Mobile Phone-based 3D Modeling Framework for Instant Interaction *

Wonwoo Lee, Kiyoung Kim, Woontack Woo
GIST U-VR Lab.
500-712, Gwangju, S.Korea
{wlee, kkim, wwoo}@gist.ac.kr

Abstract

In this paper, we present a mobile phone-based 3D modeling framework. The proposed framework enables users to create 3D content and to update the existing ones via their mobile phones. In the proposed system, users take photos of an object by using their mobile phones and give annotations on the image regions of the foreground object and the background through a touch-based user interface. The photos with annotations are transmitted to a remote server via wireless network. On the server machine, a 3D reconstruction process is conducted. Firstly, we recover the poses of cameras from the received photos. Then, the silhouettes of the object are obtained by a multi-view silhouette extraction method exploiting both the color and spatial constraints. Finally, we reconstruct a visual hull that has smooth surfaces through an iterative deformation. The created 3D models are reproduced on mobile phones by merging them on a real scene for mobile augmented reality applications.

1. Introduction

In line with advances on digital imaging technologies, digital cameras have been spread out in the world. As the digital photos become widespread, they have been used as a new source of image-based modeling. Microsoft's Photosynth [10] provides a 3D representation of a scene reconstructed from a set of images. Snavely *et al.* proposed a photo navigation system that provides virtual tours on famous sites by reconstructing them from the photos shared on the Internet [14]. In such a system, the target of reconstruction is a relatively large space shared by the photos and users can browse photos registered to the reconstructed scene.

Considering users' participation to creation of contents and experience on them, the existing services have the following limitations. Firstly, they do not have capability of incremental refinement. Once a scene is reconstructed, it is not possible to update the reconstruction although additional photos of the scene are available. Thus, only one-way interaction is available on the system. Secondly, there is no support for mobile user interaction. Users can experience the reconstructed scene only after they leave the sites and upload their photos to the modeling system. Thus, users cannot enjoy the 3D content instantly at the sites. In addition, the reconstructed scenes are represented as sparse point clouds rather than complete mesh models. Although the 3D point sets give us a perspective on the shape of the scene, it is not sufficient to represent the shape of an object in the scene.

On the other hand, mobile phones have become one of major sources in digital image production as mobile phones equipped with cameras are widely used in our daily life. Recent mobile phones adopt up to 5 megapixel imaging sensors and the quality of photos taken by mobile phones' cameras has been rapidly improved. While digital cameras provides the function of taking photos only, mobile phones give us more capabilities, such as interaction with photos, exchanging photos with other users and showing multimedia contents. These characteristics of mobile phones let us address the limitations we mentioned.

In this paper, we present a mobile phone-based 3D modeling framework that enables users to create and refine 3D content through their mobile phones. The proposed system consists of a mobile phone client and a modeling server. Users take photos of an object by using their mobile phones and give annotations to identify the target object. The photos and annotations are then transmitted to the modeling server via a network. On the modeling server, a 3D model of the object is generated from the transmitted photos. In our system, the modeling process is conducted on the remote modeling server and thus, it is required to make the process automatic. To achieve this goal, the following features are provided.

- **Touch-based interface for user annotation.** We provide a touch-based user interface, through which a user

*This research is supported by Korea Creative Content Agency (KOCCA), Ministry of culture, Sports and Tourism (MCST), under the Culture Technology(CT) Research & Development Program 2009.

2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops
978-1-4244-4441-0/09/$25.00 ©2009 IEEE

can give annotations on photos to indicate image regions that belong to the target object and the background.

- **Silhouette extraction in multi-view images.** For silhouette extraction, we exploit both information from the user annotation and a spatial consistency among multiple views together. Through our approach, the silhouettes of the object in multiple photos are obtained simultaneously without user interaction.

- **Smooth visual hull reconstruction.** We reconstruct a visual hull as a 3D representation of the object for fast reconstruction. Our method generates a visual hull that has smooth surfaces, while keeping the consistency with the silhouettes.

The reconstructed models are reproduced on the mobile phone client on demand. Users can retrieve the 3D content from the server and see them on the sites where they are physically located. To enhance users' experience, we render the models on the mobile phone's screen in the augmented reality manner, i.e., the 3D models are registered on a real-time video stream coming from the mobile phone's built-in camera.

The remainder of this paper is organized as follows. Section 2 surveys the related work to achieve our goal. Section 3 and 4 explain details of the mobile phone client and the modeling servers. Experimental results and discussions are given in Section 5. We conclude this paper with future work in Section 6.

