# Interactive Object Segmentation from Multi-view Images

Thi Nhat Anh Nguyen[a], Jianfei Cai[b,*], Jianmin Zheng[b], Jianguo Li[c]

[a]*Danang University of Technology, Vietnam*
[b]*School of Computer Engineering, Nanyang Technological University, Singapore 639798*
[c]*Intel Labs China*

**Abstract**

Despite the great progress on interactive image segmentation, image co-segmentation, 2D and 3D segmentation, there is still no workable solution to the problem: *given a set of calibrated or un-calibrated multi-view images (say, more than 40 images), by interactively cutting* $3 \sim 4$ *images, can the foreground object of the rest images be quickly cutout automatically and accurately*? In this paper, we propose a non-trivial engineering solution to this problem. Our basic idea is to integrate 3D segmentation with 2D segmentation so as to combine their advantages. Our proposed system iteratively performs 2D and 3D segmentation, where the 3D segmentation results are used to initialize 2D segmentation and ensure the silhouette consistency among different views and the 2D segmentation results are used to provide more accurate cues for the 3D segmentation. The experimental results show that the proposed system is able to generate highly accurate segmentation results, even for some challenging real-world multi-view image sequences, with a small amount of user input.

*Keywords:* Interactive image segmentation, multi-view image segmentation, image co-segmentation.

*Corresponding author.
*Email addresses:* ngt.nhatanh@gmail.com (Thi Nhat Anh Nguyen), asjfcai@ntu.edu.sg (Jianfei Cai), asjmzheng@ntu.edu.sg (Jianmin Zheng), jianguo.li@intel.com (Jianguo Li)

## 1. Introduction

This paper studies the problem of precisely segmenting a foreground object out of a set of multi-view images containing the object. The objective here is not to obtain an approximate boundary in each image, but to generate a highly accurate and coherent silhouette in each view. Such an accurate multi-view object segmentation is very useful for many applications such as 3D reconstruction [1], multi-view image editing, object recognition and teleconference [2].

One straightforward solution is to segment each image in the multi-view sequence separately. Although the performance of the existing automatic image segmentation methods is still far from satisfactory, the interactive segmentation approaches [3, 4, 5], which involve a small amount of user input, can already achieve very accurate segmentation. However, it would require too much user effort to interactively segment each individual multi-view image, considering that a multi-view image sequence often consists of $40 \sim 80$ images or more [1]. For a practical solution, it is highly desirable to only interactively segment a small number of images, say $3 \sim 4$ out of the entire multi-view image set.

Another way is to apply the recently developed image co-segmentation technique to jointly segment all the multi-view images. The image co-segmentation problem was first introduced in [6], which deals with automatically segmenting a similar foreground object from two images with unrelated backgrounds. It was later on being extended to scale invariance and multiple images and improved with a co-saliency prior or multiple cues [7, 8, 9]. User interactions have also been introduced into the co-segmentation problem in [10, 11] to improve the segmentation accuracy. However, image co-segmentation is different from multi-view segmentation. The co-segmentation algorithms do not assume that there exists strong multi-view geometry in the set of images and thus in general they are not able to achieve accurate and consistent cutout for multi-view images by only interactively editing $3 \sim 4$ images.

In addition to the possible solutions in the 2D domain, a 3D-domain approach has been proposed in [12], which deals with exactly the same problem we study here. The authors in [12] exploited the silhouette coherency constraint existing in the multi-view images and developed a fully automatic approach for the multi-view object segmentation. In particular, a 3D graph is built with edge weights derived from the corresponding projections and

foreground/background models obtained by assuming that the foreground object is located in the center of each image. Then, 3D graph-cut is performed to generate the 3D segmentation result. From the projection of the 3D segmentation result, an improved color model is updated, which is being used for the next round 3D segmentation. The whole process repeats until convergence.

Although the 3D domain approach [12] ensures the silhouette coherence well, it has some limitations. First, it only makes use of the color prior in the data term of the 3D graph-cut, which can produce good results in the cases that the foreground and background color distributions are simple and well separated. However, for many real-world examples (e.g. Fig. 6), the object is often captured in cluttered or camouflaged environment, for which the color model itself is insufficient to distinguish the foreground from the background. Second, it assumes that the object of interest is located at the center of each image, which is not always true.

A similar multi-view segmentation problem has also been studied in [13], where a nice theoretical framework is proposed to jointly consider 2D and 3D information to cluster and label 2D regions and 3D point clouds at the same time. However, the entire system is quite complex, requiring the generation of 3D point clouds, the generation and filtering of 2D regions, the visibility detection for 3D points, the graph construction using the k-nearest neighbor algorithm (kNN), etc, and only one indoor multi-view sequence is tested.

Despite these previous efforts, to the best of our knowledge, there is still no workable solution to the problem: *given a set of calibrated or un-calibrated multi-view images (say, more than 40 or 80 images), by interactively cutting* $3 \sim 4$ *images, can the foreground object of the rest images be quickly cutout automatically and accurately*? This is exactly what this paper can offer. Our work is motivated by the observation that all the multi-view images represent the same scene and hence there exists strong geometric coherence among these images and the actual 3D object-of-interest as well. This property should be considerably utilized in developing a multi-view segmentation tool. Our basic idea is thus to iteratively perform 2D and 3D segmentation, where the 3D segmentation results are used to initialize 2D segmentation, which ensures the silhouette consistency among different views, and the 2D segmentation results are used to provide more accurate cues for the 3D segmentation.

In particular, we first use interactive 2D image segmentation to segment only a few images at high accuracy so as to introduce high-level prior into the

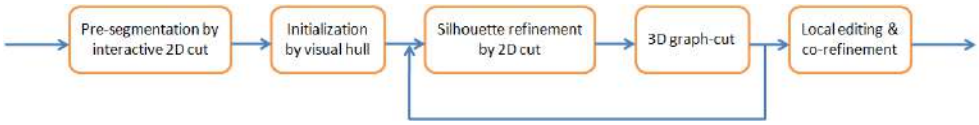



Figure 1: The diagram of the proposed multi-view object segmentation system.

system as hard constraints. Since only a few images need to be interactively segmented, the required user effort is of small amount. Second, considering that each individual multi-view image in fact contains a great amount of information, not just color, regarding the foreground and the background, we extend the iterative 3D volumetric graph-cut developed in [12] in two aspects: incorporating not only the color cue but also the silhouette cue, and enforcing the hard constraints extracted from the initial user interaction. The silhouette cue is generated and iteratively updated using an innovative 2D convex active contour model. In addition, a local editing and refinement step is also introduced to allow the user to edit the most erroneous image and quickly update the segmentation results of the entire multi-view sequence.

