MULTI-VIEWPOINT SILHOUETTE EXTRACTION WITH 3D CONTEXT-AWARE ERROR DETECTION, CORRECTION, AND SHADOW SUPPRESSION

S. Nobuhara, Y. Tsuda, T. Matsuyama, and I. Ohama

Advanced School of Informatics, Kyoto University, Japan

Keywords: Silhouette extraction, Multi-viewpoint images

Abstract

This paper presents a novel approach for silhouette extraction from multi-viewpoint images. The main contribution of this paper is a new algorithm for 1) 3D context-aware error detection and correction of 2D multi-viewpoint silhouette extraction and 2) 3D context-aware classification of cast-shadow regions. Some experiments demonstrate advantages against previous approaches.

1 Introduction

In recent years, the importance of silhouette extraction is increasing because many 3D shape reconstruction algorithms utilize the visual hull of the object given by SFS (shape-from-silhouette) method [1–3,8,11–15]. These algorithms use visual hulls as their initial estimations, and then refine them based on multi-viewpoint textures and/or other reconstruction cues.

In silhouette extraction for 3D shape reconstruction, it is reasonable and widely used to assume that we can use the background images on every viewpoint and can apply background subtraction individually. However, one of the most fundamental difficulties in the monocular silhouette extraction is ambiguities due to the comparison between the similar foreground and background colors.

To overcome this problem, we introduce an algorithm which can exploit the multi-viewpoint environment since we capture a common object from the different viewpoints for 3D shape reconstruction. First we model the observed images as a union of the following three types regions: (1) object regions, (2) cast-shadow regions, and (3) background regions. Then we define an algorithm based on the following two questions – *"Can a 2D region carving of the silhouette on a viewpoint be acceptable on the other viewpoints? If not, how can we correct it?"* and *"Can a 3D region be recognized as a cast-shadow region on every viewpoint?"* We utilize them as an inter-viewpoint constraint and introduce a 3D context-aware error detection, correction and shadow suppression for multi-viewpoint silhouette extraction of the object.

This paper is organized as follows. In Section 2, we review related work with emphasis on the differences compared to our algorithm. We describe our algorithm in Section 3, and evaluate it against other approaches in Section 4. We conclude our paper and discuss possible future work in Section 5.

2 Related work

For multi-viewpoint silhouette extraction, various approaches have been proposed so far [5] [9] [16] [4] , but they does not have explict error detection and correction on silhouette regions based on 3D geometry. Zeng and Quan proposed a method which uses an inter-viewpoint constraint and 2D segmentation of captured images [16]. Their method realized silhouette extraction without background images. However it totally depends on the accuracy of 2D segmentation of captured images and can neither detect nor correct any errors. This is because their method requires perfect segmentation of captured images which separates the object and background regions. Goldlücke and Magnor proposed a method which realized simultaneous 2D segmentation and 3D reconstruction based on graph-cuts [4]. However, it requires a dense full 3D depth-map of the background such as the studio floor, walls, and all the visible items, and it has a limitation on the camera arrangement as same as voxel coloring. Guillemaut [5] *et al.* also proposed a novel framework for multi-viewpoint environment, but it is not possible to recover errors on pixels such that foreground and background colors are quite similar. This is because their "conservative visual hull" can *fill* holes only if each of their projection is near by the pixels with vivid differences between foreground and background.

For shadow suppression in silhouette extraction, one of the most simple and effective approaches is intensity and chromaticity based categorization of shadow regions [7]. In this approach, each pixel is categorized as shadow region if it is similar to the background pixel in chromaticity but different in intensity. This approach provides significant improvement in comparison with simple background subtraction using a certain threshold. However, these 2D image based silhouette extraction do not consider 3D context of each pixel. They can suppress real cast-shadows on the floor, but they also suppress self-shadow regions on the object surface and darkly-textured regions like black hair.

The main advantages of this paper are the following points. (1) Our algorithm can detect and correct errors on the 2D segmentation of silhouettes based on an inter-viewpoint validation process, and (2) can suppress shadows with 3D context consideration. While our algorithm requires a 3D geometry of the studio floor, it is known for calibrated 3D capturing environment (described below).

3 Algorithm

The goal is to obtain the object silhouettes on every viewpoint which are consistent with each other. To achieve this, we model the observed images can be decomposed into (1) object regions, (2) cast-shadow regions, and (3) background regions. We denote this by $\mathsf{Img} \rightarrow \mathsf{Obj} + \mathsf{CS} + \mathsf{Bg}$. Based on this modelling, we introduce a two-step approach. The first step extracts $\mathsf{Obj} + \mathsf{CS}$ from Img and the second step extracts Obj from $\mathsf{Obj} + \mathsf{CS}$. As described in Section 1, the key concept of both algorithms is 3D-context.

Our algorithm assumes calibrated cameras, known background images, and the floor plane where the cast-shadows of the object can appear. This floor assumption is reasonable for 3D shape reconstruction based on SFS. This is because conventional voxel-based SFS implementations require a certain bounding box which encages the object, and it is adequate to use the studio floor as the bottom of the bounding box.

In what follows, we denote the number of cameras by N, the i-th camera by C_i, the silhouette on C_i at t-th iteration step by $\mathrm{Sil}_i(t)$, the segmented image on C_i at t-th iteration step by $\mathrm{Seg}_i(t)$, the visual hull computed with silhouettes at t-th step by $\mathrm{VH}(t)$, the projection of $\mathrm{VH}(t)$ on C_i by $\mathrm{Svh}_i(t)$.

