A Hierarchical Image Clustering Cosegmentation Framework

Edward Kim, Hongsheng Li, Xiaolei Huang
Department of Computer Science and Engineering
Lehigh University, PA 18015
{edk208,h.li,xih206}@lehigh.edu

Abstract

Given the knowledge that the same or similar objects appear in a set of images, our goal is to simultaneously segment that object from the set of images. To solve this problem, known as the cosegmentation problem, we present a method based upon hierarchical clustering. Our framework first eliminates intra-class heterogeneity in a dataset by clustering similar images together into smaller groups. Then, from each image, our method extracts multiple levels of segmentation and creates connections between regions (e.g. superpixel) across levels to establish intra-image multi-scale constraints. Next we take advantage of the information available from other images in our group. We design and present an efficient method to create inter-image relationships, e.g. connections between image regions from one image to all other images in an image cluster. Given the intra & inter-image connections, we perform a segmentation of the group of images into foreground and background regions. Finally, we compare our segmentation accuracy to several other state-of-the-art segmentation methods on standard datasets, and also demonstrate the robustness of our method on real world data.

1. Introduction

The segmentation of an image into foreground and background is a difficult problem in computer vision. Completely unsupervised methods of image segmentation typically rely on local image features, and thus lack the necessary contextual information to accurately separate an image into coherent regions. On the other hand, supervised methods of image segmentation can produce good results, but usually require large datasets of manually labeled training data. Not only is training data expensive and tedious to collect, most training sets are several orders of magnitude too small in comparison to human level recognition. Interactive, or semi-supervised, segmentation methods attempt to address the disparity between fully automatic and fully supervised image segmentation, but typically still require human intervention on each individual image. We seek to further eliminate this individual image dependency by performing automatic image segmentation on *weakly supervised data*. In the context of this paper we define a weakly supervised set of images as a collection of images that are known to contain instances of an object class. Knowing that a similar object appears in a set of images provides us with the necessary contextual knowledge to better segment an image into two classes, foreground and background.

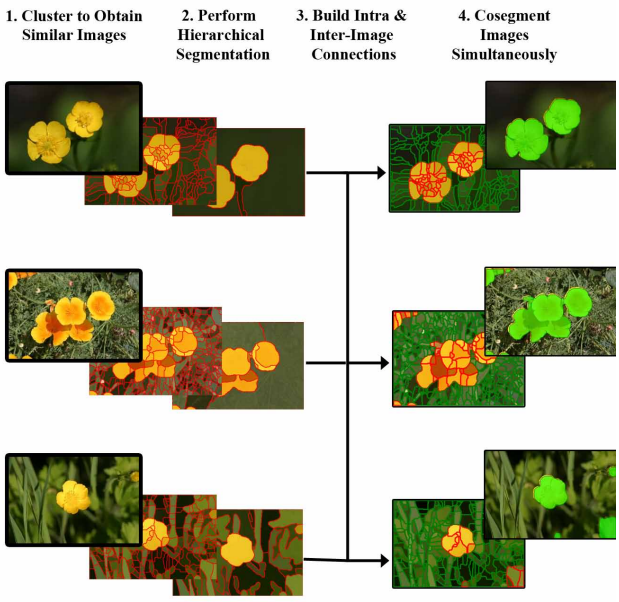

Figure 1. Overview of our framework. We first seek to reduce intra-class heterogeneity by obtaining small groups of similar images through image clustering. Then, we perform an automatic hierarchical segmentation on these images to acquire several layers with varying degrees of segmentation specificity. We can then create a graph with intra-image constraints and inter-image edges and solve for the optimal cut across all images in a group.