Despite the fact that most of the components used in the proposed multi-view object segmentation system are prior arts and the fundamental theory is not new, this paper does make noticeable contributions in terms of providing a non-trivial engineering solution to the multi-view object segmentation problem. Specifically, our major contribution is that we propose to incorporate the user prior, obtained through interactive image segmentation in a small set of images, as hard constraints, the silhouette cue obtained through the 2D convex active contour method as well as the color cue into the iterative 3D segmentation framework. Such an integration is non-trivial and the experimental results show that the proposed system is able to generate highly accurate segmentation results, even for some challenging real-world multi-view image sequences (see Fig. 6), which has not been seen in literature before. Note that there are only a few works specifically focusing on multi-view image segmentation as discussed above and publicly available multi-view sequences such as Middlebury datasets [1] are of simple background and captured in indoor environment. Thus, we have made efforts to capture some real-world multi-view image sequences with complex object shape and background. We will release the captured real-world outdoor multi-view image dataset, which is another contribution to the community.

## 2. System Overview

The primary inputs of our system are a set of $M$ multi-view color images, $\{I_m | m = 1, \cdots, M\}$, and a set of associated projection matrices that could be obtained through camera calibration. Note that the camera calibration is not an indispensable step for the proposed system. For un-calibrated multi-view images, the projection matrices can be obtained by using the structure-from-motion technique [14]. A projection matrix allows the center of any voxel $v_n \in R^3$ to be mapped to its corresponding location $x_{m,n} \in R^2$ in image $I_m$. The output of our system is the segmentation result for all the images, which can be represented by the object silhouettes or the foreground/background binary images.

Fig. 1 shows the diagram of the proposed multi-view object segmentation system. It can be seen that the entire system consists of five steps: pre-segmentation by interactive 2D cut (on a small subset of images), initialization by visual hull [15], silhouette refinement by 2D cut, 3D graph-cut and local editing and co-refinement. The detailed system flowchart is shown in Fig. 2, where the five main steps are highlighted in red.

The first step is to segment a small set of images $I_H$ (normally 3 to 4 images) sampled from the multi-view image sequence to obtain precise silhouettes using any of the existing interactive image segmentation methods such as Grabcut [3] or Geodesic [4], where the user typically needs to to draw two types of colored strokes on an image to label some pixels as foreground and background seeds to guide the segmentation process. The silhouettes of these images $I_H$ are used as hard constraints to guide the segmentation of the remaining images at the later steps.

The second step is to generate an initial silhouette for each of the remaining images, $I_m \notin I_H$. In particular, a visual hull containing the object is first generated from the pre-segmented silhouettes using the shape-from-silhouette method in [16]. The visual hull is then projected on each of the remaining images to generate an initial silhouette.

Considering that the initial silhouette obtained through 3D projection is not accurate, in the third step we extend a recently developed convex active contour model [17] to evolve the initial silhouette in each remaining image to snap to its nearby geometry features. The details of the developed convex active contour method will be discussed in Section 4.

In the fourth step, to ensure the segmentation coherence across different views, similar to [12], a 3D object segmentation via 3D graph-cut is per-

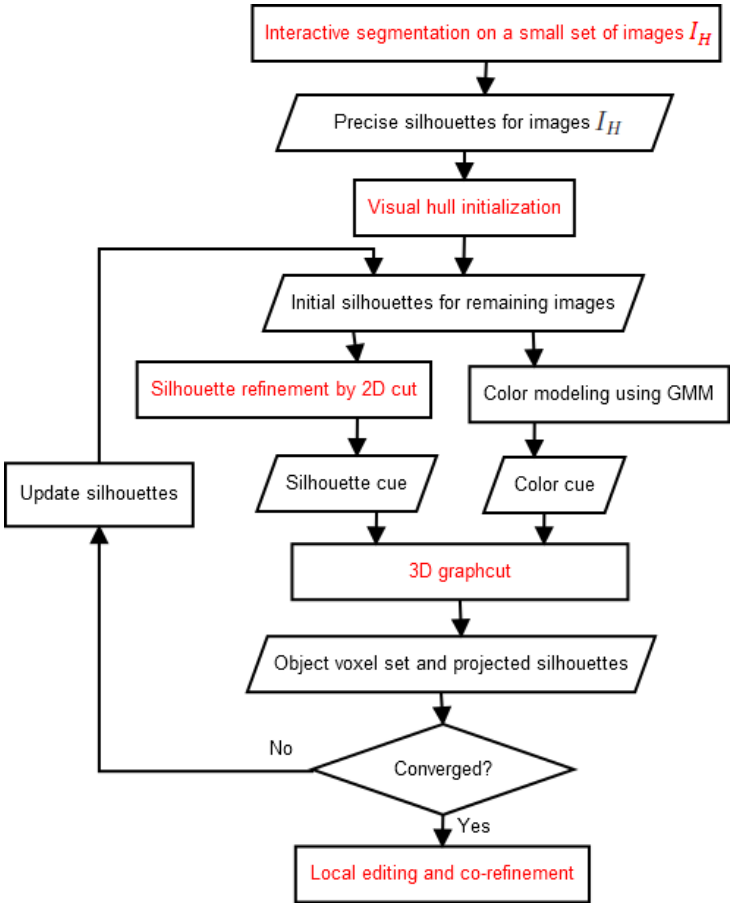



Figure 2: The flowchart of the proposed multi-view object segmentation system.

formed. However, unlike [12], the 3D graph-cut developed in this research utilizes not only a color cue but also a silhouette cue as well as hard constraints obtained from the initial user interaction. The details of the fourth step is described in Section 3. We would like to point out that step 3 and step 4 are tightly coupled and they together form an iteration process. In each iteration, the refined silhouette via the convex active contour provides better color and silhouette cues for the 3D graph-cut and in return the graph-cut produces better initial silhouette for the next-round silhouette refinement.