3.1 The 3D context-aware extraction of silhouettes and shadows

First, we introduce an algorithm which extracts both object regions and cast-shadow regions from multi-viewpoint images. That is, we extract $\mathsf{Obj} + \mathsf{CS}$ from Img on each viewpoint. To achieve this extraction, we utilize two constraints proposed by Zeng and Quan [16].

Intersection constraint (IC) : Projection of the visual hull which is computed from silhouettes on every viewpoint should be equal to the silhouette on each viewpoint.

Projection constraint (PC) : Projection of the visual hull should have outline which matches with apparent edges of captured image on each viewpoint.

As described in [16], these two constraints produce a set of silhouettes such that each of them is NOT the background region. That is, silhouettes given by these constraints cannot suppress shadows cast by the object itself because cast-shadows on the floor also satisfy IC and PC. That is, IC and PC enable us to extract $\mathsf{Obj} + \mathsf{CS}$ from Img. In addition to them, we introduce the following constraint to realize 3D context-aware error detection and correction.

Background subtraction constraint (BC) : The sum of differences between the background and captured image in projection of the visual hull should be greater than a certain threshold.

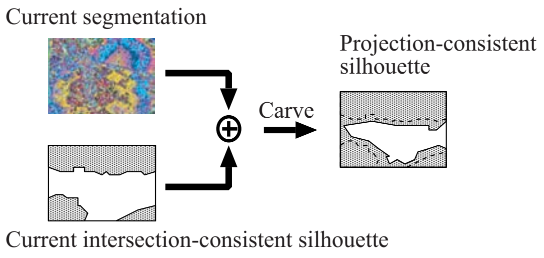

Current segmentation
Projection-consistent silhouette
Carve
Current intersection-consistent silhouette

Figure 1: Silhouette carving

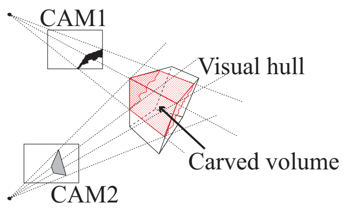


Figure 2: Intersection consistency

We employ an iterative process in which we carve silhouettes per segments so that they satisfy these three constraints. Note that our method based on the image segmentation, but segmentation errors are corrected through the iteration.

3.1.1 Silhouette carving

In each iteration step, we carve a silhouette so that it satisfies PC. We apply color-based segmentation on captured images, and consider that a silhouette satisfies PC if it is composed by a set of segments. So we define our carving algorithm as follows:

- For each segment of the captured image,
  - If any portion of the segment is located outside the silhouette, carve the whole segment region from the silhouette.

We denote this operation by $\mathrm{Carve}(\mathrm{Sil}_j(t), \mathrm{Seg}_j(t))$, where $\mathrm{Sil}_j(t)$ and $\mathrm{Seg}_j(t)$ denote the silhouette and segmented image of Camera j at t-th iteration respectively. Figure 1 illustrates this operation. Note here that this operation is done for all segments, and the order of selection does not affect the result.

3.1.2 The 3D Context-aware error detection and correction

Suppose we have a set of silhouettes which satisfies IC. If we carve a silhouette Sil_j on camera C_j so that it satisfies PC, the visual hull will also be carved and its projection on each camera will be equal to or smaller than the original silhouette which satisfies IC. For example, if we carve the black area of CAM_1 in Figure 2, the corresponding volume of the visual hull is also carved, and it is observed as the removal of the gray area on CAM_2. That is, carving one of the intersection-consistent silhouettes makes it projection-consistent, but breaks intersection-consistency between other silhouettes. Here, if the changes between the projection of visual hull $\mathrm{Svh}_i(t)$

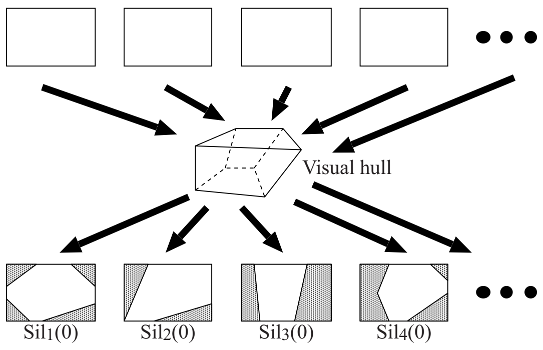


Figure 3: Initial silhouettes

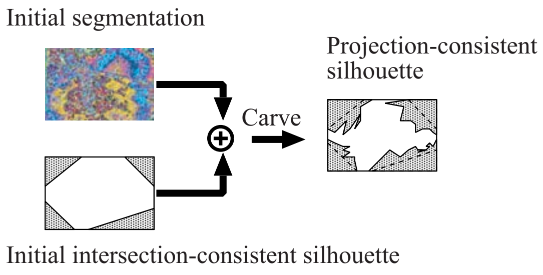

Initial segmentation
Projection-consistent silhouette
Carve
Initial intersection-consistent silhouette

Figure 4: First iteration step

and the original silhouette $\mathrm{Sil}_i(t)$ are *acceptable* on the other viewpoints, we take the projections of visual hull as new silhouettes. We use BC to determine if it is acceptable or not (described below). This process makes the silhouettes satisfy IC again, and PC on Sil_j as well.