In recent literature, this problem has been identified as the *cosegmentation* problem [19]. However, the cosegmentation problem encompasses a large range of variability and

difficulty. For example, the problem can be formulated with a completely automatic segmentation of the same object among an image pair under angle or illumination changes [19, 23, 21]. On the other end of the spectrum, the problem can utilize training, and/or interactive methods to segment a large number of images with high intra-object variability [24, 2]. In our work, we address the difficult problem of *completely automatic* segmentation from weakly supervised data with high *intra-class heterogeneity*. An overview of our system can be seen in Figure 1. Specifically, our contributions are as follows: we develop a segmentation framework that clusters weakly supervised data into globally homogeneous groups. Within these groups, we perform a hierarchical segmentation and introduce our constrained affinity matrix that describes how to connect segmentation layers together. We show that the segmentation using our single image, multi-layered representation already improves upon current multiscale image approaches. On top of this framework, we propose a novel inter-image connection methodology that can efficiently handle large numbers of images. Our inter-image affinities are only calculated among the coarsest levels of segmentation, thus greatly reducing the number of computations needed to define the inter-image relationships. We build a graph and solve a normalized Laplacian matrix for the top eigenvectors using an efficient optimization method. Finally, we show our framework is adept at segmenting similar objects in a standard set of images as well as in larger, and more heterogeneous datasets that exist in the real world.

2. Background & Related Work

The problem of cosegmentation was first addressed by Rother et al. [19] using histogram matching and a modified MRF framework. The MRF includes an additional term that penalizes the energy formulation when the foreground region histograms differ. Since then, the topic has been explored across various degrees of foreground similarity for segmentation. Hochbaum and Singh [13] utilized an efficient MRF optimization which rewarded affinities rather than penalized differences in order to segment the same object with differing backgrounds. Sun et al. extracted the foreground from background using a MRF framework under a camera flash illumination change [21]. Ferrari et al. [11] used images of the same object to create a shape model of an object for detection and segmentation. In other related works, the cosegmentation approach has also been performed across image sequences [7, 15]. Slightly different variations to the MRF/graph cut [5] formulations and classification framework have been proposed to perform cosegmentation or object detection [23, 6, 8] where the only constraint is that the objects in the foreground are *similar*. Moving away from completely unsupervised methods, others have improved classification rates by incorporating an element of object training [24, 12] or interaction [2].

A different class of cosegmentation methods uses graph partitioning to solve the foreground/background partition. The benefits of these methods are that they find the global optimum cut, and do not require any prior models. These related works follow the popularity of spectral graph theory akin to normalized cuts [20] where the solution involves an eigen-decomposition of a graph Laplacian matrix. Yu and Shi [26] introduced how to incorporate a bias term and solve the system using an efficient optimization framework, while Cour et al. [9] utilized a multiscale graph bias to solve the multiscale normalized cut on a single image. One of the first proposals to use spectral cosegmentation was the work by Toshev et al. [22] where they perform the segmentation of *co-salient* regions. Later, Joulin et al. [14] again demonstrated the effectiveness of this model in a cosegmentation framework by utilizing a normalized Laplacian with a spatial consistency term.

Following these works, we seek to create an efficient spectral segmentation framework that is robust on single images as well as across large scale, heterogeneous weakly supervised datasets. Several works utilized superpixels, or larger coherent regions within images, to speed up the cosegmentation problem [14, 16], but remain solely in one superpixel dimension; they do not further take into account the benefits of a hierarchical or multiscale representation as shown by Cour et al. [9]. Further, effective and efficient inter-image connections are continuing research problems [22] with cosegmentation. In our work we directly address these issues in a hierarchical clustering framework. We are able to intuitively encapsulate local affinities within an image, constraints across different hierarchical segmentations, and global affinities efficiently connected across images.

We will utilize the normalized cut criterion to solve for an optimal partitioning of an image into foreground and background regions. We construct a graph $G = (V, E, W)$, with graph nodes V, graph edge E, and affinity $W(i, j)$ which measures the likelihood that node i and j belong to the same class. Let D be a diagonal matrix where $D(i, i) = \sum_j W(i, j)$. Let X be a $N \times 2$ partition matrix where $X \in \{0, 1\}^{N \times 2}$, and $X(i, c)$ be the indicator function that equals 1 if node $i \in V_c$ (i.e. belongs to partition c), and 0 otherwise. The 2-way normalized cuts criterion can be expressed as the optimization of X,

$$\begin{aligned} maximize \quad & \epsilon(X) = \frac{1}{2} \sum_{c=1}^{2} \frac{X_c^T W X_c}{X_c^T D X_c} \\ & s.t. X \in \{0, 1\}^{N \times 2} \\ & X 1_2 = 1_N \end{aligned} \tag{1}$$

Where 1_N is a $N \times 1$ vector of all 1's. This system can be relaxed into a constrained eigenvalue problem and solved by linear algebra as shown by Yu and Shi [26].