Finally, considering that for some complex multi-view image sequences there might still have some errors in the silhouettes after the iterative 2D and 3D segmentation, we introduce the last step to allow the user to easily and arbitrarily refine the segmentation result of any of the images locally with extra foreground or background strokes. After the local refinement, the silhouettes for the entire sequence will be updated accordingly. The proposed local editing and co-refinement scheme ensures a fast editing and updating speed. A smart image selecting scheme is also developed to automatically suggest the most erroneous image in the sequence to the user for local editing. Details of this last step can be found in Section 5.

## 3. 3D Graph-cut Using Color and Silhouette Cues

Similar to [12], a volumetric graph-cut algorithm is employed in our system to segment the object in 3D space. In particular, a 3D array of voxels $V$ is formed within the bounding box of the object's visual hull. We create a 3D undirected graph with nodes corresponding to voxels, $v_n \in V$, and two additional nodes: an "object" terminal (a source $S$) and a "background" terminal (a sink $T$). There are two types of undirected edges in the graph: n-links (neighborhood links) connecting each pair of neighboring voxels, $\{v_i, v_j\} \in E$, and t-links, $\{v_i, S\}$ and $\{v_i, T\}$, connecting each voxel $v_i$ to each of the two terminals. Let $w(v_i, v_j)$ denote the weight of the link $\{v_i, v_j\}$, representing how strong the link is or how similar the connected two nodes are.

Following [12], our 3D graph-cut operation is formulated as an energy minimization process:

$$O^* = arg \min_{O \subset V} E_d(O, \Theta) + \lambda_1 E_s(O) \tag{1}$$

where $E_d$ is the data term measuring the energy cost of a segmentation with a corresponding object voxel set $O$ against the prior model $\Theta$, $E_s$ is the

smoothness term measuring the energy cost of disconnecting the boundary links, and $\lambda_1$ is the trade-off factor. Unlike [12], which only uses the color cue for $\Theta$, we incorporate both color and silhouette cues.

### *3.1. Smoothness Term*

As in [12], the smoothness term measures the energy cost associated with the surface area of the object, i.e. the summation of the weights of all the boundary links:

$$E_s = \sum_{\{v_i,v_j\}\in E, v_i\in O, v_j\in B} w(v_i, v_j), \tag{2}$$

where $E$ is the set of n-links, and $O$ and $B$ are the object and the background voxel sets, respectively. The n-link weight $w(v_i, v_j)$ is defined according to the minimum pixel color difference of the projections of $v_i$ and $v_j$ over all the views:

$$w(v_i, v_j) = \max_m e^{-\beta_m ||I_m(x_{m,i}) - I_m(x_{m,j})||^2} \tag{3}$$

where $I_m$ is the $m$-th image, $x_{m,i}$ and $x_{m,j}$ are the projected pixels of $v_i$ and $v_j$ on image $m$, respectively, and $\beta_m$ is a parameter estimated from image $m$ as in [3].

### *3.2. Data Term*

The data term provides the preference for a voxel to be classified as object or background according to the prior model. It can be expressed as

$$E_d = y_n w(v_n, T) + (1 - y_n) w(v_n, S) \tag{4}$$

where $y_n$ is the label of $v_n$ and

$$\begin{cases} y_n = 1, & v_n \in O \\ y_n = 0, & v_n \in B, \end{cases}$$

and $w(v_n, T)$ and $w(v_n, S)$ are the weights of the t-links $\{v_i, T\}$ and $\{v_i, S\}$, respectively.

Considering that the actual object lies entirely within the initial visual hull, the set of voxels that is outside the initial visual hull, which is denoted as $B_H$, is confirmed to be background voxels. To enforce this hard-constraint for the 3D graph-cut, we set

$$\forall v_n \in B_H, \quad w(v_n, S) = 0, \quad w(v_n, T) = A \tag{5}$$

where $A$ is a constant whose value is large enough to force any $v_n \in B_H$ to be a background voxel (see Section 6.1 for details). For other voxels, their t-link weights are computed according to the color and silhouette cues.

In particular, $\forall v_n \notin B_H$, their t-link weights are defined as

$$\begin{aligned} w(v_n, S) &= 1 + \left[ \left\{ \frac{1}{M} \sum_{m=1}^{M} L_m(x_{m,n}, \Theta) \right\} - \phi \right] \\ w(v_n, T) &= 1 - \left[ \left\{ \frac{1}{M} \sum_{m=1}^{M} L_m(x_{m,n}, \Theta) \right\} - \phi \right] \end{aligned} \tag{6}$$

where $L_m(x_{m,n}, \Theta) \in [0, 1]$ is the likelihood that the projected pixel $x_{m,n}$ of $v_n$ on image $I_m$ belongs to foreground. In (6), we combine the probabilities from the individual images with a threshold parameter $\phi \in [0, 1]$, which encodes the level of robustness to noise of the algorithm in the same way as that in [12]. In this way, a voxel $v_n$ is more likely to be classified as inside the object if $w(v_n, S) > w(v_n, T)$, which happens when the averaging probabilities across all the images $\left\{ \frac{1}{M} \sum_{m=1}^{M} L_m(x_{m,n}, \Theta) \right\}$ is larger than $\phi$. Considering the special case of perfect object segmentation in all the images, i.e. the case of ideal binary classification where $L_m(x_{m,n}, \Theta)$ is either 0 or 1, a voxel would only be classified as an object voxel when all the silhouettes agree that the projection of that voxel is always a foreground pixel, i.e. $L_m(x_{m,n}, \Theta) = 1$ for all the images. Therefore, in this case, $\phi \to 1$ since $\left\{ \frac{1}{M} \sum_{m=1}^{M} L_m(x_{m,n}, \Theta) \right\} \to 1$. In our algorithm, $\phi$ is set to a value between 0.8-0.9 to tolerate the imperfect classification in each image.

Now we show the key issue of how to calculate the probability $L_m(x_{m,n}, \Theta)$ for each individual image. Specifically, for an image $I_m$ that does not belong to the set $I_H$ of the pre-segmented images, the probability is computed as

$$L_m(x_{m,n}, \Theta) = \alpha_1 P_m(x_{m,n}) + (1 - \alpha_1) u_m(x_{m,n}) \tag{7}$$

where $P_m(x_{m,n})$ is the color cue that assesses the fitness of projected pixel $x_{m,n}$ of voxel $v_n$ on image $I_m$ to the foreground/background color models, which will be introduced in Section 3.3, $u_m(x_{m,n})$ is the silhouette cue representing the probability of pixel $x_{m,n}$ belonging to the foreground based on

the 2D cut, which will be discussed in Section 4, and $\alpha_1$ (empirically set as 0.5) is the tradeoff factor between the color cue and the silhouette cue.

