Visual Hull Construction in the Presence of Partial Occlusion

Li Guan, Sudipa Sinha, Jean-Sébastien Franco, Marc Pollefeys

Visual Hull Construction in the Presence of Partial Occlusion

Li Guan Sudipta Sinha Jean-Sébastien Franco Marc Pollefeys
Department of Computer Science
The University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175, USA
{lguan, ssinha, franco, marc}@cs.unc.edu

Abstract

In this paper, we propose a visual hull algorithm, which guarantees a correct construction even in the presence of partial occlusion, while "correct" here means that the real shape is located inside the visual hull. The algorithm is based on a new idea of the "extended silhouette", which requires the silhouette from background subtraction and the "occlusion mask" of the same view. In order to prepare the occlusion mask, we also propose a novel concept of "effective boundary" of moving foreground objects in a video obtained from a static camera. The accumulation of the effective boundary through time automatically gives robust occluder boundaries. We theoretically prove that our algorithm deterministically computes the tightest, correct visual hull in the presence of occlusion. Both synthetic and real examples are given as a demonstration of the correctness of the algorithm. Finally we analyze that this new algorithm is still within the time complexity of the traditional method.

1. Introduction

A visual hull is defined as the intersection of silhouette cones from 2D camera views, which captures all geometric information given by the image silhouettes [1]. One basic property is that a visual hull is the largest volume to be consistent with silhouette information from all views [2]. The actual shape is always contained inside the constructed visual hull. This is referred to as the *visual hull conservation constraint*. Hence multiple view shape-from-silhouette algorithms for computing the visual hull are widely used to provide a coarse 3D shape estimate which can serve as an initialization for more sophisticated 3D shape reconstruction algorithms, such as [3].

The visual hull conservation constraint will be violated when an algorithm ignores the presence of occlusion in the scene. If an object is partially occluded in a certain view, most background subtraction algorithms [4, 5] will produce an incomplete silhouette. When such silhouettes are used, the construction will be incomplete too. This is illustrated in Figure 1[1].

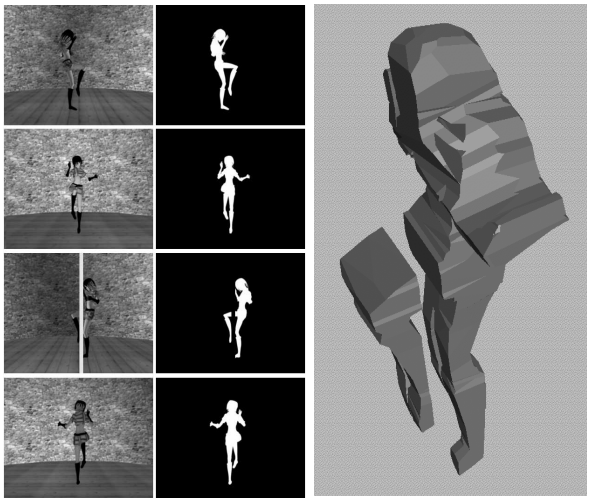


Figure 1. The effect of occlusion on visual hull construction is demonstrated using a synthetic experiment. Column 1 shows the input images and column 2 shows the extracted silhouettes. The silhouette in the third view is incomplete due to the presence of an occluder (a vertical pole). The resulting visual hull shown on the right violates the conservation constraint.

While previous visual hull methods [6, 7, 8, 9] focus on algorithm efficiency and accuracy of the visual hull estimate, they assume the silhouettes to be free of occlusion. For obtaining correct visual hulls of objects from real video footage, where occlusion is common (see Figure 2), this assumption needs to be relaxed. Such methods can only compute a correct visual hull by discarding the occluded views.

[1] Data from www.mpi-sb.mpg.de/departments/irg3/kungfu/index.html

While completely occluded views do not provide any information, partially occluded silhouettes are still useful for inferring 3D shape. Hence in the rest, we focus on partially occluded silhouettes. We also assume that the foreground object never moves to the front of the occluders, which is usually the case when static occluders are close enough to the camera or the moving range of the foreground object is not very wide.

This paper makes two contributions. First of all, in order to deal with partial static occlusions, a new algorithm is introduced as an extension to traditional shape-from-silhouette visual hull method. Images where occlusions occur are not simply discarded. With the help of an *occlusion mask*, the non-occluded part of the silhouette can still be used. The traditional method can be proved as a special case of our approach. Secondly, based on a novel concept of *effective boundary*, an automatic occlusion mask extraction algorithm is described, which uses spatiotemporal cues as well as silhouette accumulation.

