3D Reconstruction with Automatic Foreground Segmentation from Multi-View images Acquired from a Mobile Device

Ping-Cheng Kuo, Chao-An Chen, Hsing-Chun Chang,
Te-Feng Su, Shang-Hong Lai

Department of Computer Science
National Tsing Hua University, Hsinchu, Taiwan
http://cv.cs.nthu.edu.tw/index.php

Abstract. We propose a novel foreground object segmentation algorithm for a silhouette-based 3D reconstruction system. Our system requires several multi-view images as input to reconstruct a complete 3D model. The proposed foreground segmentation algorithm is based on graph-cut optimization with the energy function developed for planar background assumption. We parallelize parts of our program with GPU programming. The 3D reconstruction system consists of camera calibration, foreground segmentation, visual hull reconstruction, surface reconstruction, and texture mapping. The proposed 3D reconstruction process is accelerated with GPU implementation. In the experimental result, we demonstrate the improved accuracy by using the proposed segmentation method and show the reconstructed 3D models computed from several image sets.

1 Introduction

Three-dimensional reconstruction from multi-view images is a challenging problem in computer vision. In general, most previous 3D reconstruction systems are based on point correspondences between multi-view images [1–3]. In this paper, we develop a 3D object reconstruction system based on visual hull, which does not require any point correspondences on the object. In this work, we use a handheld camera to capture images surrounding the target object freely from different views. Compared to other handheld devices (e.g. Kinect[4]), our system has the advantages of low cost and ease of use. An example of input images are shown in Figure 1, and the flow chart of our system is depicted in Figure 2.

Inspired by GraphCut algorithm, many papers apply GrphCut algorithm into image segmentation such as Boykov and Jolly's work [5], Pham et al. [6], and Rother et al. [7]. The image segmentation method proposed in this paper is aspired by [6, 7]. We assume that the target object be placed roughly in the middle of each image. So firstly, we define a fixed bounding box in the center region to reduce user interaction. Then we compute the homography between the pattern image and input image to obtain an initial guess of the bounding

box. Finally we use GraphCut optimization to label the region in the bounding box.

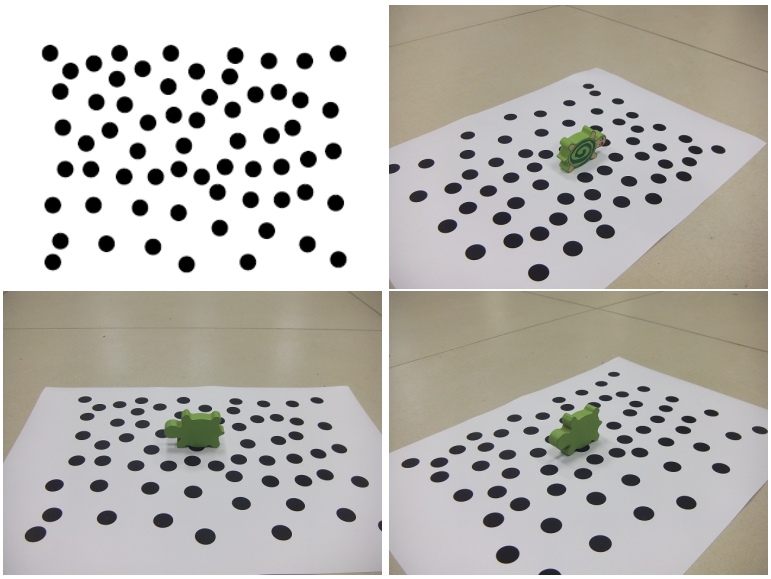


Fig. 1. The left top image is the pattern image. The others are input images to our system. The target object is placed in the middle of each image. These images are taken from various viewpoints around the target object.