3. Methodology

To handle a large number of images and high variability within these images we first perform a series of preprocessing steps, including global image clustering, and hierarchical superpixel segmentation. After the preprocessing steps, we build a normalized Laplacian matrix, constrained by our superpixel hierarchy and weighted by both intra-image and inter-image connections. Finally, we solve for the first $K(=2)$ eigenvectors of our graph Laplacian matrix utilizing an efficient optimization method shown to be linear to the number of pixels (in our case superpixels).

3.1. Clustering for Intra-class Heterogeneity

Although the images in weakly supervised data belong to the same class, intra-class variability may be detrimental to the cosegmentation problem [24, 16]. In order to deal with large datasets with large intra-class variability, we first perform an image level clustering on the dataset. For each image $\mathcal{I}$ in the dataset, we extract three global image features, a pyramid of LAB colors, a pyramid of HOG features, and a histogram of SURF features.

Pyramid of LAB colors - The pyramid of color histogram features, PLAB, represents the various color regions present in an image. We convert the pixel colors into the perceptually uniform $L^*a^*b^*$ color space. Our PLAB descriptor is also able to represent local image color and its spatial layout [17]. For each channel of the color space, we extract 3 pyramid levels, with a 16 bin histogram from each region. A pyramid is constructed by splitting the image into rectangular regions, increasing the number of regions at each level. Thus, a single channel histogram consists of 336 bins, and our complete PLAB descriptor consists of 1008 bins.

Pyramid of HOG textures - Similar to our color features, we represent texture as a pyramid histogram of oriented gradients, or PHOG feature [4]. The PHOG descriptor represents local image shape and its spatial layout. If we use an 8 bin orientation histogram over 4 levels, the total vector size of our PHOG descriptor for each image is 680 bins.

Histogram of SURF features The SURF feature [3] (Speeded Up Robust Feature) is a scale and rotation invariant detector and descriptor. We detect and extract SURF features across an entire dataset. Using *k*-means, we vector quantize the SURF vectors into a codebook containing 1000 visual words.

With these three feature descriptions, we can perform an image level *k*-means clustering to split the dataset into several groups, $\mathcal{G}$. On a large dataset, we typically assign k in order to have 10-20 images per group. By performing cosegmentation on these smaller groups we can increase the accuracy of our final result.

3.2. Superpixel segmentation

Figure 2. A hierarchical superpixel segmentation of a bird image using gPb-owt-ucm into four layers. The bottom layer $l(=1)$ (left) is the most detailed; whereas, the top layer $l(=4)$ (right) is most coarse. Additionally, on the top layer, we visualize the SURF features present in the highlighted region.

For every image in the dataset, we perform a low level segmentation of our image into several hierarchical layers, $l = 1..\mathcal{L}$, where in our experiments we set the number of layers ($\mathcal{L} = 4$). Any hierarchical segmentation method can be used; however, we have found that the gPb-owt-ucm [1] method produces the best results. Using the gPb-owt-ucm segmentation, the bottom layer, $l = 1$, typically contains 300-500 superpixels, whereas the very top layer, $l = 4$, typically contains 5-15 regions, see Figure 2.

For each superpixel region, we extract 2 histogram features and 4 scalar features. These features are 32-bin LAB color histogram (one for each channel), 64 bin codebook histogram of SURF features, centroid x position, centroid y position, superpixel area, and superpixel eccentricity.