In Section 2, we describe the automatic procedure to compute the occlusion mask for each camera view. We present the new visual hull algorithm in Section 3. We validate our approach and evaluate its effectiveness in Section 4 followed by our conclusions in Section 5.

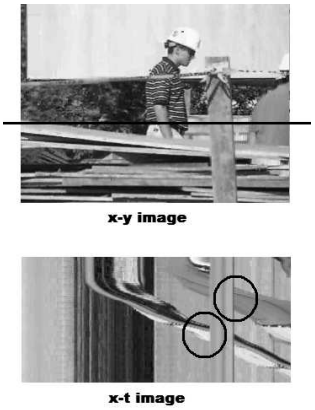


Figure 2. T-junction detection failure due to similar color. Top: workers are walking towards the right. Bottom: The horizontal line of pixels in the top image stacked over time. The two circles indicate occlusions. The right one is a T-junction. But no T-junction is detected for the left one, since the colors of the occluder and the background are similar.

2. Occlusion Mask Extraction

An *occlusion mask* is a view-dependent binary image. For every pixel, 1 denotes that an occluder is on the viewing direction and 0 otherwise. It can be considered as a part of the camera configuration, similar to the calibration parameters. Generally, regions outside of the camera window belong to the occlusion mask. But as long as the 3D object being constructed is guaranteed to be seen by all cameras, we can crop the occlusion mask to just the camera window size.

If only static occluders are considered, an occlusion mask can be precalculated. Automatic occluder segmentation has been studied before. N. Apostoloff *et.al.* [10] recently analyzed occlusions in spatiotemporal volumes using "T-junctions" and introduced a learning framework to improve occlusion detection. The T-junction—a photometric profile shaped like a "T", which is formed where the edge of an object occludes the change in intensity of another object, is a natural indicator of occlusion [11]. An example is shown inside the right circle in Figure 2. However there could be problems in two situations, both related to pixel colors. First, when the occluder and background have similar colors, T-junctions cannot be reliably detected for the occluder boundary (see the left circle in Figure 2). Secondly, when a moving object has stripe-like textures, in the spatiotemporal volume, T-junctions are detected due to the texture illusion, where no occlusions really exist (refer to Figure 3).

Other methods have been tried to bypass these problems. G. Brostow *et.al.* [12] use moving foreground object silhouettes as "active brushes" to delineate the underlying layers of the scene. The union of silhouette pixels from the whole sequence forms the *Cumulated Silhouette Image* (refer to Figure 4). Edges in this binary image are categorized into two types: (1) the occluder boundaries; (2) boundaries that between regions where the moving objects have been projected onto and regions from where only the background has been seen. We call the second type *Phantom Boundaries*. In Figure 4, the roughly horizontal envelope of the workers' helmets is an example of a Phantom Boundary (there is no such edge in the original image).

A further analysis reveals that in the Cumulated Silhouette Image, what contributes to the final image boundary is the foreground silhouette boundary in every frame. For most of the time, the object is occluded by the occluder, therefore the foreground object boundary coincides with the occluder boundary. But at the Phantom Boundary, the foreground object boundary has nothing to do with the occluder. So for the purpose of acquiring the occluder boundary, we should not accumulate the foreground silhouette or its complete boundary, but only part coincides with the occluder boundary. Therefore we introduce the idea of an *Effective Boundary*.

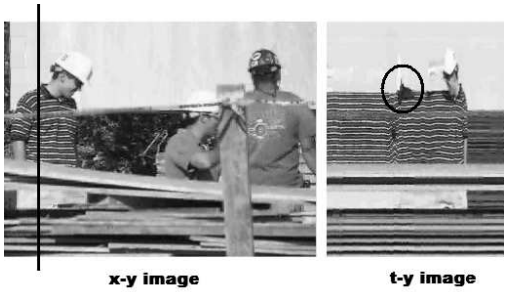


Figure 3. T-junction detection failure due to texture illusion. Left: the leftmost worker is turning his body around. Right: The vertical line of pixels in the left image changing with time. The circle indicates a T-junction caused by stripe-like T-shirt texture "intersecting" with the collar and helmet, where there is no real occlusion.

Figure 4. Cumulated Silhouette Image of the "construction site" video sequence. The binary boundaries are occlusion boundaries, except for the roughly horizontal curve at the top—the Phantom Boundary, above which foreground moving objects (the walking workers) have never reached.

