UNSUPERVISED COSEGMENTATION BASED ON SUPERPIXEL MATCHING AND FASTGRABCUT

Hongkai Yu and Xiaojun Qi

Department of Computer Science, Utah State University, Logan, UT 84322-4205
hongkai.yu@aggiemail.usu.edu and Xiaojun.Qi@usu.edu

ABSTRACT

This paper proposes a novel unsupervised cosegmentation method which automatically segments the common objects in multiple images. It designs a simple superpixel matching algorithm to explore the inter-image similarity. It then constructs the object mask for each image using the matched superpixels. This object mask is a convex hull potentially containing the common objects and some backgrounds. Finally, it applies a new FastGrabCut algorithm, an improved GrabCut algorithm, on the object mask to simultaneously improve the segmentation efficiency and maintain the segmentation accuracy. This FastGrabcut algorithm introduces preliminary classification to accelerate convergence. It uses Expectation Maximization (EM) algorithm to estimate optimal Gaussian Mixture Model(GMM) parameters of the object and background and then applies Graph Cuts to minimize the energy function for each image. Experimental results on the iCoseg dataset demonstrate the accuracy and robustness of our cosegmentation method.

Index Terms— Cosegmentation, Superpixel Matching, GrabCut, Graph Cuts

1. INTRODUCTION

Cosegmentation has been a lively research topic in the area of image segmentation. It can be potentially employed in many applications such as image retrieval, object tracking, object recognition, etc. The term of cosegmentation was first introduced by Rother *et al.* [1], wherein the goal is to simultaneously segment the common parts in an image pair. Later on, researchers refer to cosegmentation as the task of jointly segmenting the common object(s) (like a person) rather than similar stuff(s) (like grass) in a given set of images. In general, cosegmetnation techniques can be classified into two categories: supervised and unsupervised.

Supervised Cosegmentation: Cui *et al.* [2] propose to integrate a local color pattern and edge model learned from the segmentation of an example image to segment new images. Batra *et al.* [3] propose an interactive technique, which extends Graph Cuts [4] and requires human interactions to guide the cosegmentation in a group of images. Vicente *et al.* [5] train a Random Forest regressor based on powerful features extracted from the ground truth segmentation for object cosegmentation.

Unsupervised Cosegmentation: Rother *et al.* [1] propose a generative model to co-segment common parts of an image pair. This model leads to the minimization of a Markov Random Fields (MRF) based energy function which combines spatial coherency term with histogram constraints. Joulin *et al.* [6] combine normalized cuts with kernel methods in a discriminative clustering framework to assign object/background labels to all images in a set. Kim *et al.* [7] propose a distributed cosegmentation approach by exploiting the submodularity of the anisotropic heat diffusion-inspired segmentation objective. Rubio *et al.* [8] propose a MRF based method to co-segment the common objects in multiple images by minimizing an energy function containing inter-image region matching terms. Meng *et al.* [9] formulate the cosegmentation problem as the shortest path problem on a digraph using local region similarities and co-saliency.

All these aforementioned techniques produce decent cosegmentation results. However, supervised techniques either require ground truth segmentation or prior information for learning. Most unsupervised cosegmentation techniques formulate cosegmentation as an optimization problem and apply an iterative algorithm to infer object/background labels. Our proposed cosegmentation method belongs to the unsupervised type and offers the following advantages: 1) It is fully unsupervised without the need of ground truth segmentation or prior information. 2) It is able to co-segment common objects from dozens of images. 3) It allows some backgrounds to be matched and therefore is able to co-segment common objects in the images with similar backgrounds. 4) It has a simple structure and applies FastGrabCut algorithm, a more efficient variant of GrabCut algorithm, to minimize the energy function. 5) It well fits distributed systems and multicore processors because its energy function can be optimized in parallel.

The rest of the paper is organized as follows: Section 2 presents our proposed approach. Section 3 displays the experimental results. Section 4 draws conclusions and discusses the future research directions.

Fig. 1: Illustration of an original image (left) and its superpixel extraction results (right), where boundaries are shown in white.