3.2.1 Intra-image Edge Affinity

Within each hierarchical segmentation layer, we define an edge affinity between neighboring superpixels. The similarity of superpixels is determined by comparing their corresponding LAB color histograms, weighted by the length of the shared border between superpixels. Mathematically speaking, we define the edge affinity as,

$$W(i,j) = \frac{\alpha(i,j)}{\sum_{k \in N_i} \alpha(i,k)} * e^{-||\chi^2(H_i,H_j)||^2/\sigma_H} \qquad (2)$$

Where $\alpha(i,j)$ represents the shared border between superpixels, i and j, H represents the 3 channel LAB color histogram of the superpixel, σ_H represents the variance of all distances between color histograms of neighboring superpixels, and N_i represents the neighbors of i. For the distance measure between two histogram-like feature vectors, h_X and h_Y, we use the χ^2 measure defined as,

$\chi^2(h_X, h_Y) = \frac{1}{2}\sum_{k=1}^{\mathcal{K}} \frac{[h_X(k)-h_Y(k)]^2}{h_X(k)+h_Y(k)}$, where $\mathcal{K}$ is the total number of bins present in the feature vectors ($= 32$ for each LAB channel).

To obtain the shared border length between two superpixels, we first represent the superpixel regions as a connected component matrix, $\mathcal{C}$, with each superpixel region having a distinct superpixel id. Then, we can compute the gray level co-occurance matrix (GLCM) over the matrix of size $n \times m$, where the n is equal to the image height (in pixels) and m is equal to the image width. The GLCM value, and equivalently the shared border is computed by,

$$\alpha(i,j) = \sum_{p=1}^{n}\sum_{q=1}^{m} \begin{cases} 1, & \text{if } \mathcal{C}_{(p,q)} = i \text{ and } \mathcal{C}_{(p+1,q+1)} = j \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

To incorporate the various segmentation layers of an image into our system, we utilize a multiscale normalized cuts approach [9] and augment the partitioning matrix in Equation 1 to become $X_l \in \{0,1\}^{N_l \times K}$ at hierarchical layer l, $X_l(i,c) = 1$ if the superpixel node $i \in V_c$. The hierarchical partitioning matrix X and affinity matrix W are defined as,

$$X = \begin{pmatrix} X_1 \\ \vdots \\ X_{\mathcal{L}} \end{pmatrix}, W = \begin{pmatrix} W_1 & & 0 \\ & \ddots & \\ 0 & & W_{\mathcal{L}} \end{pmatrix}, \tag{4}$$

We build a constraint such that the smaller superpixels in a lower segmentation layer should have some sort of class consistency with the encompassing superpixel in the higher layer. Therefore, we define a child/parent relationship where the child of superpixel i is defined as $d \in \mathcal{D}_i$, where the area of d is completely enclosed by the area of i, and d and i exist in neighboring layers, i.e. $l_d = l_i - 1$. This definition assumes that the low level segmentation method to create the superpixels has the property that any superpixel in the lower segmentation layer has one and only one parent. In other words, the outer superpixel borders of D_i equal the borders of i.

We can now define the relationship between two layers by measuring the fractional area of a child node in relation to its parent area, using constraint matrix, $C_{l,l+1}$ of size $N_{l+1} \times N_l$, defined as,

$$C_{l,l+1}(d,i) = \begin{cases} \mathcal{A}_d/\mathcal{A}_i & \text{if } d \in \mathcal{D}_i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Where the area of superpixel d and i are represented by $\mathcal{A}_d, \mathcal{A}_i$, respectively, and the constraint across hierarchical layers in image $\mathcal{I}$ is defined as,

$$C = \begin{pmatrix} C_{1,2} & -I_2 & & 0 \\ & \ddots & \ddots & \\ 0 & & C_{\mathcal{L}-1,\mathcal{L}} & -I_{\mathcal{L}} \end{pmatrix}, \quad \textit{s.t. } CX = 0 \tag{6}$$

We will see in the following sections how the constraint matrix can be used to project our result into a feasible solution space.

3.2.2 Inter-image Edge Affinity

Just as we are able to simultaneously segment multiple layers within an image, we seek to simultaneously segment a set of images. Let $\mathcal{I}^{1..\mathcal{G}}$ denote the images in a cluster group. We again augment our weight matrix, W matrix, by putting the $W^{1..\mathcal{G}}$'s from each image on the diagonal, and augment our constraint matrix, C, in the same way. Similarly, we extend our partitioning matrix to encompass all the superpixels from all the hierarchical layers within group, $\mathcal{G}$. Our new formation becomes,

$$X = \begin{pmatrix} X^1 \\ \vdots \\ X^{\mathcal{G}} \end{pmatrix}, W = \begin{pmatrix} W^1 & & R \\ & \ddots & \\ R^T & & W^{\mathcal{G}} \end{pmatrix}, C = \begin{pmatrix} C^1 & & 0 \\ & \ddots & \\ 0 & & C^{\mathcal{G}} \end{pmatrix}, \tag{7}$$

Where R is a sparse matrix that describes our inter-image relationships. This final representation can be seen in Figure 3.