The segmentation methods in [6, 7] can be applied for images of natural environment because pixels with color similar to those of the reference background will be labeled as foreground as long as there is a little difference between them. However, this property may not be appropriate for our dataset. With illumination change and different shadow of different view point, the background in the bounding box may contain some difference to the reference background outside the box. In this work, we propose to combine the color distribution constraint [6] and the multi-view homography analysis (MHA) to solve this problem. We speed up the entire system toward segmentation and image-based visual hulls via GPU programming. For segmentation, we use GraphCut to label each pixel with foreground or background. We take each working item to roughly compute each pixels initial label for GraphCut. In image-based visual hull, each working item converts the 2D pixels to 3D points and find local neighbors for each 3D point to compute its normal vector. The 3D reconstruction process is quite time consuming, and it can be sped-up by taking advantage of parallel computing. The remainder of this paper is organized as follows. Section 2 introduces our 3D reconstruction system. In section 3, we are going to describe our proposed segmentation method in this paper. And we describe the GPU acceleration of

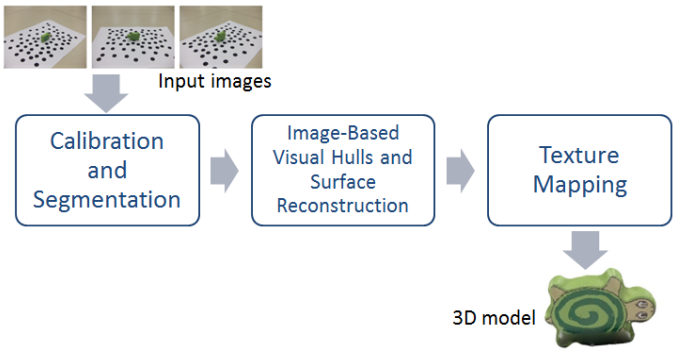


Fig. 2. The flow chart of our system. Our system only needs some pictures as input images to reconstruct the 3D model. It consists of camera calibration, foreground segmentation, image-based visual hulls, surface reconstruction, and texture mapping.

the proposed 3D reconstruction system in section 4. The experimental results are given in section 5, followed by conclusion in section 6.

2 Proposed 3D reconstruction system

2.1 Foreground Segmentation

In the beginning of our 3D reconstruction system, we need to extract the foreground region (the object we would like to reconstruct) from the background which has a preset dot pattern. The dot pattern is used for camera calibration in step 2.2. One of the challenges in foreground segmentation step is to remove the background which has dot pattern. Most segmentation methods lead to unfavorable segmentation results in this case, since they tend to be regard the black dot as the foreground. As a result, we propose a novel approach for image segmentation, which can produce a better result by solving the foreground segmentation for multiple images of the scene simultaneously in a graphcut framework. In this paper, we focus on the foreground segmentation algorithm, which will be described in details in section 3.

2.2 Camera Calibration

Our calibration utilizes a dot pattern to compute the associated camera projection matrix [8]. It is based on using a multi-view stereo algorithm to calibrate an image sequence, which is a dot pattern on the planar in calibration. We print the dot pattern paper provided from the website, and then place the object on the top of the dot pattern paper. The pattern is designed for detecting the dots from a different view and then finding the corresponding dots in each different view. We can take images around the object on the pattern to obtain camera

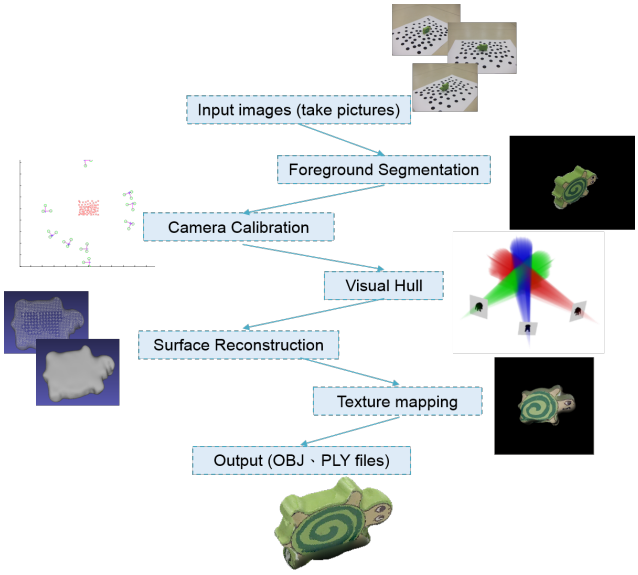


Fig. 3. The flow chart of our 3d resconstruction system.

intrinsic and extrinsic parameters. Once both camera parameters are obtained, we can compute the camera projection matrix. Thus the homography matrices can be estimated by the corresponding dots from every two images.