2. PROPOSED METHOD

The proposed cosegmentation method consists of four steps: superpixel extraction, superpixel matching, object mask generation, and FastGrabCut segmentation. In the following, we explain each step in detail.

2.1. Superpixel Extraction

The aim of superpixel extraction is to obtain perceptually meaningful entities preserving most imagery structures to facilitate later processes. The watershed algorithm is applied on each image to obtain superpixels. Fig. 1 demonstrates a sample image and its superpixel extraction results. It clearly shows that the complexity of an image is reduced to a few hundred superpixels and most imagery structures are maintained at the same time.

2.2. Superpixel Matching

The aim of superpixel matching is to explore the inter-image similarity to find the matched superpixels between an image pair. The SIFT keypoint-based composite features are applied to find the matched superpixels, which potentially contain interested objects. First, SIFT keypoints [10], stable for matching and recognition, are detected on the original image. Second, Dense SIFT (DSIFT) descriptor [11] is extracted for all pixels. Third, composite features (e.g. DSIFT, color and texture) for each candidate superpixel, which contains at least one SIFT keypoint, are used to perform the matching. This SIFT keypoint-based composite feature matching technique is chosen due to the following: 1) SIFT keypoints are robust to variations of the scale, camera viewpoint, and pose. 2) The composite features have a high discriminative power.

Establishing correspondence between two images is performed locally. Superpixels containing SIFT keypoints are exclusively considered as the candidates for matching. Each candidate superpixel is represented by visual features including 128-D DSIFT, 3-D color, and 3-D texture features. Specifically, DSIFT features are computed as the average of DSIFT features of all pixels within the superpixel. Color features are computed as the average intensity of all pixels in each hue, saturation, and value channel within the superpixel. Texture features are computed as the average, standard deviation, and entropy of gray-level intensity of all pixels within the superpixel. A normalization technique is further applied on each feature component to ensure its values fall in $[0, 1]$.

The Euclidean distance based similarity between two candidate superpixels in a pair of images is computed by:

$$\begin{aligned} Dist(CS_{a,i}, CS_{b,j}) = w_1 d(DS_{a,i}, DS_{b,j}) + \\ w_2 d(Col_{a,i}, Col_{b,j}) + w_3 d(T_{a,i}, T_{b,j}) \end{aligned} \quad (1)$$

where $CS_{a,i}$ is the ith candidate superpixel of image a, $CS_{b,j}$ is the jth candidate superpixel of image b, $DS_{a,i}$ and $DS_{b,j}$ respectively are DSIFT features of $CS_{a,i}$ and $CS_{b,j}$, $Col_{a,i}$ and $Col_{b,j}$ respectively are color features of $CS_{a,i}$ and $CS_{b,j}$, $T_{a,i}$ and $T_{b,j}$ respectively are texture features of $CS_{a,i}$ and $CS_{b,j}$, $d(\cdot)$ is the Euclidean distance function, and w_1, w_2, and w_3 are corresponding positive influence weights for DSIFT, color, and texture features with their sum to be 1. Algorithm 1, which uses the similar mechanism as in [10], is applied to find whether each candidate superpixel in one image has a matched candidate superpixel in another image.

Algorithm 1 The proposed superpixel matching algorithm

Input: Image a and image b
Output: Matching flag vector F_a

1: Find M_a candidate superpixels $CS_{a,i}$ ($1 \leq i \leq M_a$) in image a and M_b candidate superpixels $CS_{b,j}$ ($1 \leq j \leq M_b$) in image b
2: Initialize vector F_a, whose length is M_a, as 0's
3: **for** each $CS_{a,i}$ of M_a candidate superpixels **do**
4: Initialize vector B, whose length is M_b, as 0's
5: **for** each $CS_{b,j}$ of M_b candidate superpixels **do**
6: Compute $Dist(CS_{a,i}, CS_{b,j})$ using Eq. 1
7: Save $Dist(CS_{a,i}, CS_{b,j})$ in $B(j)$
8: **end for**
9: Sort B in ascending order
10: Compute a distance ratio R by $B(1)/B(2)$
11: If $R < T_1$, $F_a(i) = 1$
12: **end for**

Here, R is the ratio of the similarity level of the best matched superpixel to the similarity level of the second best matched superpixel in terms of the composite feature-based Euclidean distance. Algorithm 1 considers the candidate superpixel with a smaller ratio has a matched candidate superpixel in another image. When R is smaller than a predefined threshold T_1, a match is located in image b for superpixel $CS_{a,i}$ and $F_a(i)$ is set to 1. Otherwise, there is no match for superpixel $CS_{a,i}$ and $F_a(i)$ is set to 0.