For an image $I_m$ that belongs to the set $I_H$ of the pre-segmented images, if $x_{m,n}$ is a background pixel, then $v_n \in B_H$, the set of voxels that are confirmed to be outside of the object, for which the t-link weights are set as (5). Otherwise, if $x_{m,n}$ is a foreground pixel (labelling function $y_{m,n} = 1$), theoretically, $L_m(x_{m,n}, \Theta)$ should be equal to 1. However, we will show in Section 3.4 that $\forall I_m \in I_H$, $\forall y_{m,n} = 1$, $L_m(x_{m,n}, \Theta)$ needs to be carefully set in order to meet the hard constraints.

### *3.3. Gaussian Mixture Model Based Color Cue*

The color cue assesses the fitness of a pixel's color to the foreground/background color models. The color models represent the color distributions of the foreground and background regions. Similar to [12], we use Gaussian Mixture Model (GMM) for color modelling. Specifically, at each iteration, foreground and background GMMs are learnt from foreground and background seeds, which are derived from the current silhouette at each image. For the images that belong to the set of the pre-segmented images $I_H$, the foreground and background seeds are the entire segmented foreground and background regions. For the images that do not belong to $I_H$, it can be observed that the pixels that are far away from the object boundary are more likely to be correctly classified. Thus, we obtain the foreground and background seed regions in $I_m \notin I_H$ by shrinking the foreground and the background regions by a safe distance $D_1$. Considering that the voxels outside the initial visual hull are confirmed background voxels, the background seed region is further updated as the union of the shrinked background region and the initial background region resulted from the initial visual hull in the first step of our framework.

The foreground seed region in each image may not contain all the color of the object since one image only captures one view of the object. Thus, to build up a global color model for the object, we use the pixels from the foreground seeds of all the views. On the other hand, as the background varies over different views, a separate background GMM color model is learnt for each view. Considering that the background seed region at one view may not contain all the color information of the background at that particular view due to occlusion, we also sample the nearby views, whose camera directions are less than 45 degree away from the particular view.

Let $Pr_m(x|F)$ and $Pr_m(x|B)$ denote the probabilities that pixel $x$ in image $I_m$ fits the foreground and background GMM color models, respectively. The color cue $P_m(x)$ representing the normalized likelihood that pixel $x$ is a foreground pixel is calculated as

$$P_m(x) = \frac{Pr_m(x|F)}{Pr_m(x|F) + Pr_m(x|B)}. \tag{8}$$

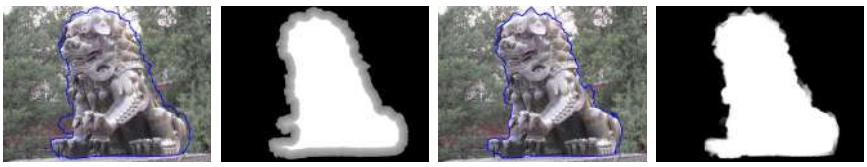
*3.4. Hard Constraint Enforcement*

In the first step of our framework, with sufficient user interaction, it is reasonable to deem that for the set $I_H$ of the pre-segmented images we obtain perfect silhouettes, which are used as hard constraints for the subsequent segmentation. However, the 3D graph-cut does not guarantee that the projection of the segmented 3D object on $I_H$ matches the hard constraints of the pre-generated silhouettes. Specifically, let $y_{m,n}$ and $\hat{y}_{m,n}$ respectively denote the segmentation results obtained in the first step of the pre-segmentation and the fourth step of the 3D graph-cut for pixel $x_{m,n}$ with $I_m \in I_H$ and $y_{m,n} = 1$. The issue here is how to set $L_m(x_{m,n}, \Theta)$ so as to ensure $\hat{y}_{m,n} = y_{m,n} = 1$.

In particular, $\forall I_m \in I_H$, $\forall x_{m,n}$, $y_{m,n} = 1$ implies that at least one of the 3D voxels lying in the optical line of pixel $x_{m,n}$ must be an object voxel. Let $v_{n_0}$ denote such an object voxel for pixel $x_{m,n}$. Considering that the t-link weights for $v_{n_0}$ is calculated according to (6), simply setting $L_m(x_{m,n}, \Theta)$=1 does not guarantee that the average likelihood over all the $M$ images is big enough to make $v_{n_0}$ being classified as an object voxel in the 3D graph-cut. On the other hand, setting $L_m(x_{m,n}, \Theta)$ to a value large enough to guarantee $\hat{y}_{m,n} = y_{m,n}$ might have the problem of making background voxels lying on the optical line of $x_{m,n}$ being classified as object voxels.

Thus, instead of assigning a fixed constant value to the foreground likelihood $L_m(x_{m,n}, \Theta)$, $\forall I_m \in I_H$, $\forall x_{m,n}$ with $y_{m,n} = 1$, we propose to find a value for $L_m(x_{m,n}, \Theta)$ adaptively through iterations. Specifically, $\forall I_m \in I_H$, $\forall x_{m,n}$ with $y_{m,n} = 1$, we initialize $L_m(x_{m,n}, \Theta) = 1$. After each iteration of the 3D graph-cut, if $\hat{y}_{m,n} \neq y_{m,n}$, we increase $L_m(x_{m,n}, \Theta)$ by a small constant value and then repeat the 3D graph-cut. The iteration stops when $\hat{y}_{m,n} = y_{m,n}$. Note that this is an inner iteration of the 3D graph-cut, which runs quite fast since there is no need to update the color and the silhouette cues.

Fig. 7 shows the projected segmentation results of a few images in the "stone" image sequence, where images numbered 1 and 5 are two of the three pre-segmented images. It can be seen that fixing $L_m(x_{m,n}, \Theta) = 1$ in the case



(a) Initial silhouette (b) Initial probability map $y$ (c) Refined silhouette (d) Silhouette prior $u$

Figure 3: An illustration of the proposed silhouette refinement. (a) an initial silhouette produced by the projection of the current 3D volume; (b) the initial probability map $y$ calculated by (10); (c) the refined silhouette using the constrained convex active contour; (d) the silhouette cue $u$ obtained by solving (9) and to be used in (7).

that $I_m \in I_H$ and $x_{m,n}$ is a foreground pixel results in significantly smaller foreground regions in all the images including images 1 and 5 (see the 2nd row of Fig. 7), while fixing $L_m(x_{m,n}, \Theta) = 1.5$ results in a bigger foreground regions (see the 3rd row of Fig. 7). On the contrary, the proposed adaptive selection of the values for $L_m(x_{m,n}, \Theta)$ leads to accurate foreground regions (see the 4th row of Fig. 7) that comply with the hard constraints of images 1 and 5.