2.3 Visual Hull

In this step, we will obtain a point cloud of 3D object and the normal vectors for all data points by applying the image-based visual hulls (IBVH) [9] after we compute the camera projection matrix and a set of silhouettes of foreground images. The IBVH algorithm [9] is an efficient image-based approach to computing and shading visual hulls from silhouette image data. In our system, we use the IBVH algorithm [9] to obtain the point cloud of 3D object, and we perform surface reconstruction from the point cloud in step 2.4.

2.4 Surface Reconstruction

The mesh of a 3D model is reconstructed in this step. After obtaining a point cloud for a 3D object in step 2.3, we apply the Poisson surface reconstruction [10] to reconstruct the mesh of a 3D object. Poisson surface reconstruction formulates surface reconstruction as the solution to a Poisson equation, and creates watertight surfaces from oriented point sets finally.

2.5 Texture Mapping

The last step is texture mapping. The texture mapping technique used in this work is multiresolution spline [11]. This method combines two or more images into a larger image mosaic, and we map this image mosaic to the mesh generated in step 2.4.

3 Proposed Foreground Segmentation Method

We define a fixed bounding box (see Fig. 4(a)) and users are requested to place the target object in the middle of images. Inspired by [6], we state this problem as an energy minimization problem (see Fig. 4(b)). Then we add a multi-view homography constraint term into the energy function. Unlike the iterated distribution matching in [6], we use homography relationship between the input image and pattern image to compute a good initial guess. Hence we can obtain the segmentation result efficiently by using GraphCut to minimize the energy function.

3.1 The Color Distribution Method (CDM)

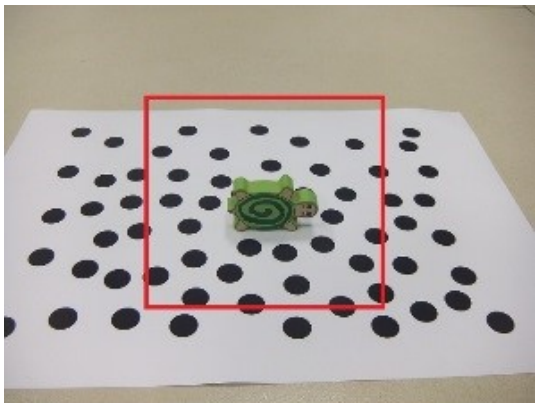
According to the three conditions and the method in [6], we implement their energy function and find that their method is sensitive to the small color distance. Even if there only exists small color difference between the pixel color values in the bounding box and the reference background, their method may determine the pixel as foreground (see Fig. 5), which is not appropriate. With the illumination change and different shadow due to different viewpoint, the background in the bounding box may exist some difference to the reference background outside the box (see Fig. 6).

Inspired by Pham et al.'s work [6], we state the problem as the minimization of an energy function, and apply the following function as the energy function.

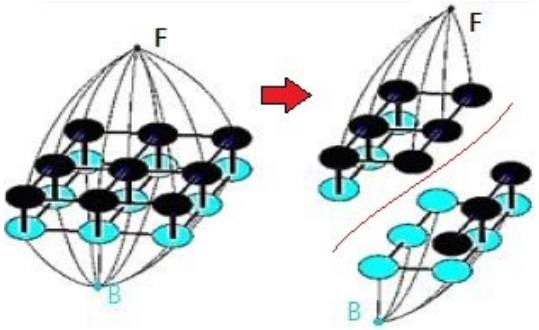
$$E_1(L) = \underbrace{B(P_1(L), H)}_{F_1(L)} - \underbrace{B(P_0(L), H)}_{F_2(L)} + \lambda S(L) \tag{1}$$