2. Background

To reconstruct an object from images, there are three major steps, camera motion recovery, object silhouette extraction, and 3D model reconstruction. In this section, we review the related works and silhouette consistency theory for multi-view silhouette extraction.

For object extraction from images, many interactive methods have been proposed. Among them, graph cut-based methods are popular. After the frontier work by Boykov and Jolly [2], many variants have been proposed [13, 9]. They show good performance with a single image although they require user interaction. In case of multiple images, spatial coherence is exploited [17, 15]. Recently, both color and silhouette consistency are integrated to solve for the optimal 3D segmentation using the graph cut method [3, 8]. These approaches have an advantage that they don't require user intervention during optimization.

When multiple cameras see the same region, a 3D volume shared by all the views exists. The shared volume can be computed by intersecting all the viewing volumes of the cameras. If an object stays inside the shared volume, the silhouette of the object is consistent in every view. For instance, the projection of a visual hull computed from consistent silhouettes corresponds to the silhouette in each view. We exploit this property as the 3D consistency constraint. However, projecting a visual hull tells us either *true* or *false* about a pixel's occupancy and thus it is not possible to precisely measure the occupancy of a pixel. Moreover, errors in the silhouettes of a view can propagates to the other views since reconstruction of a visual hull is an *AND* operation among volumes created by back-projecting the silhouettes.

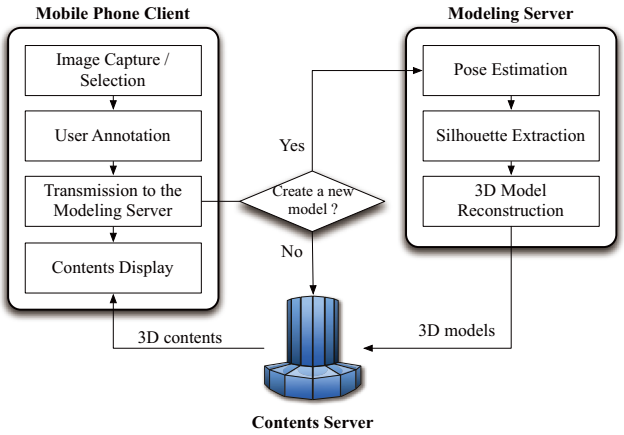


Figure 1: Workflow of the proposed system

In this work, we measure the 3D consistency by using the concept of silhouette calibration ratio proposed in [1]. This computes a pixel's occupancy in an image by counting the number of intersections between the viewing ray through the pixel and the viewing volumes from the other silhouettes. Let us consider a pixel p and its corresponding viewing ray l. We define ω_l^i as an interval along l intersected by the viewing cone of the image i. From the definition of silhouette calibration ratio [1], the occupancy of p can be measured as

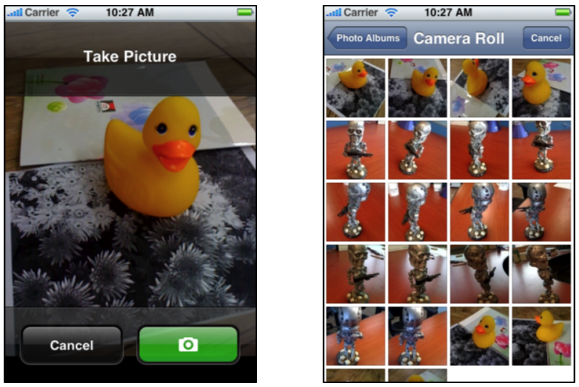
$$R_p = \frac{1}{(N-1)^2} \sum_i \max_{\omega_l^i} \left(\mathcal{N}\left(\omega_l^i\right) \right) \qquad (1)$$

where $\mathcal{N}\left(\omega_l^i\right)$ means the number of images whose viewing cones intersect with the ray interval ω_l^i.

3. Proposed System

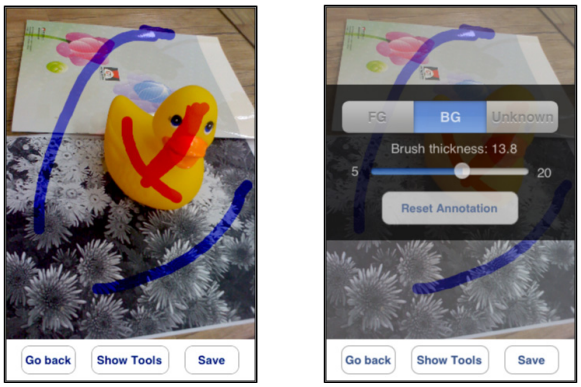
3.1. Overview

The proposed system consists of two components, the mobile phone client and the modeling server as shown in Fig. 1. On the mobile phone client a user takes photos of an object from several different viewpoints and give annotations on the target object and the background. Then, the photos and annotations are transmitted to the modeling server through wireless communication. The modeling



(a) (b)

Figure 2: Image acquisition from an object. (a) Direct photo capture from the built-in camera (b) Photo selection from an existing photo library.