If we were to augment our weight matrix and solve the normalized cut, without including the constraint matrix, all the superpixels in our image group would be considered independently (similar to the approach by [14]). This is because we lose the intra-image connections across layers. However, simply adding the hierarchical layer constraints matrix results in a trivial solution where the separation of classes occurs at image boundaries, rather than within images.

In order to propagate the cut inside the individual images, connections must be made between images. Unfortunately, there are no explicit relationships that exist between images as we saw before with the hierarchical layer constraint. Assuming a dataset containing images of $n \times m$ pixels, the number of possible connections between every pixel in each images becomes $O(n^2m^2)$, which is impractical (and in most cases, nonsensical) to implement. Thus, we exploit our layered segmentation hierarchy to create efficient inter-image weight connections. At the $l(=4)$ level, where the typical number of regions ranges between 5-15 total regions, we consider a fully connected graph, where each large region is connected to every other $l(=4)$ level within our dataset. The weights of these edges are computed by the region affinities defined by,

$$R(i,j) = \beta(i,j) * e^{-\lambda_1 \frac{||\chi^2(H_i,H_j)||^2}{\sigma_H} - \lambda_2 \frac{||\chi^2(\mathcal{S}_i,\mathcal{S}_j)||^2}{\sigma_{\mathcal{S}}} - \lambda_3 F(i,j)} \tag{8}$$

Where $\mathcal{S}$ represents a 128 bin histogram containing the frequency of SURF responses in our codebook. The three

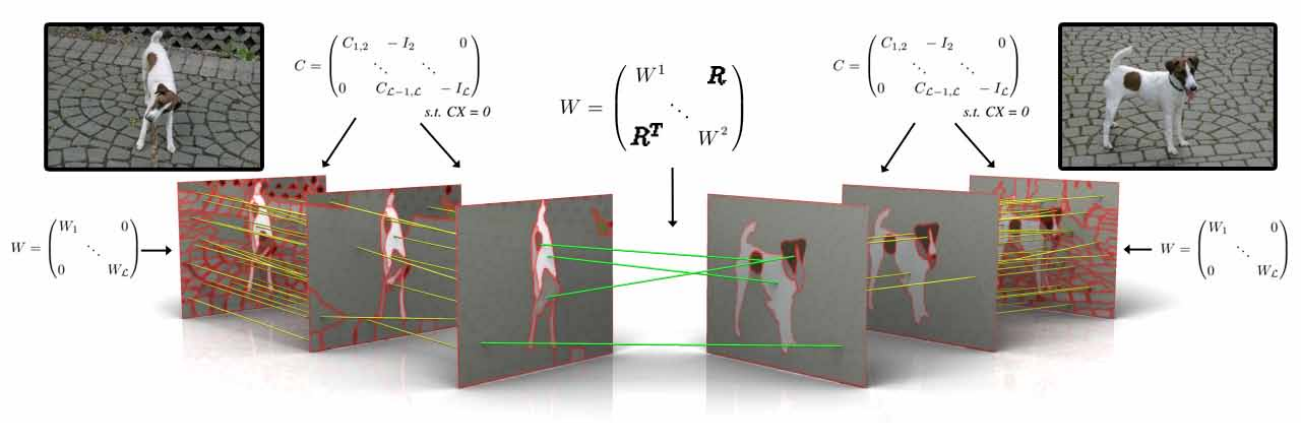

Figure 3. An illustration of our constructed graph between two images. An image is hierarchically segmented into a number of layers where the *intra-image* affinities, W, are defined between neighboring superpixels, weighted by the length of the shared border shown in red. The hierarchical constraints between layer segmentations are illustrated by the yellow connections. These connections are defined in our constraint matrix, C. The *inter-image* affinities, R, are made between images at their coarsest level of segmentation. A fully connected graph is considered and then the number of edges are trimmed, as illustrated in green. (Note: yellow and green connections are visualizations and not the actual edges).