There exists a segmentation L that separates the image region into background $R_0^L = \{p|L_p = 0\}$ and $R_1^L = \{p|L_p = 1\}$. Where $P_i(L)$ is the distribution of the region R_i^L and $B(f, g)$ is the Bhattacharyya distance which expresses the similarity of the two distributions. The terms $F_1(L)$ and $F_2(L)$, which evaluate global similarities between distributions, cannot be express purely in the form of data and smoothness terms. It follows that cannot be solved directly by GraphCut. Therefore, Pham et al.[6] estimated an upper bound of $F_1 + F_2$ and it can be expressed purely by data terms. $S(L)$ is a smoothness term that enforces smoothness property to L.

$$S(L) = \sum_{(p,q)\epsilon N} \delta_{L_p \neq L_q} \left(\frac{1}{1 + \|I_p - I_q\|^2} + \frac{\epsilon}{\|p - q\|} \right) \tag{2}$$



(a)



(b)

Fig. 4. In (a), the red frame is the fixed bounding box (i.e., each of our image has the same bounding box). (b) shows the energy minimization problem. Each pixel is a node and the sum of all edge weights is the total energy. We can minimize the energy by using GraphCut algorithm.

3.2 Multi-view Homography Analysis (MHA)

To decrease the error resulting from shadow or illumination changes, we apply multi-view homography analysis with the homography matrix, H, obtained from calibration to integrate more reliable information into the data term of the energy function. Let F^i be the i-th frame, N be the number of neighbor frames which are included into the multi-view analysis for F^i. In our experiment, we set N to 6. Let F^n be a neighbor frame of F^i , and the set of all neighboring frames are denoted by :

$$F^{i,N} = \left\{ F^n \mid i - \frac{N}{2} < n < i + \frac{N}{2}; n \neq i \right\} \tag{3}$$

If n is less than 0, we take neighbor frames from the tail frame. For each pixel p in F^i, there is a likelihood, W_p^i, which ranges from 0 to 1.

$$W_{p^i}^i = \lambda V_{p^i}^{i,C} + (1-\lambda) V_{p^i}^{i,G} \tag{4}$$

The higher value $W_{p^i}^i$ is, the more probable it is a foreground pixel. Otherwise, the pixel is more probable to be background. $W_{p^i}^i$ is determined by the color distortion $V_{p^i}^{i,C}$ and gain information $V_{p^i}^{i,G}$. λ is used to control the importance of these two terms. $V_{p^i}^{i,C}$ is the color distortion of p on the i-th frame, which is related with the angle between two color vectors.

$$V_{p^i}^{i,C} = 1 - exp\left(- \left(P_{p^i}^{i,C} \right)^2 * \alpha \right) \tag{5}$$

$$P_{p^i}^{i,C} = \frac{\sum_{n \epsilon N} f_{cd}\left(R_{p^i}^i \, , \, R_{p^{i+n}}^{i+n} \right)}{N} \tag{6}$$

$$f_{cd}\left(u, v\right) = \cos^{-1}\left(\frac{u \cdot v}{|u|\,|v|} \right) \tag{7}$$

Here, R_b^a means the color vector $[r, g, b]$ of pixel b in the a-th frame. p^{i+n} means the corresponding pixel of p^i on the neighbor frame via homography matrix, H_{i+n}

$$P^{i+n} = H_{i+n} * p^i \tag{8}$$

If $R_{p^i}^i$ and $R_{p^{i+n}}^{i+n}$ are similar (e.g. dark green and light green), the angle will be smaller, hence $V_{p^i}^{i,C}$ will be more capable to overcome the problem of illumination variation. $V_{p^i}^{i,G}$ is the gain information of p^i on the i-th frame, which is related with illumination as follows:

$$V_{p^i}^{i,G} = 1 - exp\left(-\left(P_{p^i}^{i,G}\right)^2 * \beta\right) \tag{9}$$

$$P_{p^i}^{i,G} = \frac{\sum_{n\epsilon N} f_{gain}\left(I_{p^i}^{i},\ I_{p^{i+n}}^{i+n}\right)}{N} \tag{10}$$