2.1. Effective Boundary of a Moving Object

From T-junction analysis we know that *occlusion happens when the general motion of a foreground object is suddenly prohibited by the occluder.* This can be observed only for the silhouette edges perpendicular to which the motion vector has a non-zero component, but not for the silhouette edges parallel to the motion. The Cumulated Silhouette Image also reveals that it is the latter silhouette edges that cause the Phantom Boundaries (most of the motions in the sequence are horizontal, which is parallel to the top of the helmets), while the actual occluder boundaries are contributed by the former ones. We now define the *Effective Boundary* as the "frontal parts" of the moving object silhouette in both the motion and its reverse direction (since the detection via reverse motion is also valid), see Figure 5.

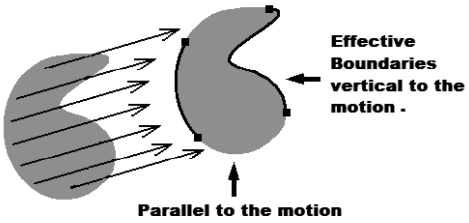


Figure 5. Effective Boundary of a synthetic moving object silhouette denoted in thick black curves. Note that the silhouette could be concave.

Now, to obtain the occlusion mask for a certain camera view, we take a short video sequence with objects moving around in the view. We then accumulate the Effective Boundaries as valid candidates for occluder boundaries and call this accumulation the *Cumulated Occluder Boundary Image* (an idea similar to the Cumulated Silhouette Image).

Specifically, after background subtraction, we obtain a moving object silhouette for every frame. Here, the subtraction does not have to be very accurate, because we assume after the information accumulation over time, we can cancel out the effect of random noise. Then we compute the *Motion History Image* (MHI) [13], an image whose pixel intensity is a function of the recency of motion in a sequence, as shown in Figure 6. From there, we can get the motion direction of the objects by averaging the gradients of pixels in the rectangle boxes, which are the bounding boxes of connected non-zero pixel regions above a certain size (for Figure 6, the minimum size is 100 pixels). Then we can accumulate the Effective Boundary over time. Figure 7 shows the Effective Boundary accumulation over the "construction site" sequence. The accumulation is robust to noise. Even though some pixels might be falsely labeled as occluder boundary during the accumulation, they will be discarded and will not show up in the final occluder boundary image, as long as they have ever been or will ever be occupied by some foreground silhouette in the whole video sequence. This is because occluder pixels are guaranteed not to be occupied by foreground objects. In other words, if a pixel has ever been occupied during the video sequence, it is definitely not an occluder pixel. For example, in Figure 7(b) (frame 100), there are some false accumulations above the wooden post, but they are occupied by some foreground silhouette between frame 101 and 274, so they are eliminated in Figure 7(c) (frame 274).

After processing the whole video sequence, we can further refine the occluder boundary by the Cumulated Silhouette Image, because boundary connectivity in this image is always preserved, despite of the Phantom Boundary prob-

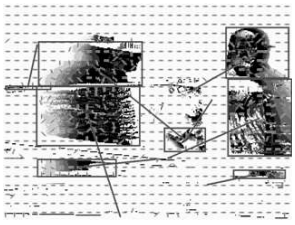
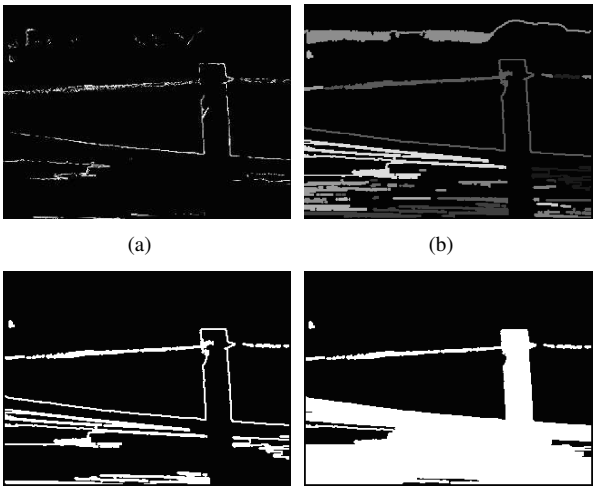


Figure 6. Using MHI to determine motion directions. The dark blobs are moving silhouettes. The long line segments indicate the motion direction of the blobs in rectangles. The tiny line segment array shows the local gradient field.