On the other hand, if the changes are not acceptable in terms of BC, it is clearly correct to regard that the last carving of the silhouette is wrong, *i.e.*, segments carved in the last $\mathrm{Carve}(\mathrm{Sil}_j(t), \mathrm{Seg}_j(t))$ operation are too large and they contain not only the background but also the object regions. This indicates that we need re-segmentation of segments in question into smaller segments, and we can retry to check if re-carving is acceptable or not.

In this error detection process, we define the function

$$\begin{aligned}&\mathrm{IsAcceptable}(\mathrm{Sil}_i(t), \mathrm{Svh}_i(t))\\ &\quad = \begin{cases} \textit{true} & \sum_p |\mathrm{Fg}(p) - \mathrm{Bg}(p)| < \text{threshold},\\ \textit{false} & \text{otherwise,}\end{cases}\end{aligned} \tag{1}$$

where p is a pixel such that $p \in \left(\mathrm{Sil}_i(t) \cap \overline{\mathrm{Svh}_i(t)}\right)$. That is, p denotes a pixel which belongs to $\mathrm{Sil}_i(t)$ but not to $\mathrm{Svh}_i(t)$. $\mathrm{Fg}(p)$ and $\mathrm{Bg}(p)$ denote the intensities of captured and background image at p respectively. This function verifies the changes between the projection of visual hull $\mathrm{Svh}_i(t)$ and the original silhouette $\mathrm{Sil}_i(t)$ if it satisfies BC or not.

3.1.3 Iterative algorithm

Using constraints and functions defined above, we introduce the following algorithm which extracts Obj + CS from Img on every viewpoint.

Step 0 Let all the silhouettes on every viewpoint be equal to the entire region of the image. Then we compute the visual hull with these silhouettes and project it onto each viewpoint. We use this projections as $\mathrm{Sil}_i(0), i = 1, \ldots, N$ (Figure 3). Here, we have a set of multi-viewpoint silhouettes which satisfies IC, and start iteration with $t = 1$.

Step t.1 Choose a camera C_j which is not chosen in the previous $N - 1$ iterations.

Step t.2 Let $\mathrm{Sil}_j(t) := \mathrm{Carve}(\mathrm{Sil}_j(t-1), \mathrm{Seg}_j(t-1))$. $\mathrm{Carve}(\cdot)$ produces silhouette which satisfies PC. Figure 4 and 1 illustrates this operation in $t = 1$ and $t \geq 1$. Then, in other viewpoints, use previous silhouettes as is: $\mathrm{Sil}_i(t) := \mathrm{Sil}_i(t-1), i \neq j$.

Step t.3 Compute the visual hull $\mathrm{VH}(t)$ using silhouettes $\mathrm{Sil}_i(t)$ and its projections $\mathrm{Svh}_i(t)$ where $i = 1, \ldots, N$.

Step t.4 If there is no differences between $\mathrm{Svh}_i(t)$ and $\mathrm{Sil}_i(t)$ for the last N iterations, quit the iteration. Here, each $\mathrm{Sil}_i(t)$ satisfies PC, IC, and BC.

Step t.5 Evaluate the differences between $\mathrm{Svh}_i(t)$ and $\mathrm{Sil}_i(t)$ by $\mathrm{IsAcceptable}(\cdot)$, where $i = 1, \ldots, N$. If the number of cameras on which the function returns *true* is greater than a certain threshold, let $\mathrm{Sil}_i(t) := \mathrm{Svh}_i(t), j = 1, \ldots, N$ and go to the next step $t + 1$. Otherwise, re-segment $\mathrm{Seg}_j(t-1)$ and go back to **Step t.2**.

This iterative algorithm can remove Bg regions in theory. We use the resultant Obj + SC as the input of the next algorithm.

3.2 The 3D context-aware shadow removal

Let Scs_j denote the silhouette on C_j given by the algorithm described in the previous section. As described above, Scs_j includes both Obj and CS regions. The goal of the algorithm we introduce in this section is removal of CS from $\mathrm{Scs}_j, j = 1, \ldots, N$.

Suppose we have visual hull Vcs computed from $\mathrm{Scs}_j, j = 1, \ldots, N$. In this computation, we assume that we use a bounding box whose bottom is equal to the floor of the capturing studio as described above. Let the floor plane be $z = 0$. Our algorithm categorizes the surface of Vcs into Obj region, CS region, or part of the floor. In this categorization process, our algorithm needs to explore the surface of Vcs. So we use triangular surface mesh model as the data structure of Vcs for simplicity. Let f denote a triangle of Vcs, F_o the set of triangles categorized as Obj, F_s the set of triangles categorized as CS, and F_f the set of triangles categorized as part of the floor.

3.2.1 Silhouette generation

Once we can categorize triangles into these types, we can obtain silhouette on C_j which contains Obj only as follows:

1. Initialize silhouette and depth buffer of C_j. Let all pixels in the silhouette buffer be background pixel, and all pixels in the depth buffer be ∞.

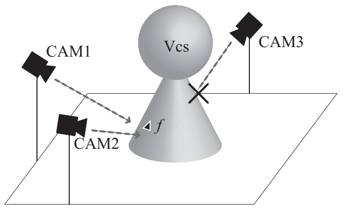
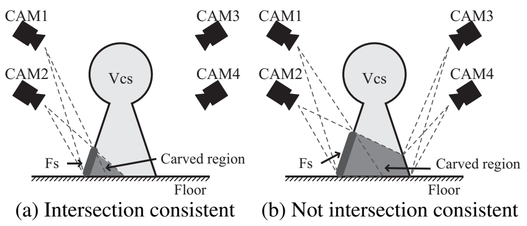


Figure 5: Visibility checking



(a) Intersection consistent (b) Not intersection consistent

Figure 6: Intersection consistent removal of cast-shadow region

2. For each triangle f in Vcs,
   - 2.1 Project f onto C_j, and let us denote the corresponding area on C_j by P.
   - 2.2 If the depth of f is smaller than those of corresponding depth buffer area, update the depth buffer by the depth of f. Otherwise, go to the next triangle.
   - 2.3 If f is in F_o, let the pixels in P be silhouette pixels. Otherwise, let them be background pixels.