## 4. Silhouette Refinement Via 2D Segmentation

The step of the silhouette refinement serves for two purposes: one is to refine the initial contour in each image produced by either the visual hull or the 3D graph-cut, and the other is to generate better silhouette cues for the subsequent 3D graph-cut. In this research, we adapt the powerful convex active contour model [17] for the silhouette refinement. The reason we choose the convex active contour model as a 2D segmentation method for the silhouette refinement lies in its strong capability to evolve the initial contour to snap to the geometry features/edges in an image and its fast processing speed due to convex optimization. Note that we have successfully adapted the convex active contour model for interactive image segmentation in [18]. For the self-contained purpose, in the following we briefly introduce the model and explain how to apply it for the silhouette refinement in the current context.

The convex active contour model proposed in [17] can be generally expressed as

$$\min_{0 \le u \le 1} \left( \int_\Omega g_b |\nabla u| dx + \lambda_2 \int_\Omega h_r u dx \right), \tag{9}$$

where $\nabla$ is the gradient operator, $u$ is a function on image domain $\Omega$ and receives a value between 0 and 1 at each pixel location $x$ in the image, function $g_b$ is typically an edge detection function (by default $g_b(x) = \frac{1}{1+|\nabla I(x)|^2}$, where $I(x)$ is the intensity of image pixel $x$), and function $h_r$ is a region function that measures the inside and outside regions. Essentially, Eq. (9) consists of two terms balanced by a tradeoff factor $\lambda_2$, where the first term is a boundary term and the second term is a region term. The boundary term favors the segmentation along the curves that the edge detection function reaches minimum and also favors the segmentation with smooth boundary curves. The second term ensures the segmentation complying with some region coherence criteria defined in function $h_r$. Once the optimization problem of (9) is solved, i,e. function $u$ has converged, a new boundary contour is found by thresholding the function $u$, i.e. the foreground pixel set $S_F = \{x|u(x) > T\}$ (by default $T = 0.5$).

Next we discuss how to apply (9) to refine the initial contour obtained by projecting the current 3D volumetric object into the particular view. In particular, let $y(x)$ denote a probability map over the image domain representing the probability that pixel $x$ belongs to the foreground object. We first set $y(x)$ according to the initial silhouette, where a value of 0 or 1 indicates a background or foreground pixel at location $x$ respectively. Considering that the areas far away from the current boundary are likely to be correctly classified, we update $y(x)$ as

$$y(x) = \begin{cases} 1, & \text{if } d(x) \geq D_2 \text{ and } y(x) = 1 \\ 0.5 + 0.5(d/D_2)^2, & \text{if } d(x) < D_2 \text{ and } y(x) = 1 \\ 0.5 - 0.5(d/D_2)^2, & \text{if } d(x) < D_2 \text{ and } y(x) = 0 \\ 0, & \text{if } d(x) \geq D_2 \text{ and } y(x) = 0 \end{cases} \tag{10}$$

where $d(x)$ denotes the Euclidean distance from pixel $x$ to its nearest boundary point, and $D_2$, a threshold value on $d$, is empirically set to $\frac{1}{10}\sqrt{R_o}$ with $R_o$ representing the size of the current foreground region.

We then build up local foreground and background GMM models for each image, where the sets of pixels with $y(x) = 1$ and $y(x) = 0$ are treated as foreground and background seeds, respectively. Note that the GMM models in Section 3.3 are different from the models here, where the previous GMM models are trained across different views and for the purpose of the 3D segmentation while the GMM models here are obtained from each individual image and for the purpose of the 2D segmentation.

With the local GMM models, the region term $h_r$ is then defined as

$$h_r(x) = \alpha_2(P_B(x) - P_F(x)) + (1 - \alpha_2)(1 - 2y(x)), \tag{11}$$

where $P_F(x)$ and $P_B(x)$ are the normalized foreground and background likelihoods respectively, calculated in a way similar to (8), and $\alpha_2$ is a tradeoff factor. The first term $(P_B(x) - P_F(x))$ in (11) ensures that the active contour evolves towards the one complying with the local GMM color models. For instance, for pixel $x$, if $P_B(x) > P_F(x)$ (resp. $P_B(x) < P_F(x)$) and $P_B(x) - P_F(x)$ is positive (resp. negative), $u(x)$ tends to decrease (resp. increase) during the contour evolution in order to minimize (9), which can lead to $u(x) \leq T$ (resp. $u(x) > T$) and the classification of the pixel belonging to the background (resp. the foreground). The second term $(1 - 2y(x))$ in (11) is to prevent the refined contour drifting too far apart from the initial silhouette. Specifically, when $y(x) > 0.5$ and $(1 - 2y(x))$ is negative, $u(x)$ tends to increase in order to minimize (9), which favors classifying the pixel as a foreground pixel, vice versa.

It is important to properly set the tradeoff factor $\alpha_2$ in (11). When the foreground and background colors are well separable, it is desired that the first term in (11) becomes dominating; otherwise, the second term in (11) should dominate. Thus, similar to [5], we set $\alpha_2$ to be the normalized Kullback-Leibler (KL) distance between the local foreground and the background GMM models.

The convex active contour model Eq. (9) can be solved efficiently using the Split Bregman method as in [18]. Once the optimization of the convex active contour model (9) is solved, we obtain the optimal solution of $u(x)$, denoted by $u_m(x)$ for image $I_m$, which represents the probability of a pixel $x$ belonging to the foreground ($0 \leq u_m(x) \leq 1$). A larger value of $u_m(x)$ means that pixel $x$ is more likely to belong to the foreground. $u_m(x)$ is then used as a silhouette cue in (7) for the next iteration of the 3D graph-cut. Fig. 3 illustrates how the proposed constrained active contour model evolves the initial silhouette to snap to the geometry features in this silhouette refinement step.

## 5. Local Editing and Co-refinement

The purpose of this local editing and co-refinement step is to allow the user to provide more strokes to edit the erroneous silhouette in one image, which will then be used to automatically refine the entire sequence.