(a) (b)

Figure 3: User annotation on the foreground and background regions. (a) Annotations given on a photo (b) User interface for annotation.

server performs 3D reconstruction process by using both the photos received from mobile phone clients.

The reconstructed 3D models are stored on the content server and shared by users. A user can display and manipulate the reconstructed 3D models on their mobile phones. We implemented an augmented reality application that synthesizes the 3D content with a real scene to provide virtual experiences to users.

The client-server approach of our system has following advantages: 1) it makes the system easy to use since the users who take photos with their mobile phones are not required to have knowledge about 3D modeling process; 2) it is more efficient to perform heavy computations on a remote server since the performance and available resources of current mobile phone platforms are limited.

3.2. Mobile Phone Client

We provide two interfaces for image acquisition from a target object. One is a direct photo capture through built-in cameras of the mobile phones. Users can take photos of the object from several different viewpoints. Another method is to select a photo from an existing photo library on his/her mobile phone. Users can take photo while they are traveling and upload the photos to the server after their journey is finished. We show both interfaces in Fig. 2.

For image-based 3D reconstruction of an object, silhouettes of the object are required. To help the silhouette extraction process, we let users give annotations on the photos to identify the image regions of the foreground and background. We provide a user interface based on touch interaction as shown in Fig. 3. Users can give strokes with their fingertips on the mobile phone's screen. We provide options, such as the thickness of strokes, through a popup menu. After strokes are finished, the annotation data is stored with the photo as a pair.

3.3. Modeling Server

The modeling server performs a 3D modeling process by using the image data sets received from mobile phone clients and previously stored image data sets. The 3D modeling process on the server is performed through several steps. Firstly, the motions of the cameras are reconstructed from the received photos. The poses of cameras are estimated through structure from motion approach and the photos are aligned in a single 3D coordinate system.

Then, a multi-view silhouette extraction is conducted to obtain silhouettes of the target object. In this step, we estimate the silhouettes of the target object from the calibrated images. To robustly perform silhouette extraction with multiple images, we adopt graph cut-based multi-view segmentation approach [8] where both color and 3D consistencies are enforced to progressively update the silhouettes of an object.

Finally, we build a visual hull of the target object. We reconstruct a smooth visual hull from input silhouettes to produce a mesh model with a regular triangulation and smooth surface. We reconstruct a voxel model from a set of silhouette images and extract an isosurface as an initial mesh model. Then, the initial model is iteratively updated based on two constraints (i.e., the silhouette consistency and smoothness), until it is consistent with the input silhouettes.

4. 3D Reconstruction Process

4.1. Camera Structure Recovery

As the first step of 3D reconstruction, we recover the pose of cameras in a 3D coordinate system from the input photos. Firstly, we extract SIFT [6] features from all im-

ages. We adopt GPU-based implementation to improve the speed of feature extraction [16]. Then, we find two images that have highest feature similarity among the input images. From the two selected images, two-view reconstruction is carried out to obtain an initial set of sparse 3D points followed by bundle adjustment. The camera pose is initialized through Nister's 5 points algorithm [12].

After the two-view reconstruction, we incrementally add the images to the reconstructed coordinate system. Since we have SIFT feature in advance, we can arrange the order of the images before we add the image to the reconstructed scene. Basically, we consider the matched numbers and the pixel distance between the images already used in the reconstruction and the rest of image set. We exploit the EPnP [11] algorithm to compute a camera pose from a set of sparse 3D points. The bundle adjustment is implemented using sparse matrices for efficient memory usage. Additionally, we perform a local bundle adjustment with recent images. As a result, the approach returns the camera poses fast with reasonable accuracy.

In pose estimation step, we assume that the intrinsic parameters of the camera are known and only the extrinsic parameters are estimated. To use different cameras that has their own intrinsic parameters, retrieving CCD sensor information from the EXIF tags of images is one of solutions as proposed in [14].

