Automatic Recognizing and Removing of Highlight Spots in Visual Hull Rendering

Jie Feng, Liang Chen and Bingfeng Zhou
Institute of Computer Science & Technology, Peking University
{fengjie, chenliang, zhoubingfeng}@icst.pku.edu.cn

Abstract

In image-based visual hull (IBVH) rendering, undesired artifacts, such as highlight spots in reference images could often cause the decrease of rendering fidelity. In this paper, we propose a method that can automatically recognize and remove highlight spots from reference images, by utilizing the features of IBVH.

First, sub-images containing highlight spots are extracted in a target reference image by analyzing its intensity histogram. Then, their counterparts in other images are calculated, according to the correspondence between reference images in visual hull rendering. At last, the illumination and color detail of the pixels in the sub-images could be retrieved from their corresponding pixels by color blending. Thus, highlight spots can be removed seamlessly without affecting other regions of the image.

Using this method, we can obtain more realistic and precise result, and re-lighting of the visual hull also becomes possible.

1. Introduction

Visual hull is an efficient technique for image-based rendering of real objects. It takes a group of images of the target object as input, and produces an approximate convex hull that contains the object. Compared with traditional geometry-based rendering methods, visual hull rendering usually has lower cost in data acquiring, and could produce more realistic result.

However, as an image-based method, visual hull methods often suffers from the highlight spots on the images, which would cause undesired artifacts in rendering, especially when the illumination in the virtual environment is changed.

Many image editing techniques have been proposed to remove highlight spots from images.

A typical type of them requires more than one image at a single position to complete this task. For instance, a flash/no flash image pair is taken at the same position in the method of Petschnigg et al. [1]. Combining the ambient qualities of the no-flash image and the high-frequency detail of the flash image, a new image with higher quality then either input could be synthesized. This method simply removes the highlight areas by selecting them interactively and then replaces them with the information from the ambient image.

Agrawal et al also introduce a pair of flash and ambient images in their work [2]. They compare the image gradients of both two images, and remove the highlight spots by combining the gradients with a linear weight. Furthermore, the reflection in the ambient image could also be reduced by a gradient projection.

This kind of method is not practical in visual hull rendering, because it will largely increase the difficulty of acquiring and storing source images, and the complexity of the computation.

Another kind of methods could work on a single image. Tan et al. proposed an illumination-constrained inpainting method [3]. It utilizes potential useful information contained by highlight pixels to estimate the underlying diffuse color, and guide the process of image inpainting. Similar works also include [4, 5].

The method of Pérez et al. [6] takes a different strategy. It changes local illumination by applying a non-linear transformation to the gradient field in the selected highlight area. The modified gradient is integrated back by solving a Poisson equation, and result in a new image without specular reflections.

However, these methods only produce good results when highlight spots lie in areas with simple or regular 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 usually have a lot of overlaps. Such redundancy 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.

In our method, the sub-images containing highlight spots could be automatically extracted from a reference image, by analyzing its intensity histogram. Then their counterparts are found in other reference images, utilizing the calibration information of the cameras, during the constructing of the visual hull. Thus, the highlight sub-images could be resampled from their counterparts, and a new image without highlights is 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. Furthermore, this method could also be used in image repairing.

* The work was supported by the NSFC grants (No. 60573149).

In this paper, we employ an Image-based Visual Hull (IBVH) rendering method [7], which is described in the second section. In the third section, we introduce our method of automatic highlight sub-image extraction. The approach to resampling sub-images and removing highlight spots in IBVH rendering is detailed in the fourth section. The experiment results and some discussions are followed in the last.

2. Visual Hull Rendering Methods

The main idea of visual hull reconstruction comes from “Shape-from-Silhouette”. Each reference image is separated into foreground and background. The foreground mask, i.e. the 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 [8].