Figure 7. The Occluder Boundaries accumulation over time at frame 5, 100 and 274 respectively. No Phantom Boundary exists in the final image.



(a) (b)

(c) (d)

Figure 8. (a) the Cumulated Occluder Boundary Image after the final frame. (b) boundaries of the Cumulated Silhouette Image, as in Figure 4. Different intensities indicate the connectivity. (c) choosing final boundaries from (b) using pixel voting from (a). No Phantom Boundary here. (d) final occlusion mask by flood-filling.

lem. The idea is to use boundary pixels in the Cumulated Occluder Boundary Image to vote for boundaries in the Cumulated Silhouette Image. Since the Phantom Boundary should have few votes from the Cumulated Occluder Boundary Image (if some, mainly are due to noise), after thresholding, we can discard the Phantom Boundary from the Cumulated Silhouette Image and get a connected occluder boundary image as shown in Figure 8(c). To compute the occlusion mask from this image, we randomly choose a pixel that is not on the boundary as a seed and flood-fill the image. If the resulting mask overlaps with regions that a foreground object has ever reached, we discard the result, otherwise we keep it. We repeat this process until no more non-boundary pixels in the image can be used as a seed. The final occlusion mask is shown in Figure 8(d).

The *Effective Boundary* is updated online, so that one can keep processing the video until a good occluder mask is obtained. This is robust to errors in silhouette extraction. We have implemented an OpenCV real-time version which works well with fairly low resolution video. When the occluder color is similar to the background, it is hard to manually segment the occluder. Our algorithm works well in this case, as long as the moving object color differs from that of the background and the occluder. More results are shown in Section 4.3 as well as the supplemental video.

3. Main Algorithm

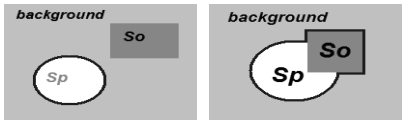
Once given the silhouette of the object and the occlusion mask of the camera, we need to determine whether the object has been occluded in the view or not. Table 1 lists the notations we use in this paper.

Table 1. Symbol Definition

P	object of interest in 3D scene
O	occluder in 3D scene
S_k^P	silhouette of the 3D object P in kth view
S_k^O	silhouette of the occluder O in kth view
H	visual hull
∂x	boundary of a 2D region x
$C_k(x)$	viewing cone of the 2D area x from kth view
$R_k(X)$	silhouette of the 3D object X in kth view

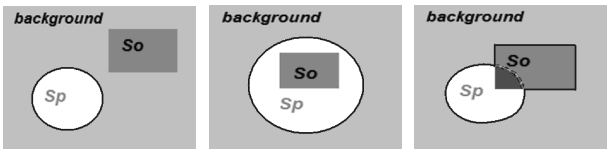
3.1. Occlusion Detection

The silhouettes of the occluder can be considered as view-dependent information for each camera, denoted by S_k^O. For every image frame, we first extract the silhouette S_k^P of the active object P (note that due to occlusion, S_k^P may be incomplete). Then using S_k^P and S_k^O, we can per-



(a) $Unoccluded$ (b) $Occluded$

Figure 9. Object-occluder configuration. The dark boundaries in both figures indicate the boundary of the extended silhouettes ∂S_k^E.



(a) (b) (c)

Figure 10. Assume that the ellipse is $R_k(P)$, and the rectangle is S_k^O. (a) and (b) show two special cases that $R_k(P)$ is equal to S_k^E. (c) $R_k(P) \subseteq S_k^E$. The sector region is actually being occluded. In all images, the this solid outline is ∂S_k^E

form occlusion detection to determine if an occlusion occurs in that frame as follows.

Fact No. 1: If S_k^P and S_k^O do not share boundaries, there is no occlusion in kth view, as shown in Figure 9(a). To express it in a formal way:

When $\partial S_k^P \bigcap \partial S_k^O = \emptyset$, there is no occlusion in the view; we now define a new term *extended silhouette* as S_k^E, and this means (1) when an occlusion is detected in kth view, S_k^E is the union of the silhouette of the object and that of the occluder; (2) while there is no occlusion happening in the image, S_k^E is equal to that of the object only. Speaking of both configurations in Figure 9, S_k^E is the region outlined by the thick boundary.

$$S_k^E = \begin{cases} S_k^P & when\ \partial S_k^P \bigcap \partial S_k^O = \emptyset, \\ S_k^P \bigcup S_k^O & otherwise. \end{cases} \quad (1)$$

The worst case happens when two objects in 2D are tangential with each other in the view, but no occlusions behind. However in order to preserve the *visual hull conservation constraint*, we have to bear this case, since we cannot know the situation behind the occluding area S_k^O.