$$f_{gain}\left(I_a, I_b\right) = \frac{|I_a - I_b|}{I_b} \tag{11}$$

Here, I_b^a means the intensity of pixel b in the a-th frame. The gain value will be high when the brightness difference between two pixels is large. Since the gain value for background shadow is still relatively small with foreground, it can be used to overcome the shadow problem. $V_{p^i}^{i,C}$ and $V_{p^i}^{i,G}$ are the color distortion and gain information in F^i. Exponential function is used to convert the range to $(0, 1]$. To expand the difference between the foreground and background values, we take the square of $P_{p^i}^{i,C}$ and then multiply it by a constant α. Similarly, we take the square of $P_{p^i}^{i,G}$ and multiply it by a constant β.

3.3 Initial Guess and Energy Minimization

We add the MHA information into the data term of the energy function referred to [6] and it can be denoted as follows:

$$\begin{aligned} &Q\left(L, L^*, \alpha\right) \\ &\quad = \sum_{p\epsilon R_1^L} m_p\left(1\right) + W_p + \left(sign\left(F\left(L^*\right)\right)\alpha + 1\right) \sum_{p\epsilon R_0^L} m_p\left(0\right) \end{aligned} \tag{12}$$

where $m_p\left(1\right)$ and $m_p\left(0\right)$ are data terms given for each p:

$$\begin{aligned} m_p\left(1\right) &= \frac{\delta_{L_p^*=0}}{2A\left(R_1^{L^*}\right)} \sum_{z\epsilon Z} K_z\left(I_p\right) \sqrt{\frac{H\left(z\right)}{P_1^{L^*}\left(z\right)}} \\ &\quad + \frac{\delta_{L_p^*=0}}{A\left(R_0^{L^*}\right)} \left(\sum_{z\epsilon Z} K_z\left(I_p\right) \sqrt{\frac{H\left(z\right)}{P_1^{L^*}\left(z\right)}} + F\left(L^*\right)\right) \\ m_p\left(0\right) &= \frac{F\left(L^*\right)}{A\left(R_0^{L^*}\right)} \end{aligned} \tag{13}$$

Here, δ is the Kronecker delta function. When $R_1^{L^*} = \emptyset$(empty), we consider $A\left(R_1^{L^*}\right)$ and $P_1^{L^*}(z)$ as 1.

We use homography projection to warp each image onto pattern image and use the color distance to obtain an initial guess. Unlike the iterated distribution matching in [6], we use homography relationship between the input image and pattern image to compute a better initial guess, hence we can obtain the segmentation result quickly by using GraphCut.

Fig. 5. The results of [6]. This method can discriminate small color difference between foreground and background. (image source: [6])

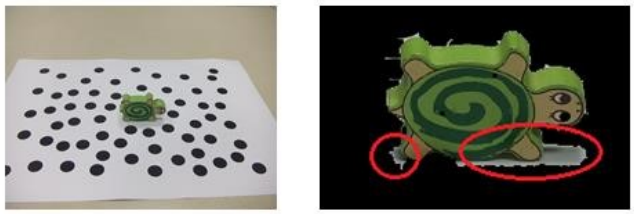


Fig. 6. The method in [6] is too sensitive to consider the shadow region to be foreground.

3.4 Model Reconstruction

When we have camera projection matrix and a set of silhouettes of foreground images, we can obtain a point cloud for the 3D object by applying the image-based visual hulls (IBVH) [9], which recovers the space of 3D viewing cones by intersecting them from different views decided by the known camera information. We set a virtual desired view that is above the model on the pattern in our system, and the remaining images are reference views. There are dense rays from the desired view that intersect with other reference views rays, and then project onto each reference view to get the set of intersecting lines. Finally, we project the intersecting lines on each reference view to the 3D space to get the model crave by the silhouette of foreground object. The algorithm can be speded up by using epipolar constraint [9]. Thus, we can obtain a set of vertices in 3D coordinates. The vertices are point cloud of the 3D model which is segmented by silhouette of reference views. We then compute the normal of each 3D vertex by finding the neighbor vertices. It is based on using the viewing direction of the ray, the segmented information provided by the reference image and the neighbors of the ray on the desired view. While we have point cloud and the normal vectors for all data points, we can apply Poisson surface reconstruction technique [10] and texture mapping algorithm to create the reconstructed 3D model.