Voxel-based methods [9, 10] 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 cannot get smooth modeling results. To solve this problem, the working volume has to be divided into larger quantity of voxels, but this will greatly increase the cost in both storing space and calculating time.

In boundary-based methods [7, 11], 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 consume less 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 structure or very complex objects.

2.1 Image-based Visual Hull Rendering Method

In this paper, we follow the main idea of a method called *Image-based Visual Hull (IBVH)*. It is a boundary-based method, originally proposed by Matusik et al [7].

In this method, the input is a group of images taken around the object with calibrated cameras. The silhouette cones of these reference images are represented as a bench of 3D rays emitted from the camera centers, each corresponds to a pixel on the silhouette. Then the intersection of the cones, i.e. the visual hull, is composed of a group of line segments.

It is remarkable that in this method the computing is limited to the image space of the reference images, and the result is view-dependent, which makes the method quite efficient.

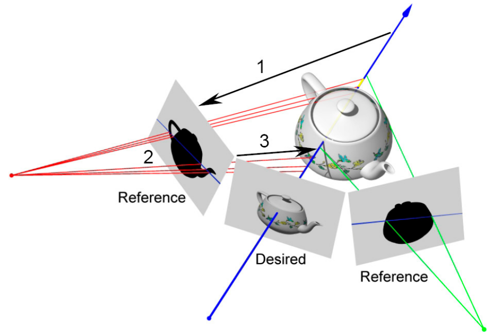


Figure 1. *The three steps of IBVH construction [7]. 1) A desired viewing ray is projected to a reference image. 2) The epipolar line intersects the silhouette. 3) The resulting 2D intervals are projected back to 3D and get corresponding 3D segments.*

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

1. For each pixel of the desired view, a ray emitted from the camera center and passing through current pixel is calculated. Using the calibration information and epipolar geometry theory, the projection of this ray on a reference image, i.e. the epipolar line could be computed.
2. The epipolar line intersects the 2D silhouette on the reference image, and results in a group of 2D intervals.
3. The 2D intervals are projected back to 3D space and get corresponding 3D segments on the viewing ray. Then the intersection of all the segments from all the reference images indicates the visual hull boundary at current pixel. Projecting the nearest endpoint of final 3D segments onto the reference images, the color of current pixel could be synthesized.

To reduce the computation cost and increase speed, the original algorithm sorts silhouette edges in so-called *bin* structures by their slopes during step 2. Here in our implementation, we make an improvement in the strategy of building *bins*. Instead of slopes, the silhouette edges are sorted by their direction angles, so that the intersection could be properly performed even when the epipole (the intersection point of all epipolar lines) falls in the object area.

3. Automatic Highlight Sub-image Extraction

In order to remove highlight spots in a target image, the sub-images that contain these highlights must be extracted first. The following calculations will be restricted to these sub-images. That is because the highlight spots usually occupy only a small part of the image. The calculation over the whole image would lower the computing efficiency and image fidelity cannot be guaranteed.

Khan et al. proposed an automatic highlight recognizing method in their work [12]. They assume that the highlights are made up of the brightest pixels of the image. Then, in the intensity histogram, the level corresponding to

(a) (b) (c)

***Figure 2**. Automatic Highlight recognition. (a) Original; (b) Recognized highlight pixels; (c) Extracted highlight sub-images.*

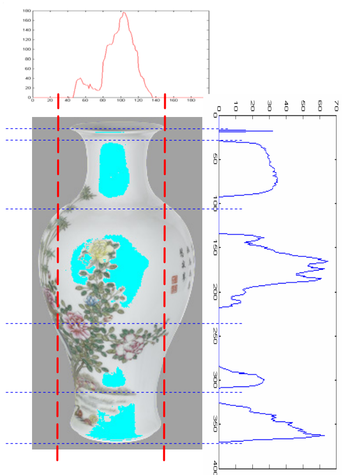


***Figure 3**. Highlight sub-image extraction.*

the minimum derivative is considered as the beginning threshold of the highlight pixels.