3.2.2 The cast-shadow model

We use the following two criteria to identify a triangle f as CS.

Geometric criterion: Cast-shadow region should neighbor the floor plane. That is, one of the neighboring triangles should be categorized as part of the floor. In addition to this, removal of cast-shadow region on Vcs surface should not affect object regions on silhouettes.

Photometric criterion: Cast-shadow region should have similar chromaticity and darker intensity in comparison with that of the background image on *visible* cameras.

Here, *visible* cameras are cameras such that it can observe the triangle in question. For example, visible cameras of f in Figure 5 are CAM_1 and CAM_2 since CAM_3 cannot observe f because of self-occlusion. Note that the above photometric criterion assumes white (non-colored) lighting environment as used in [7].

In this algorithm, we assume that we have knowledge where the object casts its shadows by the camera calibration processed beforehand. We use the floor of the studio, $z = 0$ plane, for simplicity as described above. The geometric criterion states that (1) each triangle in F_s should neighbor another triangle in F_s or F_f, and (2) silhouettes produced by the algorithm described above should be satisfy intersection consistency. Figure 6(a) and (b) illustrate intersection consistent and non-consistent case. In Figure 6(a), if we remove the region of F_s on each camera, the silhouettes on visible cameras CAM_1 and CAM_2 will be carved. This carving on the silhouettes will carve the visual hull Vcs as well, but all the silhouettes satisfies intersection constraint since the carving of Vcs cannot be observed from cameras CAM_3 and CAM_4 which cannot observe F_s. On the other hand, if we remove the region of F_s in Figure 6(b), the change of Vcs caused by the carving of silhouettes on CAM_1 and CAM_2 will be observed from cameras CAM_3 and CAM_4 which cannot observe F_s. That is, the projection of carved Vcs is not consistent with the silhouettes on CAM_3 and CAM_4, and silhouettes cannot satisfy the intersection constraint. We can detect this situation by using the following simple algorithm.

1. Let f be a triangle in question.
2. For each camera c which can observe f, project all triangles in Vcs onto it. Let p denote the pixel on which f is projected. In general, at least two triangles should be projected on p.
3. If only f and triangles in F_f are projected on p, that is, the visual cone from the projection center of c to f crosses only the floor plane, the removal of f does not affect the intersection constraint as shown in Figure 6(a). Otherwise, the removal will affect as shown in Figure 6(b).

We denote this checking by $F_G(f, C)$ where C denotes the set of all cameras $C_1, \ldots, C_N$. It returns true if the removal of f does not affect the intersection constraint, and otherwise it returns false.

The photometric criterion utilizes a pixel-wise shadow classifier which is same as those of [6, 7]. For each camera which can observe a triangle f on it, we check if the pixel corresponding to f is classified as cast-shadow or not. We refer this photometric criterion test by $F_P(v, C)$ defined as follows:

1. Let f be a triangle in question, and the number of cameras n be 0 on which f satisfies the photometric criterion defined above.
2. For each camera c which can observe f, project f onto it. Let us denote the projected area by P and a pixel in P by p. We refer the R, G, B values at p of the captured and background images on c by $F_i(p)$ and $B_i(p)$ where $i = R, G, B$.
   - If all $p \in P$ satisfies following (2) or (3), let $n := n + 1$.

$$0 \leq \sum_i (B_i(p) - F_i(p)) \leq T_1 \qquad (2)$$

$$\begin{cases} T_1 \leq \sum_i (B_i(p) - F_i(p)) \leq T_2 \\ \dfrac{\sum_i F_i(p) B_i(p)}{\sqrt{\sum_i F_i(p)^2 \sum_i B_i(p)^2}} \geq T_3 \end{cases} \qquad (3)$$

   where T_1, T_2, T_3 are certain thresholds. Equation (3) represents the photometric criterion defined above. We also use Equation (2) to let p be shadow region if it is darker up to T_1. This is because chromaticity will be too sensitive than intensity in small values.
3. If f satisfies the photometric criterion on more than one

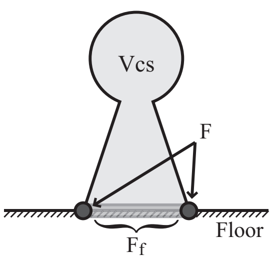


Figure 7: Initial state of shadow suppression algorithm

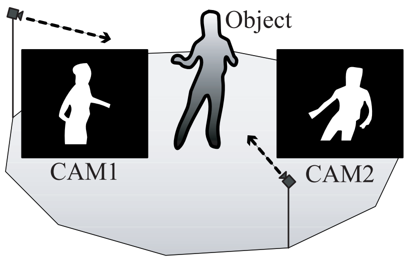


Figure 8: Apparent hole on 2D silhouette

camera, that is, $n > 1$, we categorize f as a part of cast-shadow region and let $F_P(f, C)$ return true. Otherwise, let $F_P(v, C)$ return false.