2.3. Object Mask Generation

The aim of object mask generation is to automatically obtain the mask containing the common objects for each image in the set. Algorithm 2 summarizes detailed steps to find the object mask for an image.

Fig. 2 illustrates the procedure to obtain the object mask of one sample image. The object mask of each image in the set can be similarly detected. Our matching method allows backgrounds to be matched and therefore generates the object mask stably, which makes our cosegmentation method work well for images with similar backgrounds.

Algorithm 2 The proposed object mask generation algorithm

Input: All NI images in the set

Output: Object mask for image a

1: **for** each image b in the set ($1 \leq b \leq NI, b \neq a$) **do**
2: Apply Algorithm 1 on images a and b to get F_a for a
3: Record superpixels in a that have a matched superpixel in b by finding $F_a = 1$
4: Find centroids of the recorded superpixels in a
5: Apply the convex hull algorithm on centroids to find object mask H_b (e.g., $Mask_{a,b}$)
6: **end for**
7: Intersect $NI - 1$ object masks H_b's to obtain the object mask for image a

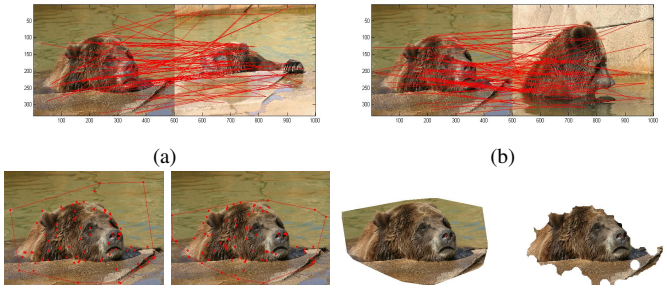

Fig. 2: Illustration of the process to obtain the object mask for the first image in a set of three images. (a) superpixel matching between the first and second images; (b) superpixel matching between the first and third images; (c) matched superpixels between the first and second images shown in red dots and their generated convex hull H_2 (e.g., $Mask_{1,2}$) shown in red lines; (d) matched superpixels between the first and third images shown in red dots and their generated convex hull H_3 (e.g., $Mask_{1,3}$) shown in red lines; (e) object mask of the first image obtained by intersecting H_2 and H_3; (f) Segmentation result obtained by the FastGrabCut algorithm.

2.4. FastGrabCut Segmentation

The widely-used GrabCut algorithm [12] can be applied to these object masks generated at the previous step to finish the cosegmentation. However, GrabCut algorithm utilizing iterative Graph Cuts for optimization is quite time-consuming. So we propose a new FastGrabCut algorithm to improve the efficiency of GrabCut algorithm. FastGrabCut shares the same input and similar principle of GrabCut algorithm. FastGrabCut algorithm simplifies the iterative Graph Cuts optimization into one-time Graph Cuts optimization to reduce the computational cost. It also uses preliminary classification to refine the labels of the observation data to accelerate convergence. Furthermore, it applies EM algorithm [13] to estimate optimal GMM parameters to achieve comparable segmentation accuracy as the original GrabCut algorithm.