However, this method was designed for HDR images, and doesn't work so well in common images. As an adaptation, we set a constant lower bound of highlight pixel intensity I_{min} according to experiences. Then, the highlight threshold could be quickly found in interval [I_{min}, I_{max}] by binary search, where I_{max} is the maximum level of the pixel intensity. In our experiments, we usually control the highlight pixels to be 0.5%~4% of the whole image. An example is given in Fig. 2(b).

Because the recognized highlight pixels are discrete and usually scattered, they are firstly grown into connected regions. This is completed with an eight-neighbored seed filling method: given a highlight seed pixel in RGB space, if one of its eight neighbors has identical color vector to it, then we also consider this neighbor as a highlight pixel, and use it as a new seed for further computation. The growing will stop when there is no more seed pixel exists. Extending highlight pixels into connected regions will prevent the appearance of fragmentary sub-images, and therefore will decrease the computing cost.

After that, the extended highlight pixels are classified into several categories, with a C-average classification method. The initial classification is found by analyzing the image's vertical and horizontal histograms. As illustrated in Fig. 3, if there are C_1 wave crests in horizontal histogram and C_2 in vertical histogram, then they will define $C_1 \times C_2$ regions. Among them, those containing highlight pixels are considered as initial categories. Calculating the centroid of each category and re-allocating all the highlight pixels by their Euclidean distance, we would have the final classification. During this course, the category with too few highlight pixels would be ignored.

Finally, the minimum sub-images containing each category could be easily extracted by finding their bounding boxes (Fig. 2(c)).

4. Removing Highlight Spots on Visual Hull

Taking advantages of the features of IBVH, the pixel correspondence between reference images could be easily found. Utilizing this correspondence, we could rapidly remove highlight spots by resampling the target highlight sub-image, using the information from its counterparts on other reference images.

4.1 Finding Pixel Correspondence

The IBVH rendering method provides great convenience in finding pixel correspondence between reference images. The approach to finding such correspondence is similar to that of rendering a desired new view. 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 now.

Fig. 4 gives an illustration of how pixel correspondences are found. Given a target image I_0, for each pixel p_0 in one of the highlight sub-images, there is also a 3D ray r emitting from camera canter C_0. Its projection on another reference image I_k, i.e. the epipolar line l_e, could be calculated by using the fundamental matrix between I_0 and I_k.

Then l_e intersects the silhouette of I_k at p_a^k and p_b^k. Therefore, two rays r_a^k and r_b^k, emitting from camera C_k and pass through p_a^k and p_b^k, are calculated respectively. Note that r, r_a^k and r_b^k are in the same plane; therefore they would intersect and result in a 3D segment (v_a^k, v_b^k).

Repeating the calculation on all of the reference images, we get a group of such 3D segments $\{(v_a^k, v_b^k) \mid k=1 \ldots n\}$ (where n is the number of images in use). Their final intersection (v_a, v_b) is considered as the intersection of r and the visual hull, and the nearer endpoint v_a is the corresponding 3D point of p_0.

Thus, utilizing the calibration information of camera C_k, v_a can be projected onto image I_k, and get p_k, the corresponding pixel of p_0.