4.2. Silhouette Extraction from Multi-view Images

The silhouettes of an object are iteratively updated through optimization based on graph cut. Before silhouette estimation, we apply mean shift clustering [4] to input photos to perform silhouette extraction based on image regions, not on individual pixels. Initial silhouettes are given by projecting the 3D shared volume onto each image.

Our graph cut segmentation is formulated as follows. In a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, the node set $\mathcal{V}$ consists of the regions of the frame images. Two adjacent regions are connected by an edge $\mathcal{E}_I$. Each node has two additional edges connected to two special nodes, the source S and sink T. Finding min-cut/max-flow of the graph solves the binary labeling problem by minimizing the energy function 2. Here, D is data term that measures the cost for assigning a label x_i to a region r. The label x_i is either 0 for background or 1 for foreground. The link term V computes the cost between two neighboring regions r and s that have different labels. It equals to zero when $x_r = x_s$.

$$E_{total} = \sum_{r} D(x_r) + \sum_{r,s \in Neigh.} V(x_r, x_s) \qquad (2)$$

The Data term gives weight on the edges connected to S $(x_r = 1)$ and T $(x_r = 0)$. The data cost of a region r is defined as a normalized form of the cost D_F and D_B, which

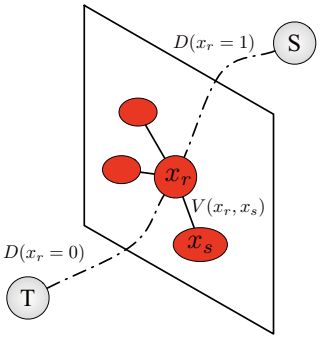


Figure 4: Graph construction on an image.

are the likelihood of r to be classified as the foreground and the background, respectively.

	$r \in F$	$r \in B$	otherwise
$D(x_r = 1)$	0	∞	$\frac{D_F(r)}{D_F(r)+D_B(r)}$
$D(x_r = 0)$	∞	0	$\frac{D_B(r)}{D_F(r)+D_B(r)}$

Basically, $D_F(r)$ and $D_B(r)$ are computed from the color models of the foreground object and the background, $\mathcal{F}_g$ and $\mathcal{B}_g$. They are initialized from the user's annotation and are updated during the iterations. Given a color model Γ, the color likelihood of a region r is computed as the mean of the individual pixel's color likelihood:

$$D(r, \Gamma) = \frac{1}{n_r} \sum_{p \in r} \mathbf{p}(c_p | \Gamma) \qquad (3)$$

where n_r is the number of pixels in the region r.

To more robustly measure $D_F(r)$, we exploit the 3D spatial consistency term $C(r)$ defined as

$$C(r) = \frac{1}{n_r} \sum_{p \in r} e^{-R_p^2/\gamma^2} \qquad (4)$$

where R_p is a silhouette calibration ratio value of the pixel p in r. The term γ controls the penalty allocated to the silhouette calibration ratio of p, since the inferred pixel occupancy can be wrong as presented in [8]. We set $\gamma = 0.8$ in this work.

Thus, $D_F(r)$ is computed as a scaled sum of the color likelihood term and the 3D spatial consistency term.

$$D_F(r) = D(r, \mathcal{F}_g) + \lambda_1 C(r) \qquad (5)$$

The background likelihood is computed in the same way by using the background color model.

$$D_B(r) = D(r, \mathcal{B}_g) \qquad (6)$$

The link term is defined as

$$V(x_r, x_s) = |x_r - x_s| e^{-\beta \|u_r - u_s\|^2}, \qquad (7)$$

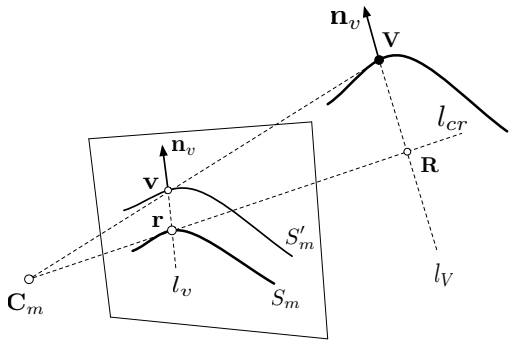


Figure 5: Computation of the silhouette force.

where u_r and u_s represent mean colors of the two regions and $\|\cdot\|$ means L_2 norm. β is defined as $\beta = \left(2\langle\|c_r - c_s\|^2\rangle\right)^{-1}$ as proposed in [13], where $\langle\cdot\rangle$ is an expectation operator.