4 GPU Acceleration

In the 3D reconstruction system, we speed up the foreground segmentation and image-based visual hulls components via parallelization with GPU programming. We are going to illustrate each part in details in the following.

4.1 Speed up Foreground Segmentation

In foreground segmentation, in order to determine each node's label, we apply GraphCut to solve this problem. We assume each node is a pixel that limit to two possible labels: foreground pixel, background pixel. It is necessary to give an initial label as input to the GraphCut procedure.

We take each GPU working item to compute each image view's color distribution and each pixel's color distance. Each GPU working item represents a pixel in the whole image. We set the homography matrix as global parameters and every GPU working item can compute the color distance from pattern view to other view through the homography matrix. Once obtaining each pixel's color distance, we can roughly decide the initial label from the distance.

4.2 Speed up Image-Based Visual Hull

In the image-based visual hulls, we parallelize two parts in our 3D reconstruction system to speed up the computation. For every reference view and the desired view on the ray intersection, we can obtain the visual hull vertices, which come from the 2D image on a pixel required by the known camera projection information, and they are converted to 3D spatial coordinates. Thus, we first assign each pixel to GPU working item to calculate their 3D coordinates by known projection matrix.

Second, we parallelize the procedure of finding each voxel's neighbors for calculating a voxel's normal, which is required in the Poisson surface reconstruction. Against all of the 3D points we look for the neighbor-ray, and for every ray we select a closest point. Once we obtain a certain number of neighbors of a 3D point, the normal vector of 3D point can be determined by the set of neighboring points.

The GPU acceleration of the above components in our 3D reconstruction system can speed up the computational performance and the result will be shown in section 5.3.

5 Expermemtal Results

In this section, we demonstrate the performance of the proposed segmentation method and compare the segmentation results with the GrabCut algorithm [7] implemented by OpenCV. Besides, we show our reconstruction system by using three image datasets. There are 14 images of resolution 608x456 in each dataset. We evaluate the results of segmentation and 3D model reconstruction.