Fact No. 2: $R_k(P) \subseteq S_k^E$, where $R_k(P)$ is the "real silhouette" of object P without occlusion. In other words, $R_k(P) - S_k^P$ is the part of the object being occluded.

Fact No. 2 shows that "real silhouette" of P is contained in the "extended silhouette" in every view. $R_k(P) = S_k^E$, as a special case, happens (1) when there is no occlusion in this view, namely $\partial S_k^P \bigcap \partial S_k^O = \emptyset$, or (2) S_k^O is completely enveloped by S_k^P, which is expressed as $S_k^O \subset S_k^P$. As shown in the first two cases of Figure 10.

3.2. Main Algorithm

Here is the main algorithm for the construction of visual hull in the presence of occluders.

Algorithm 1 General Visual Hull Construction Algorithm

1: Occlusion mask extraction for every camera view;
2: Camera calibration;
3: To form "extended silhouette" S_k^E of P in view k, $k \in [1, n]$, n is the number of camera views, according to silhouette boundary overlapping;
4: To construct visual hull $H_{with\ occluder}$ with all extended silhouettes, *i.e.* $H_{with\ occluder} = \bigcap C_k(S_k^E)$, $k \in [1, n]$.

4. Algorithm Performance Analysis

In this section, we first analyze the correctness of the algorithm, for which we prove that the constructed visual hull satisfies the *visual hull constraint of conservation*—the real object is completely located inside the visual hull. Then we compare the shape of our visual hull with the one constructed with traditional method without occlusion and analyze the time complexity of the algorithm. Finally some examples are demonstrated.

4.1. Correctness of the Algorithm

Theorem 1: $H_{with\ occluder}$ ***preserves the conservation.***
Proof:

Let's denote $H_{without\ occluder}$ as the occlusion-free visual hull, for which we have

$$H_{without\ occluder} = \bigcap_{k \in [1,n]} C_k(R_k(P)) \quad (2)$$

We also have for the result of our algorithm

$$H_{with\ occluder} = \bigcap_{k \in [1,n]} C_k(S_k^E) \quad (3)$$

According to Fact No. 2 in Section 3, $R_k(P) \subseteq S_k^E$, for all $k \in [1, n]$, we have $C_k(R_k(P)) \subseteq C_k(S_k^E)$. And since the intersection of all pair-wise smaller sets is also smaller than the intersection of all the pair-wise larger sets, we always have $H_{without\ occluder} \subseteq H_{with\ occluder}$.

Because $H_{without\ occluder}$ has already been preserving the *constraint of conservation*, $H_{with\ occluder}$ preserves the constraint as well.

End of proof □

4.2. Shape Comparison with $H_{without\ occluder}$

Theorem 1 shows that the tightest $H_{with\ occluder}$ we can get is $H_{without\ occluder}$, and this only happens when $S_k^E = R_k(P)$ for all $k \in [1, n]$, as special cases discussed in Fact No. 2 in Section 3. But Theorem 1 does not give us an upper bound of $H_{with\ occluder}$.

We now analyze how much extra volume is added to preserve correctness in the presence of occlusion.

Define the "$n-1$ hull with respect to kth view", denoted by H_k^{n-1}, as follows.

$$H_k^{n-1} = \bigcap_{j\in[1,n],j\neq k} C_j(S_j^E) \tag{4}$$

Basically, it represents the visual hull constructed with $n-1$ views (all n views except the kth) and then re-projected onto kth view.

Theorem 2: $R_k(P) \subseteq R_k(H_k^{n-1})$ ***for all*** $k \in [1, n]$.

Proof:

From Theorem 1 (Section 4.1) and Equation (4), it follows that H_k^{n-1} preserves the visual hull conservation constraint.

So $R_k(P) \subseteq R_k(H_k^{n-1})$ is always true.

End of proof □

This means that the re-projection of the $n-1$ hull to kth view contains the silhouette of the object in kth view.

Fact No. 2 in Section 3 tells us that for the first two configurations, $R_k(P) = S_k^E$.