scalar value affinities e.g. x,y centroid positions and eccentricity, are contained in a vector F, where the difference is measured by euclidean distance,

$$F(i,j) = ||F_i - F_j||^2/\sigma_F \tag{9}$$

In our experiments, we set the $\lambda_1, \lambda_2, \lambda_3 = 1$. For the edge weight strength between two regions, we define $\beta(i,j)$ as the symmetric strength determined by the total affinity weight of image $\mathcal{I}$ and image $\mathcal{J}$, divided by the number of edge connections between the two images, i.e.,

$$\beta(i,j) = \beta(\mathcal{I},\mathcal{J}) = \begin{cases} \frac{\sum W^{\mathcal{I}} + \sum W^{\mathcal{J}}}{\mathcal{I}_{N_4} \times \mathcal{J}_{N_4}}, & \text{if } \beta(\mathcal{I},\mathcal{J}) > t \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

Where $i \in \mathcal{I}$ and $j \in \mathcal{J}$. Recall that N_4 indicates the total number of superpixels in $\mathcal{L} = 4$. Additionally, t is an adaptive threshold on $\beta(\mathcal{I},\mathcal{J})$ that trims the total number of connections between images to maintain only the top matching cases ($\sim$40%). This threshold has the benefit of maintaining smoothness in our final resulting segmentation. If too many opposite labeled neighborhood connections are made, superpixel islands have a tendency to appear.

3.3. Graph Cosegmentation

Finally, we can solve for our binary partition matrix X. Let $P = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ be the normalized affinity matrix. As we can see, all of the images in our group are contained in our affinity matrix; therefore, our system has the benefit of computing an image segmentation across all images simultaneously. We incorporate our intra-image constraint C by creating Q as a projector onto the solution space,

$$Q = I - D^{-\frac{1}{2}} C^T (C D^{-1} C^T)^{-1} C D^{-\frac{1}{2}} \tag{11}$$

We solve the matrix QPQ for the first $K(=2)$ eigenvectors $\mathcal{V}$ as described by the general Rayleigh-Ritz Theorem in [26]. Because $\mathcal{V}$ is continuous, we normalize $\mathcal{V}$ and search for the best rotation to a discrete solution X. Thus, our final solution satisfies the binary and exclusion constraints in equation 1. Also, the solution to QPQ is shown to be linear to the number of superpixels if Q is expanded to a chain of smaller matrix-vector operations [9]. Thus, we can efficiently compute the cosegmentation over large groups of images. We note that with our formulation, it is trivial to extend our formulation to find more than 2 graph partitions, e.g. $K > 2$.

4. Experiments and Results

We perform experiments on two datasets, the MSRC [25] dataset and ImageNet [10]. The MSRC dataset contains 591 images from 23 object classes. Additionally, the pixel level ground truth labeling is given. The ImageNet database is an enormous database that contains the nouns of the WordNet hierarchy. As of November 2011, ImageNet contains over 14 million images and 21,000 synsets. Because ground truth is not available for ImageNet, we will utilize the bounding boxes provided by ImageNet users as our ground truth labeling.

For our quantitative results, we measure the *overlap* of our segmentation with ground truth defined as $\frac{|R_1 \cap R_2|}{|R_1 \cup R_2|}$. Also known as the Jaccard coefficient, this quantitative evaluation metric is used by the PASCAL VOC community and commonly used for other region comparisons.

4.1. MSRC dataset

For the MSRC dataset, we ran experiments on 13 classes of images containing 30 images each. For the experiments, we randomly select 100 pairs of images within a class and perform a cosegmentation on these pairs. We report the mean overlap score in Table 1 with comparisons against 2 state-of-the-art methods in cosegmentation [14, 16]. For two of the methods [14, 16], we ran identical experiments to ours with their publicly available code. In both cases, we used their default parameters, but in the code of [16], the default number of segments was k=4. To come up with a final foreground, background segmentation using this method, we chose the best combination of regions as the final segmentation for their result. Also, for this method, we use Turbopixels [18] to generate the underlying superpixel representation as this was the default method included with their code. For [14], there was no specified default superpixel code, so for this method, we utilized the same superpixel code as used by our method, gPb-owt-ucm [1].