Calculate the color histogram of the segmented object region in each image.

Normalize the histogram by the size of the corresponding object region.

The normalized color histogram of each of the pre-segmented images is used as the reference.

**for** each non-pre-segmented image $I_m$

Calculate the correlation between the normalized histograms of $I_m$ and each of the pre-segmented images.

Calculate the maximum correlation among the calculated correlations for $I_m$. This maximum correlation represents the similarity between the segmented object region of $I_m$ and the true object.

**end for**

Images are in turn suggested for editing in ascending order of the similarity between their segmented object regions and the true object.

Figure 4: Pseudocode for automatically selecting images for editing.

Considering that it is troublesome and time-consuming for the user to browse through the whole image sequence to select an image for editing, in this research we suggest an automatic method that selects the most erroneously segmented image for user editing. In particular, inspired by the co-segmentation idea in [6], i.e. the same foreground object in two different images shares similar histograms of image features, we compare the color histogram of the segmented foreground region in each non-pre-segmented image $I_m \notin I_H$ with that of the color histograms in the pre-segmented images $I_H$ to figure out the most erroneous segmentation.

In our implementation, we use only color histogram to identify the images with the most erroneous segmentation result but it can easily be extended to use other image features. Figure 4 gives the pseudocode for the algorithm to suggest the images for editing.

Note that even for the most erroneously segmented image, the erroneous areas are typically small. Thus, the user only needs to add strokes at the erroneous areas and there is no need to segment the entire image again. In particular, we use the same constrained convex active contour method to

(a) Input image (b) Initial segmentation result & extra user strokes (c) Tri-map (d) Segmentation result after editing

Figure 5: Local editing at an erroneous image

refine the user edited image here. A region $R$ with radius $r$ around the newly added user strokes are defined and only the pixels within this local region need to be relabelled, which is realized by setting all other pixels as foreground and background seeds according to their current labels.

Figure 5 illustrates the local editing process for the segmentation result of image 10 in the "lion" sequence. Initially, there are segmentation errors at the leg area of the lion (see Figure 5(b)). A few user strokes are added to those areas, where red and blue color strokes indicate foreground and background seeds respectively. Figure 5(c) shows the extended local region that needs to be relabelled (region $R$) and Figure 5(d) shows the refined segmentation result.

After editing and refining the most erroneously segmented image, we then perform the iterative 3D segmentation again to update the segmentation results of other images. Now the pixels in region $R$ of the edited image also become hard-constraints for the 3D graph-cut. To speed up the process, only the t-link weights of the voxels that have projected pixels lying in region $R$ need to be recalculated. Since only a small number of nodes are changed, the 3D graph-cut converges very fast, typically within 2 iterations. The fourth row of Fig. 6 gives an example of the overall local editing and refinement, where image 10 is the selected image added with a few user strokes, and eventually not only image 10 but also images 14 and 15 are refined.

## 6. Experimental Results

### *6.1. Parameters*

There are quite a few parameters in our proposed system. Most of the parameters have been discussed previously except $\lambda_1$ in (1), $A$ in (5), and $\lambda_2$

Table 1: Parameters used in the system.

| Parameter | Value |
|---|---|
| $\lambda_1$ in (1) | 0.2 |
| $\lambda_2$ in (9) | 100 |
| $A$ in (5) | 2 |
| $\phi$ in (6) | 0.85 |
| $\alpha_1$ in (7) | 0.5 |
| $\alpha_2$ in (11) | Normalized KL distance between local FG and BG GMMs |
| $D_1$ in Section 3.3, $D_2$ in (10) | 10% of the FG size |

in (9). Parameters $\lambda_1$ and $\lambda_2$ are empirically set to 0.2 and 100, respectively. Parameter $A$ in (5) needs to be a relatively large value to enforce the hard constraint. Particularly, we set $A = 2$ to be larger than $6\lambda_1$ times of the maximal n-link weight as well as larger than the t-link weight defined in (6). Note that unless specified, all the parameters are set in the same way in the experiments. Table 1 summarizes the parameters used in our system.