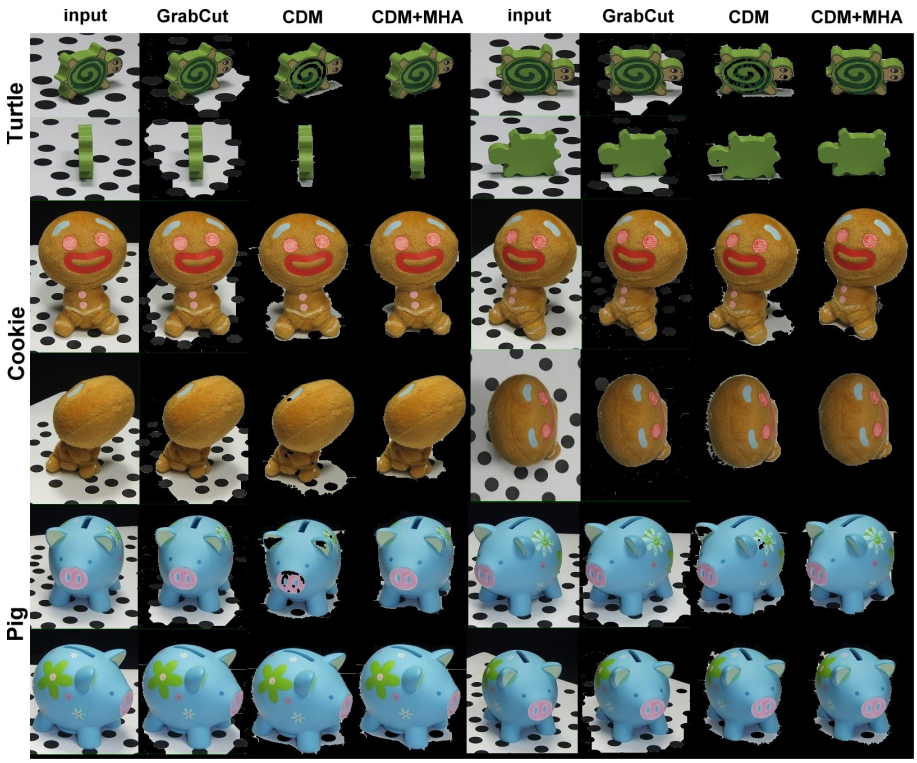


Fig. 7. Comparison of segmentation results. Each row shows results from a different dataset. The columns show (1,5) input image, (2,6) the results of OpenCV GrabCut, (3,7) the results of CDM, and (4,8) the results of CDM with MHA.

5.1 Evaluation of Segmentation

We manually label the ground truth and employ error rate as a measurement to evaluate the results

$$errorrate = \frac{No.of\ misclassified\ pixels}{No.pixels\ in\ inference\ region}$$

Figure 7 shows the segmentation results of our proposed method and prior work on several datasets. For each dataset, we report the average error rate in Table 1. According to Table 1, the experimental results show that the average error rate of GrabCut [7] is about 10%, and the average error rate of the Color Distribution Method (CDM) [6] is much less than the error rate of GrabCut [7]. Our proposed method outperforms the CDM [6] since we include the Multi-view Homography Analysis (MHA) to improve better results. Furthermore, the average error rate of our proposed method is less than 2% in average from the three datasets, and these errors has quite small influence to the 3D reconstruction results.

Table 1. Comparison of the average error rate

Dataset	GrabCut [7] (%)	CDM [6] (%)	CDM with MHA (%)
Turtle	10.99	2.99	0.31
Cookie	9.54	4.31	1.22
Pig	10.73	4.10	2.95

5.2 Evaluation of Reconstructed 3D Model

We use the coverage rate to evaluate 3D models:

$$Coverage\,rate = \frac{P_{GT} \cap P_i}{P_{GT}}$$

where P_i is the plane pixels projected by 3D model reconstructed from segmentation results of i-th method. The coverage rate is the number of pixels overlapped by P_{GT} and P_i divided by the number of all pixels on P_{GT}. The results are reported in Table 2. Our 3D reconstruction results are depicted in Figure 8.

Table 2. Comparison of the coverage rates

Method	Coverage rate (%)
GrabCut [7]	84.99
CDM [6]	88.17
CDM with MHA	93.50

5.3 Performance by GPU Acceleration

We have developed a set of parallel algorithms to speed up the original sequential version of the 3D reconstruction system, which includes foreground segmentation and image-based visual hulls. In GPU version, each part is speded up by about 4-7 times. The sequential version of the 3D reconstruction process is time consuming, taking about 40 to 60 seconds, and for the parallel version it takes only 20 to 40 seconds. The GPU acceleration was implemented on AMD Radeon ^{TM} HD 7850 Graphics with OpenCL, and the CPU version was implemented on a Intel i5-3570 CPU with 8GB RAM. The results are shown in Table 3.

In addition, we also run our system on InFocus M320 mobile phone. Its CPU version uses MediaTek MT6592, 1.7GHz, 8 core, and the GPU version uses Mali-450. As fig. 9, we port our system to Android 4.2.2 to evaluate its performance. The kernel which includes segmentation and image-based visual hulls part in the sequential version takes about 153.3481 seconds for a 3D reconstruction task,