FastGrabCut algorithm takes the object mask of each image as the input. These masks are convex hulls loosely covering common objects of images. Like GrabCut, superpixels within the convex hull may belong to either objects or backgrounds while superpixels outside the convex hull are all considered as backgrounds. FastGrabCut algorithm then builds a graph with the same structure in Graph Cuts [4], where each superpixel $S_{a,i}$ (ith superpixel in image a) has two t-links: one connecting to the background terminal (e.g., $(S_{a,i}, T_B)$) and one connecting to the object terminal (e.g., $(S_{a,i}, T_O)$). The edge costs of these two t-links, namely, $C(S_{a,i}, T_B)$ and $C(S_{a,i}, T_O)$, represent the similarity of $S_{a,i}$ with regards to background and object, respectively. Each superpixel $S_{a,i}$ also has n-links connecting to its neighboring superpixels. The edge costs of these n-links are computed using Euclidean distance of neighboring superpixels in color space, which measures the local coherence of $S_{a,i}$ with its neighboring superpixels.

After constructing the graph for image a, the proposed FastGrabCut algorithm applies parameter initialization, preliminary classification, EM estimation, and energy minimization to segment image a. In the following, we explain each of these steps in detail.

Step 1: Parameter Initialization. Its goal is to quickly estimate the GMM parameters θ_a (e.g., $\theta_a(O)$ for object and $\theta_a(B)$ for background) for image a. We initially assume superpixels within the object mask belong to objects. K-means algorithm (e.g., $K = 5$) is applied on all the superpixels within the object mask to obtain K clusters, representing the object GMM model. Similarly, the same K-means algorithm is applied on the superpixels outside the object mask to obtain K clusters, representing the background GMM model. Once each superpixel is uniquely assigned to a Gaussian component corresponding to its cluster, the initial parameter θ_a^0 of the object and background GMM models can be easily computed. θ_a includes weights π, means μ, and covariances Σ of $2K$ Gaussian components of object and background GMM models in image a. In other words, $\theta_a = \{\pi(\alpha, k), \mu(\alpha, k), \Sigma(\alpha, k)\}$, where $\alpha(S_{a,i})$ represents the label of superpixel $S_{a,i}$ and $k(S_{a,i})$ represents the Gaussian component of this superpixel. Here, $\alpha(S_{a,i}) \in \{0, 1\}$ with 0 being background and 1 being object. Superpixel $S_{a,i}$ is assigned to a unique GMM component $k(S_{a,i}) \in \{1, ..., K\}$ according to $\alpha(S_{a,i}) = 0$ or 1.

Step 2: Preliminary Classification. This step is to accelerate convergence of the algorithm by quickly classifying each superpixel within the object mask. Edge costs of t-links for each superpixel $S_{a,i}$ within the object mask of image a are computed using θ_a^0. Specifically, the t-link connecting to the background terminal, $C(S_{a,i}, T_B)$, is computed as the minimum of negative log-likelihoods of the superpixel to image a's object model. The t-link connecting to the object terminal, $C(S_{a,i}, T_O)$, is computed as the minimum of negative log-likelihoods of the superpixel to image a's background model. Each edge cost of t-links is a penalty computed by:

$$\begin{aligned} C(S_{a,i}, T_B) &= D(\alpha(S_{a,i}), k(S_{a,i}), \theta_a^0(O)) \\ C(S_{a,i}, T_O) &= D(\alpha(S_{a,i}), k(S_{a,i}), \theta_a^0(B)) \end{aligned} \quad (2)$$

where

$$
\begin{aligned}
D(\alpha(S_{a,i}), k(S_{a,i}), \theta) &= -\log\pi(\alpha(S_{a,i}), k(S_{a,i})) \\
&\quad -\log p(S_{a,i}|\alpha(S_{a,i}), k(S_{a,i}), \theta) \\
k(S_{a,i}) &= \underset{k(S_{a,i})}{\operatorname{argmin}} D(\alpha(S_{a,i}), k(S_{a,i}), \theta)
\end{aligned} \tag{3}
$$

where $p(\cdot)$ is the Gaussian probability distribution.