The silhouette extraction process is iteratively performed until there are no more regions are updated. After the process finishes, we obtain the silhouettes of the target object in all the photos. If there are already existing silhouettes and only a photo that newly comes to the system needs silhouette extraction, the convergence becomes faster since we can get an initial silhouette with better accuracy by projecting the previously reconstructed 3D model of the object.

4.3. Mesh Deformation for Smooth Visual Hull

For a vertex $\mathbf{V}$ of the initial mesh model, two forces $F_s(\mathbf{V})$ and $F_i(\mathbf{V})$ are computed from the two constraints, respectively. The silhouette constraint enforces the mesh model's consistency with the input silhouette images. The silhouette force $F_s(\mathbf{V})$ is defined as the average of all the silhouette forces of the each viewpoint.

$$F_s(\mathbf{V}) = \frac{1}{k}\sum_{m=1}^{k} F_s^m(\mathbf{V}) \tag{8}$$

where k is the number of views that can see $\mathbf{V}$ without occlusion. $F_s^m(\mathbf{V})$ is the silhouette force computed from mth view out of k views.

The force $F_s^m(\mathbf{V})$ is computed as follows. Let the projection of $\mathbf{V}$ be $\mathbf{v}$ and the input silhouette be S_m. When we project the mesh model on the view m, we obtain S'_m, the current silhouette of the mesh model. If $\mathbf{v}$ is inside the boundary of S'_m, $F_s^m(\mathbf{V})$ is set to zero, since moving $\mathbf{V}$ does not contribute to the consistency between S_m and S'_m. Otherwise, in the case of the $\mathbf{V}$ is on the boundary of the S'_m, $F_s^m(\mathbf{V})$ is computed as 9.

$$F_s^m(\mathbf{V}) = dist(\mathbf{R}, \mathbf{V}) \cdot \mathbf{n}_V \tag{9}$$

where $dist(\mathbf{R}, \mathbf{V})$ represents the distance between two vertices $\mathbf{R}$ and $\mathbf{V}$. $\mathbf{R}$ is the destination where $\mathbf{V}$ should move in 3D space to keep the silhouette constraint.

To compute the coordinates of $\mathbf{R}$, we project $\mathbf{V}$ and its normal vector $\mathbf{n}_V$. Then, we have $\mathbf{r}$ which is the intersection of the line l_v and the silhouette S_m. We define $\mathbf{r}$ as the position where $\mathbf{v}$ should move in the 2D image space for silhouette consistency. As shown in Fig. 5, l_V is the line passing through V in its normal direction and l_{cr} is the line passing through the camera center C_m and $\mathbf{r}$. $\mathbf{R}$ is the intersection of l_V and l_{cr}. Since it is possible that two lines do not intersect exactly in 3D space, we compute the least square solution for $\mathbf{R}$

The smoothness constraint is for smooth surface generation after deformation. Since the silhouette force is applied to the vertices on the silhouette boundary, the surface will be jagged if we apply the silhouette force only. The force $F_i(\mathbf{V})$ guarantees that a vertex does not move too far from its neighboring vertices. $F_i(\mathbf{V})$ is defined as

$$F_i(\mathbf{V}) = \frac{1}{n}\sum_{j=1}^{n} (\mathbf{V} - \mathbf{V}_{adj}(j)) \tag{10}$$

where $\mathbf{V}_{adj}(j)$ represents the set of vertices adjacent to $\mathbf{V}$ on the mesh. n is the number of elements in $\mathbf{V}_{adj}$.

The final force $F(\mathbf{V})$ is the scaled sum of the two forces.

$$F(\mathbf{V}) = \alpha F_s(\mathbf{V}) + \beta F_i(\mathbf{V}) \tag{11}$$

$F(\mathbf{V})$ is iteratively updated and applied to the vertex $\mathbf{V}$ to determine its new position $\mathbf{V}_{new}$.

$$\mathbf{V}_{new} = \mathbf{V} + F(\mathbf{V}) \tag{12}$$

The deformation process is conducted until the difference between the silhouettes and the projections of the current 3D model is less than a certain threshold ψ. Typically, the deformation process converges after 20 iterations in our experiments. We set the parameters as $\alpha = 1$, $\beta = 0.3$ and $\psi = 1.5$ in practice.

5. Experimental Results

5.1. System Setup

We implemented the client on a mobile phone platform with ARM CPU working at 412MHz. The resolution of photos captured by the mobile phone's built-in camera is 1600x1200. The modeling was performed on a PC with 2.4GHz CPU and 3GB RAM. For experiment we used 5 real data sets depicted in Fig. 6. We obtain three data sets, *Robot*, *Duck*, and *Lighthouse*, by using our mobile phone client. To examine our modeling framework, we also performed experiments with images from the existing multi-view image data sets (*Dancer* [1] and *Dino* [2]) whose have known camera calibration parameters.