So in either of the following cases:

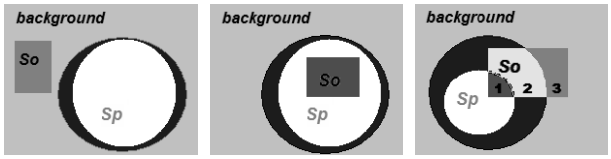
- *No occlusion in the view, as shown in Figure 11(a);*
- *The occluder is completely contained within the object, as shown in Figure 11(b).*

the following is true

$$H_{with\ occluder} = H_{without\ occluder} \tag{5}$$

This means we have fully recovered the occluded parts for our visual hull.

The first condition also means that the traditional visual hull algorithm without the presence of the occluder (see Figure 9(a)) is a special case of our algorithm. For Figure 11(c), the rectangle S_k^O is separated into three parts. Part 1 contributes to preserving the correctness. Part 3 does not affect $H_{with\ occluder}$, because the corresponding region in 3D has already been outside of H_k^{n-1}. Therefore, the only part adding extra volume to $H_{with\ occluder}$ due to occlusion, is part 2. We can express this as follows.



(a) (b) (c)

Figure 11. Three general configurations of a scene from kth view. Let the larger ellipse represent $R_k(H_k^{n-1})$, the smaller S_k^P, the rectangle S_k^O. (a) no occlusion (b) S_k^O completely contained in S_k^P. (c) partial occlusion, the most common case.

$$S_k^{extra} = (R_k(H_k^{n-1}) - R_k(P)) \bigcap S_k^O \tag{6}$$

And it will add to $H_{with\ occluder}$ amount of

$$C_k(S_k^{extra}) \bigcap H_k^{n-1} \tag{7}$$

Since we do not know the shape behind the occluder in advance, there is no way to decrease the area of part 2 in Figure 11(c). The best we can get for each view is the extended silhouette as we defined. Therefore, we claim that our approach gives the tightest visual hull in the presence of occlusion. In some cases, $C_k(S_k^{extra})$ may be significantly large, but this is the smallest visual hull which satisfies the conservation constraint in the presence of occlusion.

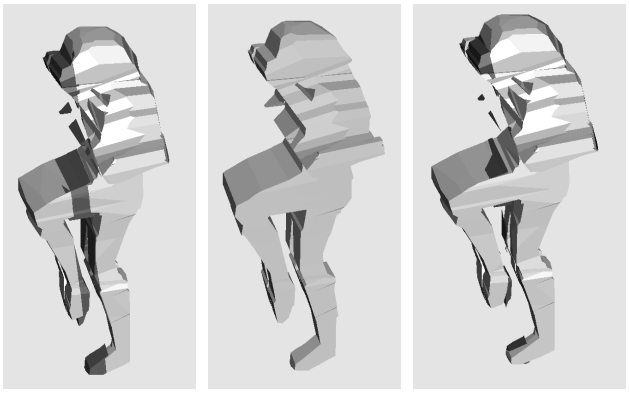
Note that the region between the large and small ellipses in Figure 11 is the amount of redundancy we have carved away by using the partially occluded view, instead of discarding the view and using only $n-1$ views to get the visual hull as the traditional method does. And this illustrates the original idea why we want to use partially occluded views—to use as much information as we have to get a better representation of the 3D shape.

This algorithm also works when multiple views have occlusion. However as the degree of occlusion increases, less information is available in all the views. Hence to preserve the conservation constraint, the visual hull may be larger than $H_{without\ occluder}$. Since we use S_k^E instead of S_k^P, our algorithm has the same time complexity as the traditional method.