Edge costs of t-links reflect the similarity to object and background. If the edge cost of a superpixel's background t-link is bigger than or equal to the edge cost of its object t-link, the superixel is considered to be more similar to background. Therefore, we define the label $\alpha(S_{a,i})$ for each superpixel $S_{a,i}$ in the object mask of image a by comparing its t-links as follows:

$$
\alpha(S_{a,i}) = \begin{cases} 0, if\ C(S_{a,i}, T_B) \geq C(S_{a,i}, T_O) \\ 1, otherwise \end{cases} \tag{4}
$$

Step 3: EM Estimation. The purpose of this step is to find the maximum likelihood estimates of parameters for both object and background GMM models. It starts with already classified object and background superpixels as observations X. It then applies the following iterative procedures until the convergence condition is satisfied.

E-step: Calculate the conditional expectation of the log-likelihood under the current estimates and observations:

$$
Q(\theta_a; {\theta_a}^j) = E_{Z^*|X,{\theta_a}^j}[\log(p(X, Z^*|\theta_a))] \tag{5}
$$

where Z^* is a set of unobserved latent data or missing values and ${\theta_a}^0$ obtained in Step 1 is used to initialize GMM parameters θ_a. This initialization makes the EM algorithm converge more quickly compared to the random initialization of the parameters.

M-step: Compute the maximum likelihood estimates of parameters θ_a by maximizing the conditional expectation. Then θ_a is updated by:

$$
{\theta_a}^{j+1} = \underset{\theta_a}{\operatorname{argmax}} Q(\theta_a; {\theta_a}^j) \tag{6}
$$

The EM algorithm yields optimal GMM parameters ${\theta_a}^*$ after convergence. Each superpixel is then assigned to an optimal GMM component, $k_{opt}(S_{a,i})$, which is updated by ${\theta_a}^*$:

$$
k_{opt}(S_{a,i}) = \underset{k(S_{a,i})}{\operatorname{argmin}} D(\alpha(S_{a,i}), k(S_{a,i}), {\theta_a}^*) \tag{7}
$$

Step 4: Energy Minimization. This step models the segmentation problem as a binary labeling of MRF and optimizes it. This labeling problem re-assigns the label $\alpha(S_{a,i}) \in \{0, 1\}$, $i = 1, ..., N$ for each of N superpixels in image a. Based on ${\theta_a}^*$ and $k_{opt}(S_{a,i})$, the labeling problem for image a is then solved by minimizing the following MRF-based Gibbs energy function $E_a(\alpha, k_{opt}, \theta_a^*)$:

$$
E_a(\alpha, k_{opt}, \theta_a^*) = U_a(\alpha, k_{opt}, \theta_a^*) + V_a(\alpha) \tag{8}
$$

This energy function of image a is computed as the sum of the data term U_a and the smoothness term V_a. Here, U_a refers to edge costs of t-links and evaluates the overall fit of the labels to the given model θ_a^* in image a. It is computed as:

$$
U_a(\alpha, k_{opt}, \theta_a^*) = \sum_{i=1}^{N} D(\alpha(S_{a,i}), k_{opt}(S_{a,i}), \theta_a^*) \tag{9}
$$

V_a refers to edge costs of n-links and encourages coherence in local regions with similar features in image a. It is computed as:

$$
V_a(\alpha) = \sum_{(m,n)\in NS} [\alpha_{S_{a,m}} \neq \alpha_{S_{a,n}}] \exp(-\beta \left\| S_{a,m} - S_{a,n} \right\|^2) \tag{10}
$$

where $[\phi]$ denotes the indicator function taking values of 1 for a true predicate and 0 for a false predicate, β is a scale parameter (e.g., $\beta = 0.01$), and NS is the set of pairs of neighboring superpixels. The MRF-based Gibbs energy function of image a is finally minimized by one-time Graph Cuts to obtain the optimal labels for each superpixel in image a. Fig. 2 also shows the segmentation result of applying the proposed FastGrabCut algorithm on the first sample image.

Finally, the FastGrabCut algorithm is separately applied on each image in the set to optimize its corresponding energy function. Please note that the energy function in Eq. 8 is applied on one image. For NI images in a set, we execute FastGrabCut algorithm on the object mask of each image in parallel. In other words, NI threads of FastGrabCut algorithms can be simultaneously run on NI images after executing Algorithm 2. Thus, our method well fits distributed systems and multi-core processors.