3.2.3 Algorithm

Using criteria described above, we define the algorithm which removes shadow regions from silhouettes given by the algorithm in Section 3.1. The key point of this algorithm is the order of triangles to be checked if it is a part of shadow regions or not. We traverse the mesh surface from the triangle which is a part of the floor, *i.e.*, $f \in F_f$. Since we assumed that all cast-shadows should be neighbored by the floor regions F_f, we can avoid checking the triangles which is a part of self-shadow or darkly-textured regions not located around the floor.

Step 1 Compute Vcs as polygonal surface model. We used a discrete marching cubes method [10].

Step 2 Let F_f the set of triangles such that each of them is on $z = 0$ plane. Using F_f, we define F as the set of triangles such that each of them is not in F_f and at least one neighboring triangle is in F_f (Figure 7). We also initialize F_o and F_s as empty set.

Step 3 For a triangle $f \in F$,

Step 3.1 If f is classified as cast-shadow, *i.e.*, both $F_G(f, C)$ and $F_P(f, C)$ return true, let $F := F + U - f$ and $F_s := F_s + f$ where U denotes set of triangles such that each of them does not belong to F_o nor F_s.

Step 3.2 Otherwise, let $F_o := F_o + f$ and $F := F - f$.

Step 4 If $F \neq \emptyset$, go back to **Step 3**. Otherwise, project Vcs onto each viewpoint and obtain final silhouettes by the algorithm described in Section 3.2.1.

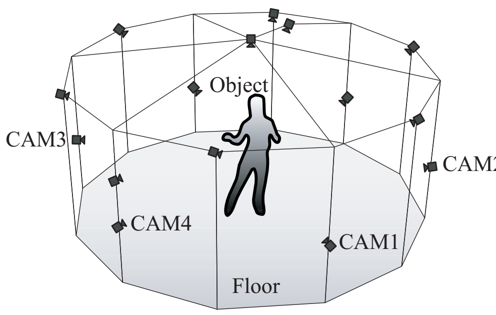


Figure 9: Camera arrangement

3.3 Discussions

Termination of the algorithm: In the proposed algorithm in Section 3.1, carved regions can be recovered due to the error correction scheme as shown in Figure 16. However, it cannot go into infinite loop since we re-segment the recovered regions so that the region is split into smaller segments up to pixel level (Figure 17). This one-way re-segmentation process makes the iteration converge if we omit errors in single pixel level.

Limitation on error detection: It is clear that at least one camera should observe the carved region to detect errors in the algorithm in Section 3.1. If carving of the silhouette on j-th camera at **Step** t**.2** does not affect other silhouettes at **Step** t**.4**, any carving is accepted. In this situation, our algorithm just carves the silhouette based on the current segmentation since we cannot use any information from other viewpoints.

Partial view case: In case that we cannot capture the whole of the object as shown in Figure 10, we cannot use a naive implementation of SFS method as follows – “If we have N cameras, all the portions of the visual hull should be projected onto silhouette region at all of N cameras.” We can use the following definition instead – “All the portions of the visual hull should be projected onto silhouette region at all of *observable* cameras.” Here, we define a camera is observable if the portion of the visual hull is projected inside its imaging window.

Topology of silhouette region: Our algorithm carves silhouettes from outside, and it is clear that $\mathrm{Carve}(\cdot)$ operation cannot carve holes inside the silhouette. That is, if the object has a real 3D hole, our algorithm cannot estimate it since the hole does not appear as a part of out-most contour of the 2D silhouette and the out-most contours satisfy IC. However, we have chance to carve if it is an apparent 2D hole and not observed as holes on other cameras. In Figure 8, the 2D hole under the arm on CAM_1 is not real 3D hole. So if the corresponding region is carved on CAM_2 by $\mathrm{Carve}(\cdot)$ at iteration **Step** t**.2**, the hole can be carved at iteration **Step** t**.5**.

4 Experiments

Figure 9 illustrates our camera arrangement. We use 13 XGA cameras on the wall and 2 XGA cameras on the ceiling. All cameras are calibrated beforehand. Figure 10 and 11 show 4 of 15 input and background images captured by cameras