[1] https://charibdis.inrialpes.fr/html/sequences.php
[2] http://vision.middlebury.edu/mview/

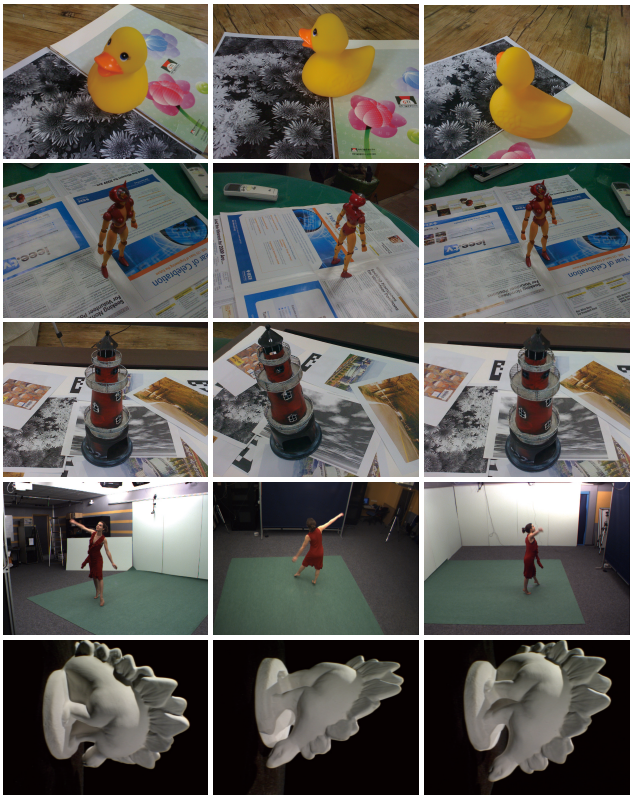


Figure 6: Selected images of data sets used for experiments. From top to bottom, *Robot* (12 views), *Duck* (8 views), *Lighthouse* (8 views), *Dancer* (8 views), and *Dino* (16 views).

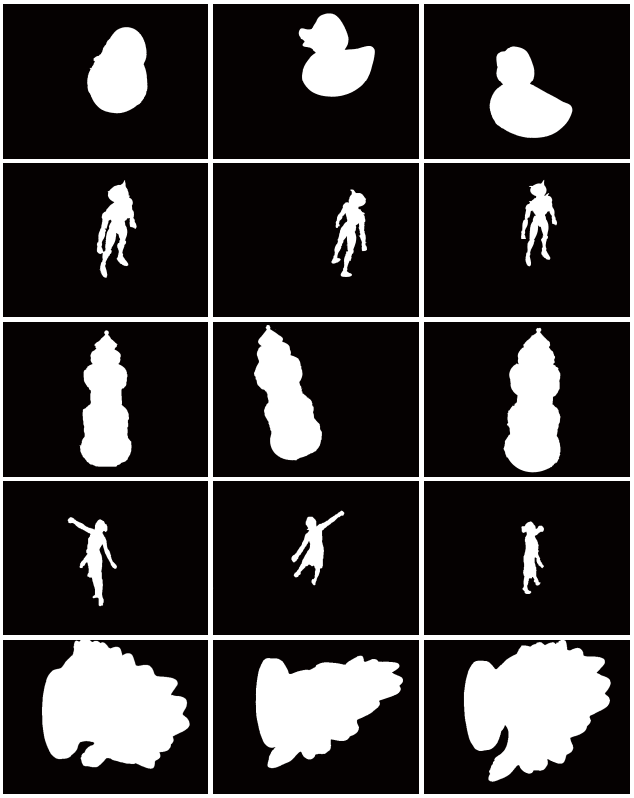


Figure 7: The object silhouettes extracted by our method.

Table 1: Silhouette consistency measurement. Ideally, a set of silhouettes with full consistency has the value of 1.