3. EXPERIMENTAL RESULTS

To evaluate our method, we first compare the GrabCut algorithm with the proposed FastGrabCut algorithm and then compare our method with two state-of-the-art unsupervised cosegmentation algorithms. Both experiments are performed on the challenging iCoseg dataset [3], which has significant changes in scale, illumination, and camera viewpoint. This iCoseg dataset contains 643 images of 38 classes with manually labeled segmentation ground truth. To ensure fair comparison, we use the same groups of 16 test classes reported in [8] for our experiments. The same segmentation accuracy measure, which is computed as the ratio of the number of correctly labeled pixels to the total number of pixels in an image, is used for the comparison. All experiments are performed on a computer with 2.83GHz CPU and 3GB memory.

3.1. Comparison of GrabCut and FastGrabCut

Fig. 3 compares segmentation results of the proposed FastGrabCut and the default 12-iteration GrabCut algorithms on one "Red sox players" image. It clearly shows that the two algorithms achieve similar segmentation results.

We respectively run the proposed FastGrabCut and the GrabCut algorithms on five randomly selected images in each of 16 iCoseg test classes to compare their performance. For each chosen image, the user drags an input mask around

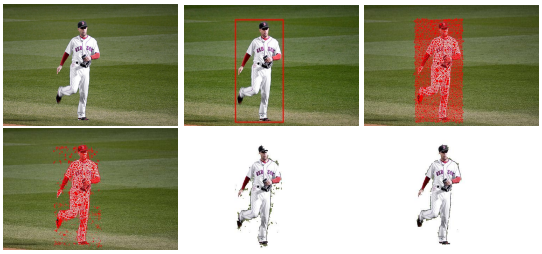

Fig. 3: Comparison of the segmentation results obtained by FastGrabCut and the default 12-iteration GrabCut algorithms. Images at the upper row: original image (left), red object mask outlined by the user (middle), and initial object superpixels shown in red (right). Images at the lower row: object superpixels after preliminary classification shown in red (left), segmentation result of FastGrabCut (middle), and segmentation result of 12-iteration GrabCut (right).

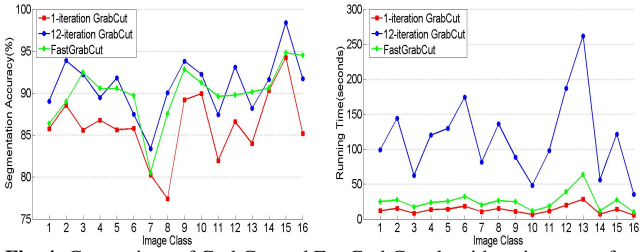

Fig. 4: Comparison of GrabCut and FastGrabCut algorithms in terms of segmentation accuracy (left plot) and running time (right plot).

the object, which is then given to 1-iteration GrabCut, 12-iteration GrabCut, and FastGrabCut algorithms to compare their segmentation accuracy and running time. We supply four kinds of masks, e.g., the exact rectangle covering the object, the bigger rectangle containing the object, the exact convex hull covering the object, and the bigger convex hull containing the object, for each chosen image to simulate typical sample inputs from different users. For each class, we then compute the average segmentation accuracy and running time of five chosen images under four input masks for the compared algorithms. Fig. 4 shows that the average segmentation accuracy of 1-iteration GrabCut, 12-iteration GrabCut, and FastGrabCut for 16 classes is 86.1%, 90.8%, and 90.0%, respectively. Their average running time for 16 classes is 13.1, 115.0, and 25.1 seconds, respectively. FastGrabCut achieves just 0.8% lower segmentation accuracy than 12-iteration GrabCut but spends only 21.8% running time of 12-iteration GrabCut. Due to EM estimation, FastGrabCut spends 12.0 seconds more than 1-iteration GrabCut but achieves a 3.9% higher segmentation accuracy, which is a significant improvement.