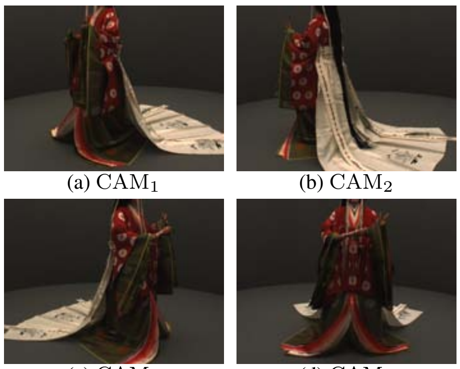

(a) CAM_1 (b) CAM_2 (c) CAM_3 (d) CAM_4

***Figure 10: A lady in Japanese traditional** kimono*

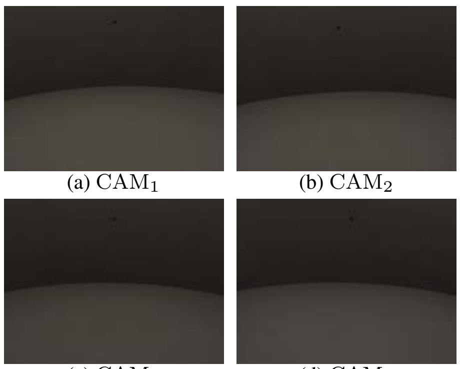

(a) CAM_1 (b) CAM_2 (c) CAM_3 (d) CAM_4

Figure 11: Background images

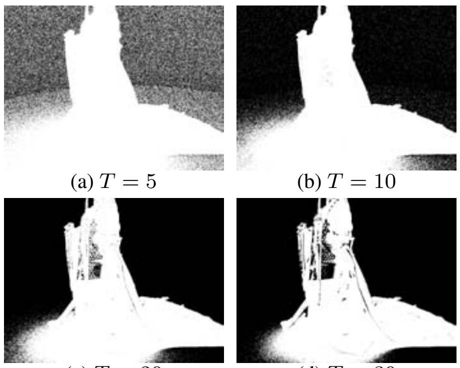

(a) $T = 5$ (b) $T = 10$ (c) $T = 20$ (d) $T = 30$

Figure 12: Naive background subtraction (**CAM_1)***

respectively. We can observe that the object region contains some darkly-textured regions, *e.g.*, long black hair.

Figure 12 shows the results of naive background subtraction in which each pixel p is categorized as silhouette if it satisfies

$$|\mathrm{Fg}(p) - \mathrm{Bg}(p)| >= T, \tag{4}$$

where $\mathrm{Fg}(p)$ denotes the intensity of captured image at pixel p, $\mathrm{Bg}(p)$ that of the background image, T a certain threshold. It is clear that there is no magic threshold which produces accurate silhouette without cast-shadows. On the other hand, Figure 13 shows the results given by pixel-wise shadow suppression algorithm by Horprasert *et al.* [7]. White and gray regions in this figure denote the object and shadow regions respectively. The results are much better than those of the naive approach, but some darkly-textured regions, *e.g.*, long black hair regions in CAM_2, are misclassified as shadow. It is hard to avoid this kind of misclassification. This is because each pixel in both cast-shadow and darkly-textured regions has no difference in pixel-level.

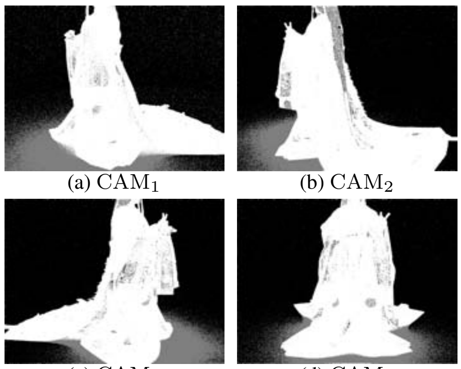

(a) CAM_1 (b) CAM_2 (c) CAM_3 (d) CAM_4

Figure 13: Pixel-wise method [7]

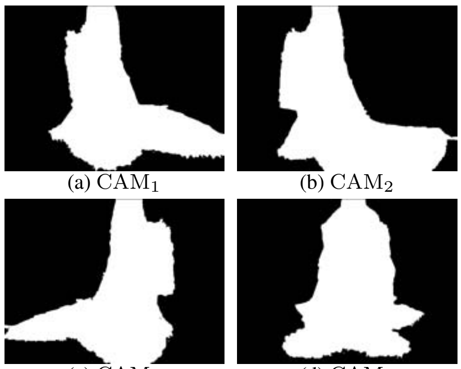

(a) CAM_1 (b) CAM_2 (c) CAM_3 (d) CAM_4

Figure 14: Proposed algorithm

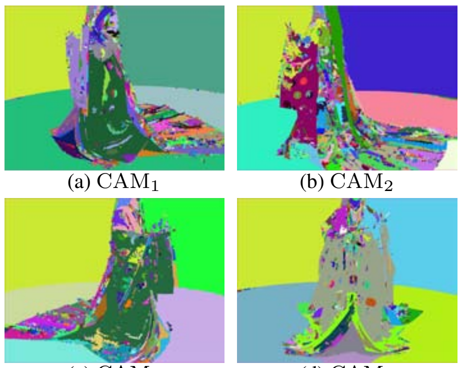

(a) CAM_1 (b) CAM_2 (c) CAM_3 (d) CAM_4

Figure 15: Initial segmentations

Figure 14 shows the final result of our method. The processing cost is approximately three days by Intel Pentium4 3GHz. Compared with Figure 13, we can observe that our algorithm does not mis-classify the regions which are misclassified as shadow regions by the pixel-wise method.

To obtain these silhouettes, we first extract silhouettes including cast-shadow regions by the algorithm described in Section 3.1. Figure 15 shows the initial segmentation of captured images. We can observe that they include some segmentation errors, *i.e.*, some segments include both object and background regions. For example, in CAM_4, the segment