4.3. Examples

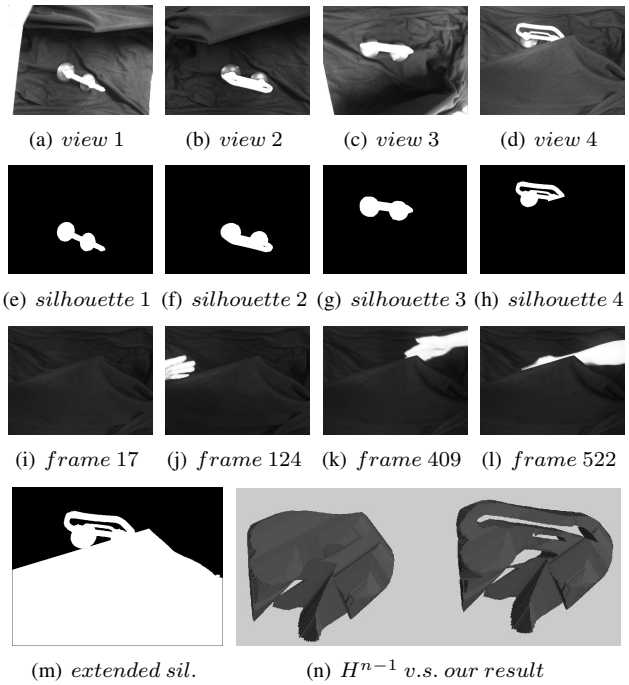
4.3.1 Synthetic Kung Fu Data

This example takes input data as those shown in Figure 1. The result is displayed in Figure 12. Figure 12(a) shows



(a) $H_{with\ occluder}$ (b) H^{n-1} (c) $H_{without\ occluder}$

Figure 12. (a) visual hull $H_{with\ occluder}$ with our approach. (b) occlusion-free visual hull H_k^{n-1} with only 3 occlusion-free views. It is larger than that in (a). (c) $H_{without\ occluder}$. It is a little bit smaller than (a).



(a) $view\ 1$ (b) $view\ 2$ (c) $view\ 3$ (d) $view\ 4$

(e) $silhouette\ 1$ (f) $silhouette\ 2$ (g) $silhouette\ 3$ (h) $silhouette\ 4$

(i) $frame\ 17$ (j) $frame\ 124$ (k) $frame\ 409$ (l) $frame\ 522$

(m) $extended\ sil.$ (n) H^{n-1} $v.s.$ $our\ result$

Figure 13. Row 1 is the 4 camera views. Row 2 is the silhouette images from background subtraction. Row 3 is the process to acquire occlusion mask for (d). (m) is the extended silhouette for view 4. (n) H_k^{n-1} and $H_{withoccluder}$ of our algorithm. Note that our visual hull preserves the correct topology.

the result of our algorithm. Comparing with Figure 1, the occluded part (the dark red region) is correctly recovered. H_k^{n-1} in Figure 12(b) only uses the occlusion-free view 1, 2 and 4, and is fatter than our visual hull in Figure 12(a) (especially see the chest part), meaning that our method obtains a tighter visual hull representation. Because this is a synthetic data set, we are able to get the foreground silhouettes without occluder, thus construct $H_{without\ occluder}$, as shown in Figure 12(c). The difference between Figure 12(c) and our result (Figure $H_{with\ occluder}$) is displayed as the dark green part in Figure 12(c). As one can see, there is little region marked dark green, meaning in this example, the visual hull constructed from our method is very close to the optimal (occlusion-free) situation.

4.3.2 Lab Data and Effective Boundary Accumulation

Taken from a lab camera network setting of PointGrey DragonFly 640x480 color cameras working at 15 fps, we apply the "effective boundary" algorithm in advance to obtain occlusion masks for all views. Row 3 of Figure 13 shows frame 17, 124, 409 and 522 in the video sequence (609 frames in total) of view 4. The similar colors of the occluder and background make manual segmentation difficult for this data set. But our automatic algorithm works well with a moving hand whose color is distinguishable from that of the background and the occluder. Figure 13(m) shows S_k^E of the same S_k^P in Figure 13(h). In order to get no seam at the shared boundary between occlusion mask and S_k^P, we dilate the occlusion mask 2 pixels outwards and take the union of it with S_k^P. Figure 13(n) shows two visual hulls. The one on the left is constructed with only three occlusion-free views. The one on the right is constructed with all extended silhouettes from four views using our proposed algorithm. Of the two, only the right one preserves the two holes in the object. This shows that to preserve the right topology, we had better not discard the partially occluded view during construction.

4.3.3 Real Office Data

As shown in Figure 14, this example deals with cluttered office scene, where quite a few occluders are in the presence, e.g. a chair, a lamp and a table. Except in view 3, the person is always partially occluded and thus not valid for traditional construction method. The left visual hull shows that the person is hardly recognizable in real environment if not using the "extended silhouettes". In comparison on the right, our algorithm computes a correct visual hull. Considering the number of cameras and complexity of the scene, the construction is fairly effective.