3.2. Comparison of Cosegmentation Results

Table 1 compares the cosegmentation accuracy and its standard deviation of our method with two state-of-the-art unsupervised cosegmentation algorithms [6] and [8]. To ensure fair comparison with [6], we randomly select the same number of images in each of 16 iCoseg test classes for the first experiment. To ensure fair comparison with [8], we randomly select (at most) 10 images in each of 16 iCoseg test classes,

Table 1: Comparison of cosegmentation accuracy and its standard deviation. First column: 16 iCoseg classes with its total number of images. Second and third columns: results of the first experiment where the number of selected images for each class is reported in the corresponding bracket in the second column. Last two columns: results of the second experiment. Bold numbers indicate classes in which our method outperforms the compared method.

iCoseg	Our method	[6]	Our method	[8]
Alaskan bear (19 images)	**80.0%** (9)	74.8%	78.0%	86.4%
Red sox players (25 images)	**85.4%** (8)	73.0%	86.4%	90.5%
Stonehenge 1 (5 images)	**85.4%** (5)	56.6%	**87.4%**	87.3%
Stonehenge 2 (18 images)	71.5% (9)	86.0%	74.7%	88.4%
Liverpool (33 images)	**87.7%** (9)	76.4%	**82.9%**	82.6%
Ferrari (11 images)	80.7% (11)	85.0%	**86.1%**	84.3%
Taj Mahal (5 images)	**85.2%** (5)	73.7%	85.4%	88.7%
Elephants (15 images)	**82.5%** (7)	70.1%	**82.2%**	75.0%
Pandas (25 images)	75.6% (8)	84.0%	**76.8%**	60.0%
Kite (18 images)	86.6% (8)	87.0%	85.2%	89.8%
Kite Panda (7 images)	**77.8%** (7)	73.2%	**80.1%**	78.3%
Gymnastics (6 images)	85.2% (6)	90.9%	**90.7%**	87.1%
Skating (11 images)	77.7% (7)	82.1%	**77.3%**	76.8%
Hot Balloons (24 images)	**87.9%** (8)	85.2%	86.3%	89.0%
Liberty Statue (41 images)	84.7% (10)	90.6%	87.3%	91.6%
Brown Bear (5 images)	**81.7%** (5)	74.0%	**81.5%**	80.4%
Average Accuracy	82.2%	78.9%	83.0%	83.5%
Standard Deviation	4.7%	9.0%	4.6%	8.1%

which is the same setting in [8], for the second experiment. We then co-segment the group of images in each class and average the segmentation accuracy for each class. For each class, we repeat the random selection of the same number of images for 3 times for both experiments. The final cosegmentation accuracy of a class is computed as the average of cosegmentation accuracy of all 3 random selections. In our experiments, we set $w_1 = 0.6, w_2 = w_3 = 0.2$ and $T_1 = 0.9$.

Tabel 1 also lists the overall average accuracy and standard deviation for each compared method across all classes in the last two rows. It shows that our method achieves the average accuracy of 82.2% in the first experiment. Compared to [6], it achieves better accuracy for nine classes and comparable accuracy for one class (Kite). On average, our accuracy is 3.3% higher than [6]. Compared to [8], our method achieves the comparable average accuracy of 83.0% (just 0.5% lower) and better accuracy for nine classes in the second experiment. However, [8] uses more complex region matching and optimization algorithms than our method. Particularly, our method achieves much higher accuracy than both [6] and [8] for Elephant class, which has high similarity in backgrounds. This impressive result is mainly because FastGrabCut uses the object mask containing object and some backgrounds as an input and allows some backgrounds to be matched. Among 16 classes, our method achieves the worst cosegmentation accuracy for the two classes, Stoneheng 2 and Pandas. This is mainly caused by matching errors in complicated situations such as extremely different lighting conditions in Stoneheng 2 class and complex composition of black and white furs in Pandas class. However, our method is more stable and robust than [6] and [8] for different image classes because of the smaller standard deviation.