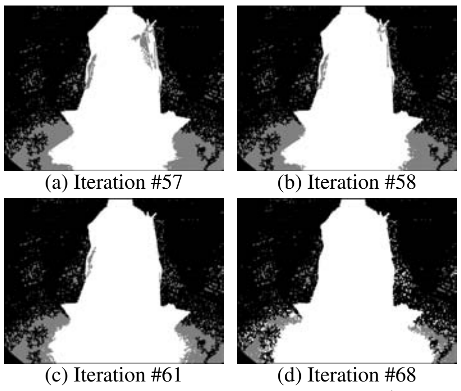


Figure 16: Error correction at CAM_4

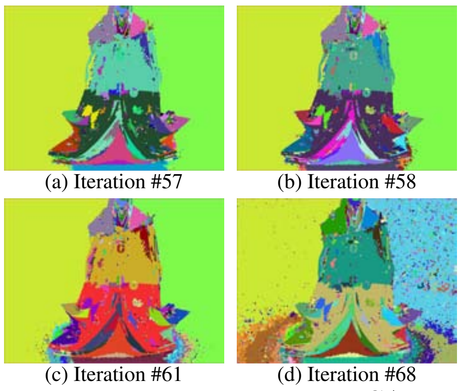


Figure 17: Re-segmentation at CAM_2

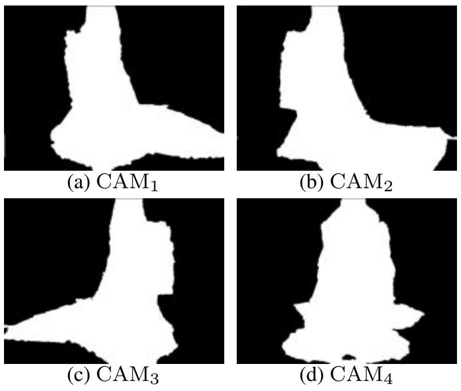


Figure 18: Result of object and cast-shadow extraction

around the left shoulder of the lady includes the background regions as well. This mis-segmentation carves object region as shown in Figure 16(a). Gray areas in this figure denote regions which are carved in the previous iteration, but recovered since the carving of them violates the intersection consistency on other viewpoints. Figure 16 (b), (c), and (d) show how the recovery of mis-carved regions is performed. Figure 16 shows re-segmentation process in this error correction. We can observe that the region which corresponds to the left shoulder of the lady is split into smaller segments. This error detection and correction example indicates that the multi-viewpoint silhouette extraction algorithm by Zeng and Quan [16] will fail because it is straightforward and cannot correct mis-segmentations as shown in Figure 16(a). Figure 18 shows the result of the algorithm described in Section 3.1. Compared with ground truth silhouettes given by hand (Figure

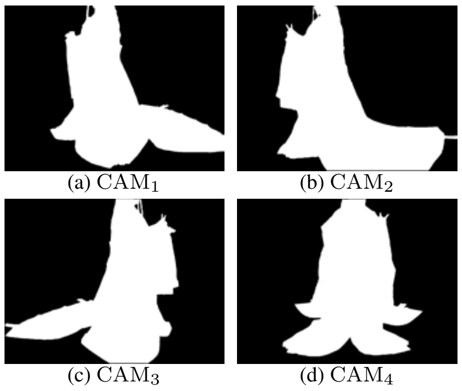


Figure 19: Ground truth

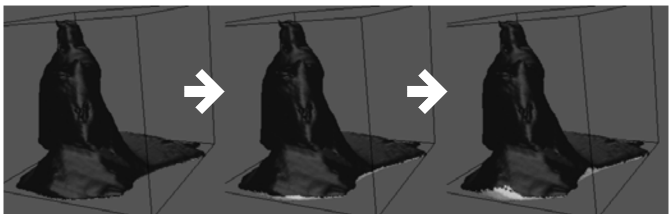


Figure 20: Cast-shadow detection

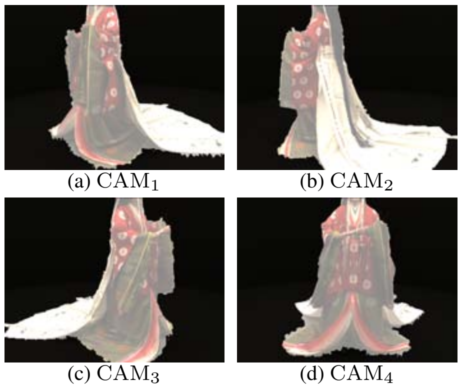


Figure 21: Results of proposed algorithm composed with the input images

19), it is clear that the silhouettes include cast-shadow regions around the object.

Then we use these silhouettes as the input of the algorithm described in Section 3.2. Figure 20 shows how our algorithm detects cast-shadow regions. Here, gray areas denote regions classified as cast-shadow. We can observe that our algorithm starts cast-shadow detection from the triangles on the floor, and proceeds to its neighbors. Finally, we generate silhouettes by projecting the surface of visual hull which is categorized as object region as described in Section 3.2. Figure 21 shows the final result composed with the input images. We can observe that our algorithm can suppress cast-shadows as same as the pixel-wise method shown in Figure 13, and can preserve darkly-textured areas, *e.g.*, long black hair, as object region.

Quantitative evaluation To evaluate our method quantitatively, we categorize the pixels in a silhouette into following four types:

T_1	3	3	3	3	3	3
T_2	150	200	250	150	200	250
T_3	0.990	0.990	0.990	0.995	0.995	0.995
Recall	0.988	0.988	0.988	0.992	0.988	0.988
Precision	0.962	0.962	0.962	0.926	0.961	0.961
F	0.975	0.975	0.975	0.974	0.975	0.975

Table 1: F-measure of our method (Figure 14)

T	5	10	15	20	25	30
F	0.619	0.804	0.855	0.869	0.872	0.869

Table 2: F-measure of naive approach (Figure 12)