Fig. 8. Our results of 3D model

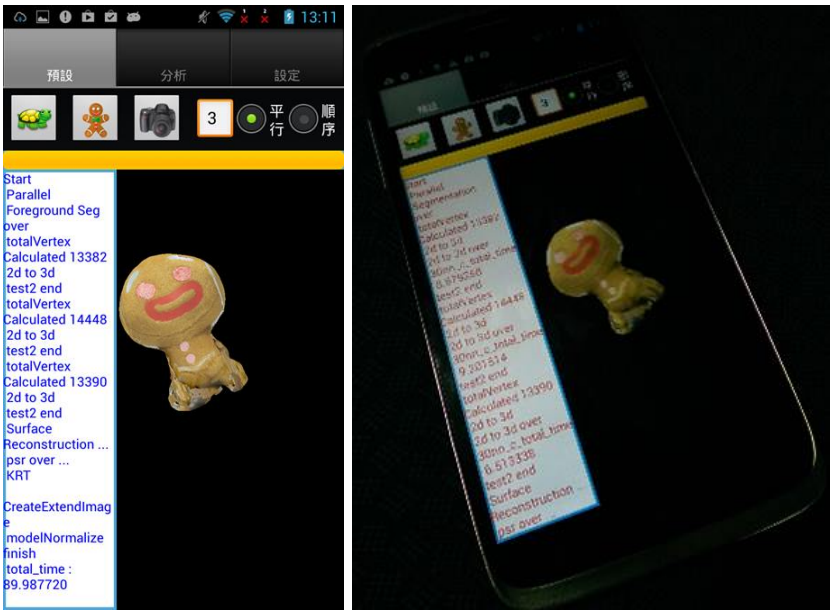


Fig. 9. Our system's srceenshot on mobile phone

while the parallel version only takes about 26.34496 seconds. In other words, comparing with the sequential C program, the kernel with GPU acceleration can achieve about 5.8 times speedup in this case.

Table 3. Comparison of the execution time by the parallel algorithm by GPU and sequential version on CPU on PC platform

Module	**GPU(sec)**	**CPU(sec)**	**speedup ratio**
Segmentation	2.208	17.413	7.89
Visual hulls (2D to 3D)	0.009	0.037	4.11
Visual hulls (find neighbor)	0.646	3.983	6.15
Execution time	24.757	43.504	1.76

Table 4. Comparison of the execution time between the parallel algorithm with GPU speedup and the sequential version on ImFocus M320 mobile phone

	Sequential (sec)	**Parallel** (sec)	**speedup ratio**
Kernel	153.3481	26.34496	5.820777
Total	211.7985	92.04223	2.301102

6 Conclusion

We propose a novel approach for image segmentation from multi-view images, which is included in an automatic 3D reconstruction system. For each view, we compute the homography with the first view, and we analyze the multi-view information about the color distortion and gain value. We use these multi-view information to improve the segmentation results of CDM [6]. From our experimental results, we can see the reconstructed 3D model is satisfactory from its visual appearance. In addition, the 3D reconstruction system is quite efficient by using GPU implementation to speed up the computation.

References

1. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. 2 edn. Cambridge University Press, New York, NY, USA (2003)

2. Laurentini, A.: The visual hull concept for silhouette-based image understanding. IEEE Trans. Pattern Anal. Mach. Intell. **16** (1994) 150–162
3. Pollefeys, M., Koch, R., Gool, L.J.V.: Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. International Journal of Computer Vision **32** (1999) 7–25
4. Wang, R., Choi, J., Medioni, G.G.: Accurate full body scanning from a single fixed 3d camera. In: 3DIMPVT, IEEE (2012) 432–439
5. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In: ICCV. (2001) 105–112
6. Pham, V.Q., Takahashi, K., Naemura, T.: Foreground-background segmentation using iterated distribution matching. In: CVPR, IEEE (2011) 2113–2120
7. Rother, C., Kolmogorov, V., Blake, A.: "grabcut": interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. **23** (2004) 309–314
8. Vogiatzis, G., Hernandez, C.: Automatic camera pose estimation from dot pattern (2010) http://jabref.sourceforge.net/help/LabelPatterns.php.
9. Matusik, W., Buehler, C., Raskar, R., Gortler, S.J., McMillan, L.: Image-based visual hulls. In: SIGGRAPH. (2000) 369–374
10. Kazhdan, M., Hoppe, H.: Screened poisson surface reconstruction. ACM Trans. Graph. **32** (2013) 29:1–29:13
11. Burt, P.J., Adelson, E.H.: A multiresolution spline with application to image mosaics. ACM Trans. Graph. **2** (1983) 217–236