Data sets	Manual operation	Our method
Duck	0.94	0.95
Robot	0.92	0.91
Lighthouse	0.96	0.89
Dancer	0.95	0.94
Dino	0.91	0.95

5.2. Results

The silhouettes extracted by our method is shown in Fig. 7. As we can see, the silhouettes of each object are extracted in good quality by the proposed approach. To evaluate our silhouette extraction method, we measured the silhouette consistency among all the silhouettes. The silhouette consistency is computed by using silhouette calibration ratio, which should be 1 in ideal case. Silhouette consistencies of the silhouettes obtained by our method and the one obtained by manual hand interaction are computed. As shown in Table 1, the silhouettes obtained by our method resulted in higher or almost equal consistency compared to the silhouettes obtained by manual operations. In case of the *Lighthouse* data sets, the target object has colors similar to the background, thus the quality of extracted silhouette is decreased. Thanks to the multi-view consistency constraint, our method extracted silhouettes consistent among the views.

In Fig. 8, the reconstructed visual hull models are depicted. The first two columns show the exact visual hulls reconstruction with the method proposed in [5] and the last one shows the visual hull obtained by our method. Our smooth visual hull models are smoother and keep the shape of the model better. The reconstructed 3D model should be consistent with the input silhouette images. We measured how the mean silhouette errors between the 3D model and the input silhouettes change during the iterative deformation. The mean silhouette error is computed as the average distance between the contour of the original silhouette and the contours of the reconstructed mesh model. As we can see in Fig. 9, the difference between the original silhouettes and the mesh model decreases in each iteration. After the iterations are terminated, the silhouette errors are reduced to about 1 pixel. Thus, the shape features of the exact visual hull are preserved in our reconstruction while maintaining smooth surfaces.

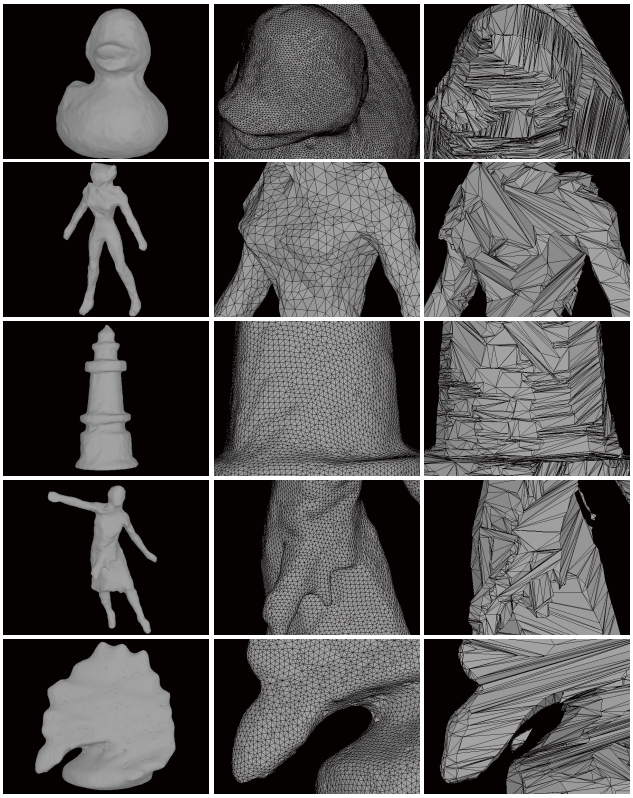


Figure 8: Visual hull reconstructed from silhouettes. From the left, smooth visual hulls, magnified views of smooth visual hulls, and magnified views of the exact visual hull [5].

5.3. Interactive 3D Content Display on Mobile Phones

Users can see the created content on their mobile phones and interact with them. The contents are rotated or scaled based on users' touch interaction. We provide two options for displaying the contents on mobile phones. One is displaying 3D models in a conventional way. The contents are rendered at the center of the screen and users interact with them. Another is displaying the contents in the manner of augmented reality. The 3D models are registered to a video captured by the built-in camera in real-time. By merging virtual contents with a real scene, the user's experience can be enhanced better. Fig. 10 depicts two different interactive display approaches.

6. Discussions and Conclusions

In this paper, we presented a mobile-phone based 3D modeling framework for taking photos, giving annotations, recovering 3D model from the photos, and displaying the reconstructed 3D models. We evaluated the proposed system from the real data sets and our system shows good performance as in the experimental results.