True-positive: $S(p) = 1$ and $\hat{S}(p) = 1$,
True-negative: $S(p) = 0$ and $\hat{S}(p) = 0$,
False-positive: $S(p) = 1$ and $\hat{S}(p) = 0$,
False-negative: $S(p) = 0$ and $\hat{S}(p) = 1$,

where $S(p) = 1$ and $S(p) = 0$ denote that a pixel p is estimated as object and background respectively, and $\hat{S}(p) = 1$ and $\hat{S}(p) = 0$ denote that p is a part of object and background respectively in the ground truth image. Here, we use silhouettes given by hand (Figure 19) as the ground truth. Using this classification, we evaluate our method by F-measure defined as follows:

$$\mathrm{Recall} = \frac{N_{\mathrm{TP}}}{N_{\mathrm{TP}}+N_{\mathrm{FN}}}, \quad (5)$$

$$\mathrm{Precision} = \frac{N_{\mathrm{TP}}}{N_{\mathrm{TP}}+N_{\mathrm{FP}}}, \quad (6)$$

$$\mathrm{F-measure} = \frac{2\times\mathrm{Recall}\times\mathrm{Precision}}{\mathrm{Recall}+\mathrm{Precision}}, \quad (7)$$

where N_{TP}, N_{FP}, and N_{FN} denote the number of true-positive, false-positive, and false-negative pixels in the estimated silhouette image respectively.

Table 1 and 2 show F-measures of our method (Figure 14) and those of naive approach (Figure 12) for some thresholds. F-measure of pixel-wise method in Figure 13 is 0.950, and that of ours in Figure 18 which includes cast-shadow regions is 0.969. In Horprasert's result (Figure 13), F-measure, recall, and precision are 0.950, 0.953, and 0.946. In our result which includes cast-shadow regions (Figure 18), F-measure, recall and precision are 0.969, 0.989, and 0.949. These values indicate that our algorithm outperforms not only the naive background subtraction algorithm but also pixel-wise method.

5 Conclusion

In this paper, we proposed a novel algorithm for silhouette extraction from multi-viewpoint images. Our algorithm realizes 3D context-aware error detection, correction, and shadow suppression. The quantitative experiments demonstrate that our algorithm outperforms both naive background subtraction and monocular pixel-wise shadow suppression algorithms. The experiments also show that our algorithm can correct errors which a conventional multi-viewpoint algorithm cannot recover.

Since our algorithm is frame-wise and does not employ temporal-consistency of silhouettes. So our future work will concentrate on (1) extending for videos, and (2) integrating the both 2D multi-viewpoint silhouette estimation and accurate 3D shape reconstruction based on proposed constraints and texture-matching between viewpoints. Development of efficient computation scheme including coarse-to-fine and parallel-processing on GPU is also left for future work since our algorithm is much slower than pixel-wise algorithms due to the iteration.

Acknowledgements

This research was supported by the Ministry of Education, Culture, Sports, Science and Technology of Japan under the Leading Project: "Development of High Fidelity Digitization Software for Large-Scale and Intangible Cultural Assets", and the Ministry of Internal Affairs and Communications of Japan under the SCOPE-C Project 062307002.

References

[1] 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. In *Proc. of CVPR*, pages 375–382, June 2003.

[2] G. Cross and A. Zisserman. Surface reconstruction from multiple views using apparent contours and surface texture. In *Proc. of NATO Advanced Research Workshop on Confluence of Computer Vision and Computer Graphics*, pages 25–47, 2000.

[3] P. Fua and Y. G. Leclerc. Using 3-dimensional meshes to combine image-based and geometry-based constraints. In *Proc. of ECCV*, pages 281–291, 1994.

[4] B. Goldlüecke and M. Magnor. Joint 3d-reconstruction and background separation in multiple views using graph cuts. In *Proc. of CVPR*, pages 683–688, 2003.

[5] J. Y. Guillemaut, A. Hilton, J. Starck, J. Kilner, and O. Grau. A bayesian framework for simultaneous matting and 3d reconstruction. In *Proc. of 3DIM*, pages 167–176, 2007.

[6] H. Han, Z. Wang, J. Liu, Z. Li, B. Li, and Z. Han. Adaptive background modeling with shadow suppression. In *Proc. of Intelligent Transportation Systems*, pages 720–724, 2003.

[7] T. Horprasert, D. Harwood, and L. S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *ICCV Frame-Rate WS*, 1999.

[8] J. Isidoro and S. Sclaroff. Stochastic mesh-based multiview reconstruction. In *Proc. of 3DPVT*, pages 568–577, July 2002.

[9] Y. Ivanov, A. Bobick, and J. Liu. Fast lighting independent background subtraction. *IJCV*, 37(2):199–207, 2000.

[10] Y. Kenmochi, K. Kotani, and A. Imiya. Marching cubes method with connectivity. In *Proc. of ICIP*, pages 361–365, Kobe, Japan, oct 1999.

[11] A. Laurentini. How far 3d shapes can be understood from 2d silhouettes. *PAMI*, 17(2):188–195, 1995.

[12] T. Matsuyama, X. Wu, T. Takai, and S. Nobuhara. Real-time 3d shape reconstruction, dynamic 3d mesh deformation and high fidelity visualization for 3d video. *CVIU*, 96:393–434, December 2004.

[13] S. N. Sinha and M. Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In *Proc. of ICCV*, pages 349–356, 2005.

[14] J. Starck, A. Hilton, and G. Miller. Volumetric stereo with silhouette and feature constraints. In *Proc. of BMVC*, 2006.

[15] G. Vogiatzis, P. H. S. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *Proc. of CVPR*, pages 391–398, 2005.

[16] G. Zeng and L. Quan. Silhouette extraction from multiple images of an unknown background. In *Proc. of ACCV*, pages 628–633, 2004.