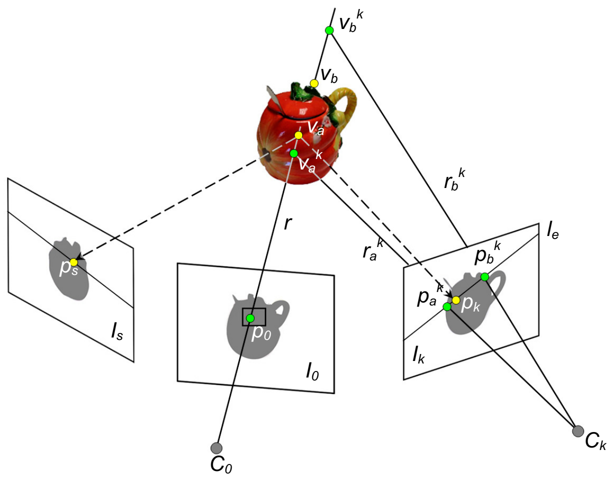


***Figure 4**. Finding the corresponding pixels of p_0 in other reference images.*

4.2 Highlight Sub-image Resampling

When the counterparts of a highlight sub-image are found, we could recalculate 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 of the corresponding region of current sub-image is out of highlight area (Fig. 5). Based on this fact, we assume 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 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 intensity level deviation of the pixels. That makes the calculation simpler. Let g_k be the intensity level of pixel p_k, and g_{avg} be the average level of $\{p_k \mid k=0\ldots n\}$, that is:

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

Then the weight of p_k could be defined as

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

Here, c_k is a constant coefficient to guarantees the sum of w_k to be 1. Finally, the blended RGB color of p_0 is calculated as

***Figure 5**. The counterparts of a highlight sub-image.*

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

where (R_k, G_k, B_k) is the color vector of p_k.

So far, 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, blending more pixel colors would also 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 own feature, i.e. the complexity of its texture.

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

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

and the deviation sum is

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

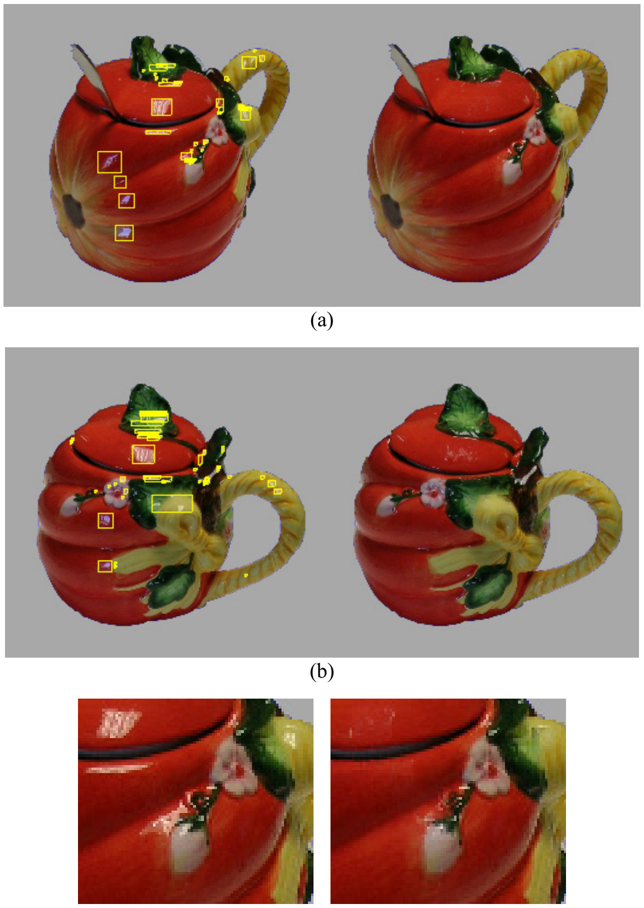
Then a given threshold τ determines the value of n:

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

When b_{dev} is larger than the threshold, it means that the sub-image has relatively more complex texture, and should be given a smaller n to prevent blurring; contrarily, when b_{dev} is smaller than the threshold, the sub-image is simpler in texture, and should be given a larger n for smoother result.

Experimental Results

Applying the method described above, we can remove highlights from input images and get satisfying rendering results. A group of input data usually includes about 20



(a)

(b)

Before After

(c)

Figure 6. *The result of removing highlight spots by pixel blending. (a) & (b) are two images from a sequence. The left are the original images with extracted highlight sub-images; and the right are resampled image without highlight spots. (c) The texture of the object is well preserved after resampling.*

images, taken around the object by a common digital camera and under natural illumination. Obvious highlight spots exist in each image.