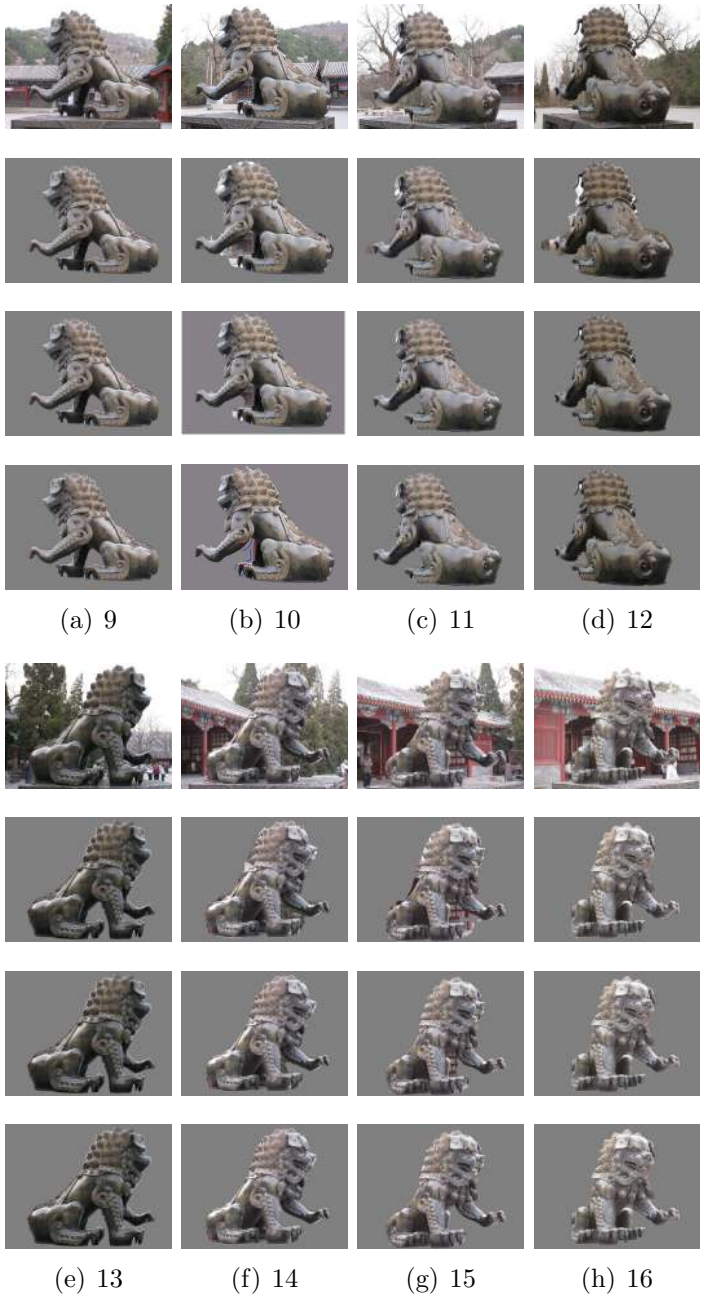
*6.2. Segmentation Results*

We first test two real-world multi-view image sequences: the 17-view 704x528 "lion" sequence and the 14-view 704x528 "stone"sequence, whose camera parameters are pre-computed. Fig. 6 shows the results of the "lion" sequence, where the average proportion of the object size in the images is 38.7%.. The "lion" sequence is a very challenging one because the foreground and background colors are very similar and the shape of the lion is very complex. It can be seen from Fig. 6 that with only the color cue, the multi-view object segmentation system is unable to produce an accurate object boundary in each image (see the second row of Fig. 6). In contrary, with both the color cue and the silhouette cue, the system can generate much more accurate segmentation results where in general the segmentation boundaries are smooth and snapped to the geometry features (see the third row of Fig. 6). In addition, with a little additional user input to image 10, we are able to generate almost perfect segmentations (see the fourth row of Fig. 6) through the local editing and refinement, as mentioned in Section 5. Fig. 7 shows the segmentation results of the "stone"sequence, where the average proportion of the object size in the images is 40.8%. Similarly, the proposed system is able to produce very accurate silhouette for each view.

Next, we consider segmenting relatively large-scale multi-view image sequences. Instead of the time-consuming process of capturing large number of multi-view images through placing the camera at different locations, we capture a video sequence using one camera moving around the rigid object and the video sequence is then uniformly sampled to generate a multi-view image sequence. Despite the convenience in data acquisition, the multi-view image sequence generated from video recording brings in more challenges. In particular, as the camera is moving during the recording, many of the images in the sequence are blurred, which results in blurred boundaries between foreground and background regions and also makes the automatic camera calibration less accurate, compared to the case of static multi-view image acquisition.

Fig. 8 shows the generated "tea pot" sequence, which consists of 56 frames with a resolution of 640x480 and four of them are pre-segmented. On average, the object regions occupy 17.6% of the images. It can be seen that the blurring and the shadow at the lower part of the tea pot makes it difficult to identify the object boundary even for human being (see the first row of Fig. 8). Thus, the pre-segmented silhouettes may not be perfect and may provide a poor hard-constraint for the multi-view segmentation. We therefore relax the hard constraints in (5) and Section 3.4, and directly set the likelihood $L_m(x_{m,n}, \Theta)$ to respectively be 1 and 0 for the foreground and background pixels within a small band around the pre-segmented silhouette. Surprisingly, without strong hard constraints, the proposed multi-view segmentation framework can still produce accurate silhouettes without using any additional user input (see the second row of Fig. 8). This is mainly because of the relatively large number of multi-view images available, which provides stronger 3D coherence and more hints among different views. Note that a demo video for segmenting this "tea pot" sequence is available at "http://www.ntu.edu.sg/home/asjfcai/MultiviewCut.wmv". Fig. 9 shows one application of making use of the resulted silhouettes in multi-view images, together with the bundled depth-map merging method [19], to construct a 3D model.

In addition to the above real-world multi-view sequences, our system is also tested on three commonly used Middlebury datasets: "DinoSparseRing", "DinoRing" and "TempleSparseRing". Table 2 summarizes the segmentation performance of all the sequences, with and without the silhouette cue, in terms of the error rate, i.e. the number of the mislabelled pixels over the object size. In all these experiments, four images are selected from each



(a) 9 (b) 10 (c) 11 (d) 12

(e) 13 (f) 14 (g) 15 (h) 16

Figure 6: An illustration of the segmentation results of the "lion" image sequence. The 1st and 5th rows show some of the original images. The 2nd and 6th rows are the results of our method with only color cue. The 3rd and 7th rows are the results of our method with both color and silhouette cues and the 4th and 8th rows show the final results after further local editing, where a little user input is added to image 10 (red and blue strokes are foreground and background seeds respectively). Note that images 9 and 13 are two pre-segmented images.

(a) 1 (b) 2 (c) 3 (d) 4 (e) 5 (f) 6

Figure 7: An illustration of the segmentation results of the "stone" image sequence. The first row contains some of the original images. The 2nd, 3rd and 4th rows are respectively the projected results with the fixed likelihood values $L_m(x_{m,n}, \Theta) = 1$, $L_m(x_{m,n}, \Theta) = 1.5$, ($\forall I_m \in I_H$, $\forall x_{m,n}$ with $y_{m,n} = 1$), and the proposed adaptive values as discussed in Section 3.4. Note that images 1 and 5 are two pre-segmented images.

(a) 24 (b) 28 (c) 32 (d) 36 (e) 44

Figure 8: An illustration of the segmentation results of the "tea pot" video sequence, which is casually captured by a hand-held camera. For the uniformly sampled 56 image frames, images 1, 16, 31 and 46 are pre-segmented in the first step. The first row shows some of the original images, and the second row shows the corresponding results of our system.

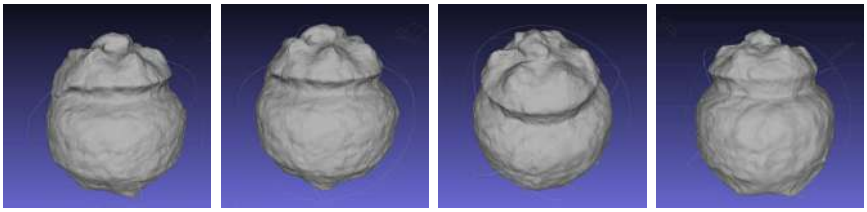


Figure 9: The 3D model reconstructed from the silhouettes of the "tea pot" video sequence.