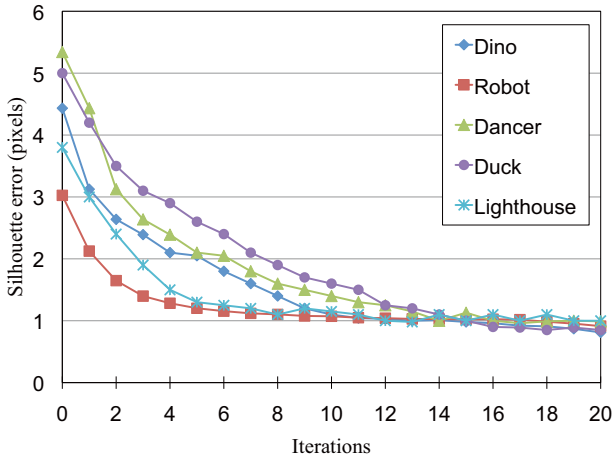


Figure 9: Silhouette errors in iterative deformation.

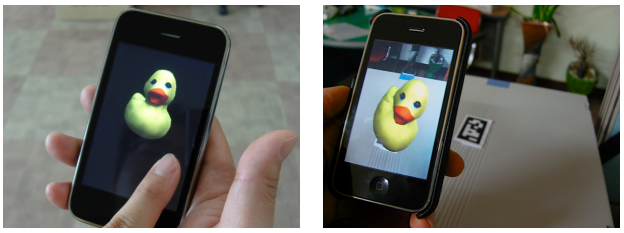
In our current implementation, several limitations still exist. Firstly, we did not implement the capability of finding a matched object from the existing database when a new image comes to the server. As the number of objects increases identifying a matching data sets become problematic. A possible solution to this classification problem is using not only image features, but also other information, such as the location of a user and the time a photo is taken. Secondly, the current visual hull representation is obviously not enough for high-quality 3D content. The 3D reconstruction of the object can be updated with multi-view stereo methods. If the reconstruction process takes too long time, an option is conducting model update on the server in offline, while providing fast feedback to users with visual hulls. Thirdly, the created mesh models need to be optimized for mobile phone platforms to provide 3D content online. Since mobile phones are not able to handle heavy mesh models due to their limited resources, an optimized version of mesh models are required. Mesh simplification or progressive mesh representation [7] can be used for this purpose.

In the future, we will focus on addressing the limitations of current implementation, e.g., on finding a set of images matched to a query image. For this, a data format including different information, (e.g., locations, image features, 3D structures, etc.) and a fusion method will be required.

References

[1] E. Boyer. On Using Silhouettes for Camera Calibration. In *7th Asian Conference on Computer Vision*, pages 1–10. Springer-Verlag, January 2006.

[2] Y. Boykov and M.-P. Jolly. Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images. In *IEEE International Conference on Computer Vision*, volume 1, pages 105–112, 2001.



(a) (b)

Figure 10: Interactive 3D content display on the mobile phone client by (a) displaying them on the mobile phone's screen and (b) synthesizing them on a real environment.

[3] N. Campbell, G. Vogiatzis, C. Hernández, and R. Cipolla. Automatic 3D Object Segmentation in Multiple Views using Volumetric Graph-Cuts. In *British Machine Vision Conference*, volume 1, pages 530–539, 2007.

[4] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[5] J.-S. Franco and E. Boyer. Exact polyhedral visual hulls. In *14th Fourteenth British Machine Vision Conference*, pages 329–338, 2003.

[6] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[7] H. Hoppe. Progressive meshes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108, New York, NY, USA, 1996. ACM.

[8] W. Lee, W. Woo, and E. Boyer. Identifying foreground from multiple images. In *7th Asian Conference on Computer Vision*, pages 580–589, December 2007.

[9] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy Snapping. In *ACM SIGGRAPH*, volume 23, pages 303–308, 2004.

[10] Microsoft. Photosynth. http://photosynth.net/, 2009.

[11] F. Moreno-Noguer, V. Lepetit, and P. Fua. Accurate non-iterative o(n) solution to the pnp problem. In *IEEE International Conference on Computer Vision*, Rio de Janeiro, Brazil, October 2007.

[12] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–777, June 2004.

[13] C. Rother, V. Kolmogorov, and A. Blake. GrabCut-Interactive Goreground Extraction using Iterated Graph Cuts. In *ACM SIGGRAPH*, volume 24, pages 309–314, 2004.

[14] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from Internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, November 2008.

[15] M. Sormann, C. Zach, and K. Karner. Graph cut based multiple view segmentation for 3d reconstruction. In *The 3rd International Symposium on 3D Data Processing, Visualization and Transmission*, 2006.

[16] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). http://cs.unc.edu/ ccwu/siftgpu, 2007.

[17] G. Zeng and L. Quan. Silhouette Extraction from Multiple Images of An Unknown Background. In *6th Asian Conference on Computer Vision*, volume 2, pages 628–633, 2004.