Fig. 5 shows sample results of two images after jointly segmenting a selected number of images in a class. It demonstrates our method is effective in co-segmenting multiple images with similar backgrounds under different poses, scales,

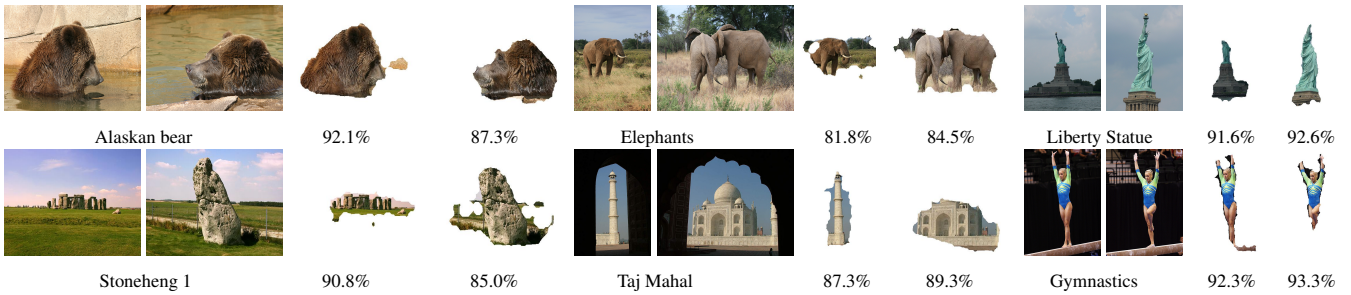

Fig. 5: Sample cosegmentation results for the iCoseg dataset. Three classes are reported in each row. For each class, the first two columns show two original images and the last two columns show their corresponding cosegmentation results. The segmentation accuracy of each image is reported below its cosegmentation result.

and camera viewpoints.

4. CONCLUSIONS

We present a novel unsupervised method to co-segment multiple images with common objects. Our contributions are: 1) Designing a simple superpixel matching algorithm to explore the inter-image similarity. 2) Designing an effective framework to make our cosegmentation method work well for images of similar backgrounds. 3) Designing a FastGrabCut algorithm to simultaneously improve the segmentation efficiency and maintain the segmentation accuracy of the GrabCut algorithm. The experimental results demonstrate the accuracy and robustness of our method.

In the future, we will focus on designing robust object matching algorithms to help the FastGrabCut algorithm get better cosegmentation results.

5. REFERENCES

[1] C. Rother, T. Minka, A. Blake, and V. Kolmogorov, "Cosegmentation of image pairs by histogram matching-incorporating a global constraint into mrfs," in *Proceedings. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006, pp. 993–1000.

[2] J. Cui, Q. Yang, F. Wen, Q. Wu, C.Zhang, L.V. Gool, and X. Tang, "Transductive object cutout," in *Proceedings. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1–8.

[3] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen, "icoseg: Interactive co-segmentation with intelligent scribble guidance," in *Proceedings. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 3169–3176.

[4] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary region segmentation of objects in nd images," in *Proceedings. IEEE International Conference on Computer Vision (ICCV)*, 2001, pp. 105–112.

[5] S. Vicente, C. Rother, and V. Kolmogorov, "Object cosegmentation," in *Proceedings. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 2217–2224.

[6] A. Joulin, F. Bach, and J. Ponce, "Discriminative clustering for image co-segmentation," in *Proceedings. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 1943–1950.

[7] G. Kim, E.P. Xing, L. Fei-Fei, and T. Kanade, "Distributed cosegmentation via submodular optimization on anisotropic diffusion," in *Proceedings. IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 169–176.

[8] J.C. Rubio, J. Serrat, A. Lpez, and N. Paragios, "Unsupervised co-segmentation through region matching," in *Proceedings. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 749–756.

[9] F. Meng, H. Li, G. Liu, and K.N. Ngan, "Object co-segmentation based on shortest path algorithm and saliency model," *IEEE Transactions on Multimedia*, vol. 14, no. 5, pp. 1429–1441, 2012.

[10] D.G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[11] A. Vedaldi and B. Fulkerson, "Vlfeat: An open and portable library of computer vision algorithms," in *Proceedings. ACM Conference on Multimedia*, 2010, pp. 1469–1472.

[12] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 309–314, 2004.

[13] G. McLachlan and T. Krishnan, *The EM algorithm and extensions*, vol. 382, John Wiley & Sons, 2007.