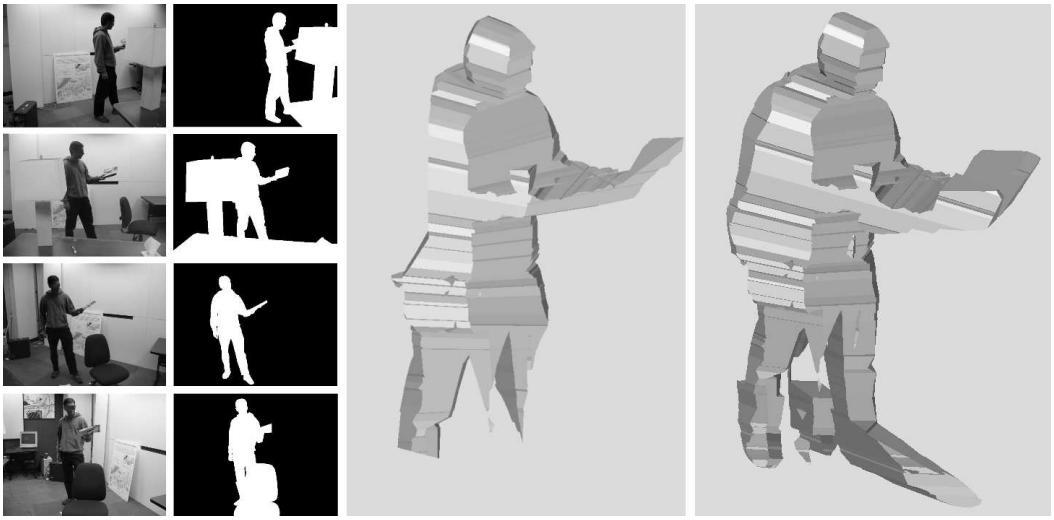


Figure 14. 3 out of 4 views have partial occlusion, which is very common in indoor scenes.

5. Conclusion

In this paper, we extend the traditional shape-from-silhouette visual hull algorithm to work in static partial occlusion situations. We are not using silhouettes obtained from background subtraction directly, but using an *Extended Silhouette* combining both foreground object silhouette and occlusion information in that view. We also provide a novel method to automatically and robustly detect and segment static occluders using a new concept—*Effective Boundary* of a moving foreground object in the scene. We also prove that the constructed visual hull contains the actual 3D object and it is the smallest one that we can get using binary deterministic silhouette representation. Moreover, the time complexity of the construction is the same as the traditional method.

Acknowledgments

We acknowledge the support of a Packard Fellowship for Science and Technology, the NSF CAREER award IIS 0237533 and the US National Library of Medicine contract N01LM33514 (3D Telepresence for Medical Consultation).

References

[1] B. G. Baumgart. Geometric modeling for computer vision. *Technical Report Artificial Intelligence Laboratory Memo AIM-249*, Stanford University, 1974.

[2] A. Laurentini. The Visual Hull Concept for Silhouette-Based Image Understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 94(16), 1994.

[3] O. Faugeras and R. Keriven. Variational principles, surface evolution, PDE's, level set methods and the stereo problem. *IEEE Transactions on Image Processing*, 98(7):336—344, 1998.

[4] C. Stauffer, W. Grimson. Adaptive background mixture models for real-time tracking. *In Proc. CVPR*, 1999.

[5] A. Elgammal, D. Harwood, and L. Davis. Non-parametric Model for Background Subtraction. *In Proc. ICCV FRAME-RATE Workshop*, 1999.

[6] M. Potmesil. Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics and Image Processing*, 87(40):1—20, 1987.

[7] G. K. M. Cheung, T. Kanade, J-Y. Bouguet and M. Holler. A real time system for robust 3D voxel reconstruction of human motions. *In Proc. CVPR*, 00(2):714—720, 2000.

[8] C. R. Dyer. Volumetric Scene Reconstruction from Multiple Views. *Foundations of Image Understanding*, L. S. Davis, ed., Kluwer, Boston, 2001.

[9] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L.McMillan. Image Based Visual Hulls. *In In Proc. Siggraph*, 2000.

[10] N. Apostoloff and A. Fitzgibbon. Learning spatiotemporal T-junctions for occlusion detection. *In Proc. CVPR*, 2005.

[11] P. Favaro, A. Duci, Y. Ma, and S. Soatto. On Exploiting Occlusions in Multiple-view Geometry. *ICCV*, 2003.

[12] G. Brostow and I. Essa. Motion Based Decompositing of Video. *ICCV*, 1999.

[13] J. Davis. Hierarchical Motion History Images for Recognizing Human Motion. *IEEE Workshop on Detection and Recognition of Events in Video*, Canada, 2001.