Fig. 6 gives a group of experimental results of a red cup. There are 23 reference images used in total, and in sub-image resampling, N_{min} and N_{max} are set to be 4 and 9 respectively. Fig. 6(a) and (b) are two images from the sequence, the left part of each is the original image, and the small rectangles are automatically extracted highlight sub-images; in the right, the pixels in the sub-images are blended with their corresponding pixels, and result in images without highlight spots. Fig. 6(c) shows the detail of an image before and after highlight removal. We can see that while the highlights are smoothly removed, the texture of the cup is well preserved.

Another example of a white porcelain vase is given in Fig. 7. Note that the base color of the vase is so light that the contrast between the highlight spots and the vase is very weak. This would increase the difficulty to automatic

***Figure** 7. The experimental result of a white porcelain vase. The white base color makes it more difficult to automatically recognizing 按 and removing the highlight spots.*

highlight recognizing. In such case, applying more times of our algorithm to the image, or add interactive correction to the automatic selection will help to get better result.

More highlight removal results are shown in the left part of Fig. 8. With these resampled reference images, we can run visual hull method again to render the object at new viewpoints. The new rendering results are free from highlights and are more realistic and precise in color (right of Fig. 8).

Summary and Discussion

In this paper, we proposed a new method to remove highlight spots in image-based visual hull rendering. It takes advantage of the features of IBVH, therefore can rapidly remove the highlights by resampling the highlight sub-images from other reference images. Both of the recognition and the removal of highlight spots are automatic.

In fact, the application of our method is not limited in highlight removal. It can also be used in image repairing. For instance, if in one of the reference images, the object was partially blocked by some obstacles, it could be easily recovered from other images with our method, by interactive sub-image selection (As shown in Fig. 9).

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.

However, there are still some problems to be solved.

As an image-based method, our method requires the input images to be calibrated in advance, and the highlight removal results are quite sensitive to the accuracy of calibration, especially when the size of input image is small.

That is because incorrect correspondence of pixels would led to texture blurring.

Another question is how to determine the thresholds in highlight recognition and sub-image resampling. In current experiments, they are set according to experience and statistical analysis. Self-adaptive thresholds would make the method more efficient and produce more accurate results.

Finally, the calculation of the algorithm is not real-time so far. That will restrict the application of the method. Thus, improving the efficiency is also one of our future targets.

References

[1] G. Petschnigg, M. Agrawala and H. Hoppe, Digital Photography with Flash and No-Flash Image Pairs, *ACM Transactions on Graphics 23(3)*, pg. 664-672, 2004.

[2] A. Agrawal, R. Raskar, S.K. Nayar and Y. Li, Removing Photography Artifacts using Gradient Projection and Flash-Exposure Sampling, *ACM Transactions on Graphics 24(3)*, pg. 828-835, 2005.

[3] 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.

[4] Iddo Drori, Daniel Cohen-Or, Hezy Yeshurun. Fragment-Based Image Completion. *ACM Transactions on Graphics 22(3)*, pg. 303-312, 2003.

[5] J. Sun, L. Yuan, et al., Image Completion with Structure Propagation. *ACM Transactions on Graphics 24(3)*, pg. 861-868, 2005.

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

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

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

[9] 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.

[10] 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.

[11] 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.

[12] E. A. Khan, E. Reinhard, et al., Image-based Material Editing, *ACM Transactions on Graphics 25(3)*, pg. 654-663, 2006.

***Figure 8**. Visual hull rendering result without highlight. Top row: original reference images with highlight spots (left) and their rendering result (right); Bottom row: resampled reference images (left) and the new rendering result without highlights (right).*

***Figure 9**. The application of our method in image repairing. Left of each: original image with an obstacle blocking part of its texture; Right of each: obstacle removed with our method, by interactive selection.*