Figure 5*. The results of removing highlight spots from several reference images, and the corresponding visual hull rendering result. Top row: reference images (left) with highlight spots and their rendering result (right); Bottom row: resampled reference images (left) and the new rendering result without highlights (right).*

More highlight removal results are shown in the left part of Fig. 5. With these resampled reference images, we can run visual hull method again to render the object at new viewpoints. The rendering results are free from highlights and are more realistic and precise in color (right of Fig. 5).

Furthermore, highlight-free visual hulls would make it possible to change the illumination in the virtual environment, and re-light the object. All we need to do is re-calculating the diffuse and reflection color of the object according to the virtual lights.

References

[1] A. Agrawal, R. Raskar, S.K. Nayar and Y. Li, Removing Photography Artifacts using Gradient Projection and Flash-Exposure Sampling, Proc. ACM SIGGRAPH 2005, pg. 828. (2005).

[2] P. Tan, S. Lin, L. Quan and H. Shum, Highlight Removal by Illumination-Constrained Inpainting, Proc. IEEE International Conference on Computer Vision 2003, pg. 164. (2003).

[3] P. Pérez, M. Gangnet and A. Blake, Poisson Image Editing, Proc. ACM SIGGRAPH 2003, pg. 313. (2003).

[4] W. Matusik, C. Buehler and R. Raskar, Image-Based Visual Hulls, Proc. ACM SIGGRAPH 2000, pg. 369. (2000).

[5] B.G. Baumgart, Geometric Modeling for Computer Vision, Ph.D. thesis, Stanford University, October 1974.

[6] M. Li, Towards Real-Time Novel View Synthesis Using Visual Hulls, Ph.D. thesis, Max-Planck-Institut für Informatik, September 2004.

[7] K.N. Kutulakos and S.M. Seitz, A Theory of Shape by Space Carving, International Journal of Computer Vision, Marr Prize Special Issue, 38, 3, pg. 199. (2000).

[8] G. Slabaugh, B. Culbertson, T. Malzbender and R. Schafer, A Survey of Methods for Volumetric Scene Reconstruction from Photographs, Proc. International Workshop on Volume Graphics 2001, pg. 81. (2001).

[9] K.M. Cheung, S. Baker and T. Kanade, Visual Hull Alignment and Refinement Across Time: A 3D Reconstruction Algorithm Combining Shape-From-Silhouette with Stereo, Proc. IEEE Conference on Computer Vision and Pattern Recognition 2003, pg. II-375. (2003).

Author Biography

Jie Feng received her Bachelor degree from School of Mathematics Science, Peking University, in 2000, and Ph. D. degree of Engineering from Center for Information Science, Peking University, in 2005. Her research interests include 3D modeling, digital geometry processing, image-based rendering etc.

Liang Chen received his Bachelor degree (with honor), from School of Electronic Engineering & Computer Science, Peking University in 2004. He is now a graduate student in the same department. His research interests include computational photography, non-photorealistic rendering, computer animation etc.

Bingfeng Zhou received his PHD degree in Peking University. He is now a researcher and doctoral supervisor in the Institute of Computer Science and Technology, Peking University. His research interests including digital geometry processing, image based rendering, none photorealistic rendering, solid modeling, GPU technology and digital video processing.