Table 2: Error rates of the segmentation results of different multi-view sequences.

| Sequence | # of Views | Error Rate (%) | |
|---|---|---|---|
| | | with only color cue | with color & silhouette cues |
| Lion | 17 | 5.2 | 3.8 |
| Stone | 14 | 0.58 | 0.6 |
| Tea Pot | 56 | 2.1 | 0.9 |
| Dino Sparse Ring | 16 | 1.7 | 0.66 |
| Dino Ring | 48 | 1.72 | 0.6 |
| Temple Sparse Ring | 16 | 1.4 | 1.1 |

sequence for pre-segmentation. From Table 2, we can see that except for the "stone" sequence, where the error rate does not change much, for all the other sequences the error rates reduce significantly with the additional silhouette cue. Note that the results reported in Table 2 are the results without any local editing.

In all the experiments, the pre-segmented images in $I_H$ are selected in such a way that the projection angles from the object to these images are approximately equally spaced in 3D so that most of the points on the object surface are captured in these pre-segmented images and the initial visual hull can achieve a good approximation for the object volume. To evaluate the impact of the number of pre-segmented images on the final result, we repeat the experiment on the "tea pot" sequence using different numbers of sample frames for pre-segmentation. In particular, the number of pre-segmented frames is increased to 6, 8 and 10, and the resulted error rates are reduced to 0.16, 0.12 and 0.1, respectively. This suggests that the more frames we include in the pre-segmented image set, the better the segmentation result will be.

The proposed system is implemented in C++ and tested on a quad-

core Intel 3.33 GHz Xeon Processor with 16 GB RAM. To achieve a decent running speed, we use the Open Multi-Processing (OMP) API to parallelize several major processes. Particularly, in step 3 of the silhouette refinement, up to 8 threads are used to process multiple images in parallel. In step 4 of the 3D graph-cut, multiple voxels in the voxel array are also processed in parallel. The parallelized program spends about 4 minutes to segment the "lion" sequence with the voxel array size of $150^3$ and 1 minute to update the entire segmentation results after the local editing with the additional user strokes.

### *6.3. Limitations*

In general, the proposed system can handle most of the common objects, but it would not be able to perform well on the objects with a lot of trivial boundaries such as trees and hair, which is due to the inherent limitation of the adopted 2D and 3D segmentation methods. In addition, our current system only utilizes the color cue and the silhouette cue. It is possible to make use of other cues such as the stereo-matching cue to further improve the segmentation performance at the cost of increasing the difficulty in trading off among multiple cues.

## 7. Conclusion

In this paper, we have proposed a multi-view object segmentation system, which is able to accurately segment a foreground object out of a set of multi-view images with a small amount of user input and acceptable processing speed. The system is a nice and coherent integration of several techniques including interactive image segmentation, 3D graph-cut and the convex active contour. The experimental results have demonstrated that the proposed system is a practical and effective tool that can perform well for accurate multi-view object segmentation even for very challenging real-world outdoor multi-view image sequences.

## References

[1] S. Seitz, B. Curless, J. Diebel, D. Scharstein, R. Szeliski, A comparison and evaluation of multi-view stereo reconstruction algorithms, in: CVPR, 2006, pp. 519–528.

[2] J. Lu, V. A. Nguyen, Z. Niu, B. Singh, Z. Luo, M. N. Do, CuteChat: a lightweight tele-immersive video chat system, in: ACM Multimedia, 2011, pp. 1309–1312.

[3] C. Rother, V. Kolmogorov, A. Blake, "GrabCut": Interactive foreground extraction using iterated graph cuts, in: SIGGRAPH, 2004, pp. 309–314.

[4] X. Bai, G. Sapiro, A geodesic framework for fast interactive image and video segmentation and matting, in: ICCV, 2007, pp. 1–8.

[5] W. Yang, J. Cai, J. Zheng, J. Luo, User-friendly interactive image segmentation through unified combinatorial user inputs, IEEE Transactions on Image Processing 19 (9) (2010) 2470–2479.

[6] C. Rother, T. P. Minka, A. Blake, V. Kolmogorov, Cosegmentation of image pairs by histogram matching - incorporating a global constraint into MRFs, in: CVPR, 2006, pp. 993–1000.

[7] L. Mukherjee, V. Singh, J. Peng, Scale invariant cosegmentation for image groups, in: CVPR, 2011, pp. 1881–1888.

[8] K. Chang, T. Liu, S. Lai, From co-saliency to co-segmentation: an efficient and fully unsupervised envery minimization model, in: CVPR, 2011, pp. 2129–2136.

[9] S. Vicente, C. Rother, V. Kolmogorov, Object cosegmentation, in: CVPR, 2011.

[10] J. Cui, Q. Yang, F. Wen, Q. Wu, C. Zhang, L. J. V. Gool, X. Tang, Transductive object cutout, in: CVPR, 2008.

[11] D. Batra, A. Kowdle, D. Parikh, J. Luo, T. Chen, icoseg: Interactive co-segmentation with intelligent scribble guidance, in: CVPR, 2010, pp. 3169–3176.

[12] N. Campbell, G. Vogiatzis, C. Hernandez, R. Cipolla, Automatic 3D object segmentation in multiple views using volumetric graph-cuts, Image and Vision Computing 28 (1) (2010) 14–25.

[13] J. Xiao, J. Wang, P. Tan, L. Quan, Joint affinity propagation for multiple view segmentation, in: In ICCV, 2007, p. 43.

[14] N. Snavely, S. M. Seitz, R. Szeliski, Photo tourism: Exploring photo collections in 3D, in: SIGGRAPH, 2006, pp. 835–846.

[15] J.-S. Franco, , J. sbastien Franco, E. Boyer, Exact polyhedral visual hulls, in: In British Machine Vision Conference, 2003, pp. 329–338.

[16] A. Laurentini, The visual hull concept for silhouette-based image understanding, IEEE Trans. PAMI (1994) 150–162.

[17] X. Bresson, S. Esedoglu, P. Vandergheynst, J. Thiran, S. Osher, Fast global minimization of the active contour/snake model, Journal of Mathematical Imaging and Vision 28 (2) (2007) 151–167.

[18] A. Nguyen, J. Cai, J. Zhang, J. Zheng, Robust interactive image segmentation using convex active contours, IEEE Trans. on Image Processing 21 (8) (2012) 3734–3743.

[19] J. Li, E. Li, Y. Chen, L. Xu, Y. Zhang, Bundled depth-map merging for multi-view stereo, in: CVPR, 2010, pp. 2769–2776.