In addition to several cosegmentation methods in Table 1, we also report the results of our method using only a single image. As a baseline comparison for our single image implementation, we also compare with MNcut [9]. For the single image algorithms, we set k=2 and choose the region that provides the best accuracy to the ground truth.

As shown in Table 1, our multi-image cosegmentation method consistently scores among the top performers in nearly all categories. Additionally, our single image segmentation is an improvement from the traditional MNcut algorithm in all categories but one. Of particular interest is the *signs* category where both single image methods (our single image method and MNcut) are more accurate than all three cosegmentation methods. Here, it appears that the *sign* images stand out well on their own, but when coupled with another random *sign* image, the cosegmentation accuracy drops. A feasible explanation may be that signs were designed in the real world to, 1. stand out from their surroundings to be easily seen, and 2. not look like other signs so they can be quickly and easily distinguishable from each other. From our quantitative evidence, it appears that these *signs* are well designed for real world use.

For our qualitative results, we provide a visual comparison of our method in Figure 4. To obtain these results, we cluster the images in the dataset, using the method described in Section 3.1, such that the number of images in each cluster ranges from 4-8 images. Given a group, G, we cosegment the images in that group, and visualize the results of one of the images in that group. All cosegmentation methods are given the same images as their input. Similar to our quantitative results, we also show our single image results with a comparison to multiscale normalized cuts.

Table 1. Results on several classes from the MSRC dataset. We compare the overlap score of our method to three other state of the art algorithms for automatic image segmentation.

	Multi image	Single image	CoSand [16]	DClust [14]	MNcut [9]
Bike	42.1	39.5	**42.3**	42.0	40.8
Bird	**32.8**	29.5	31.7	30.3	28.1
Car	54.4	49.5	56.2	**61.6**	43.5
Cat	**44.6**	40.3	41.7	40.9	37.6
Chair	**42.9**	41.0	39.9	42.3	33.2
Cow	**52.3**	50.8	40.1	35.5	38.9
Dog	42.1	38.9	41.9	**45.3**	32.2
Face	37.6	35.5	36.7	**39.4**	33.9
Flower	**58.9**	53.7	53.8	43.1	45.1
Plane	32.7	29.5	**35.1**	26.5	27.3
Sheep	**62.1**	59.1	43.8	36.1	41.7
Sign	53.3	**60.1**	51.7	52.6	58.8
Tree	61.2	58.5	58.9	**62.0**	47.3

4.2. ImageNet dataset

To evaluate our framework on more diverse, large scale datasets, we chose to test on ImageNet. Because the images collected by ImageNet are categorized into synsets, we are able to view synset groups as weakly supervised datasets. For our experiments, we randomly selected six synsets that contained over 1,000 images each. We randomly select a subset of 200 images from each synset where ground truth bounding boxes are available and cluster them into homogeneous groups that typically contain 10-20 images. Our clustering step has both a computational and accuracy benefit. Computationally, it reduces the number of images and inter-image connections in our cosegmentation. Without this significant reduction, we would need to perform more aggressive pruning in Equation 10, or produce a more coarse segmentation at the highest level to improve scalability. In terms of accuracy, the clustering step typically improves the overlap score by 1-5%. A more significant accuracy improvement is not observed because our inter-image affinities already drop weak connections that exist between dissimilar images.

We present our multi image and single image results in Table 2, and compare with the results of CoSand [16] and MNcut. For large databases such as these, we perform the superpixel segmentations and feature extraction steps offline. Additionally, we can save more time by precomputing the W matrix for each image. Thus, the only computation that is variable in our cosegmentation are the inter-image edges, that will change with different groups or numbers of images. Typically this precomputation step takes 10-15 minutes per group of 10 images on an Intel Xeon 2.53 GHz processor with 24 GBs of RAM. We store these features in a MySQL database for fast indexing and retrieval. Af-

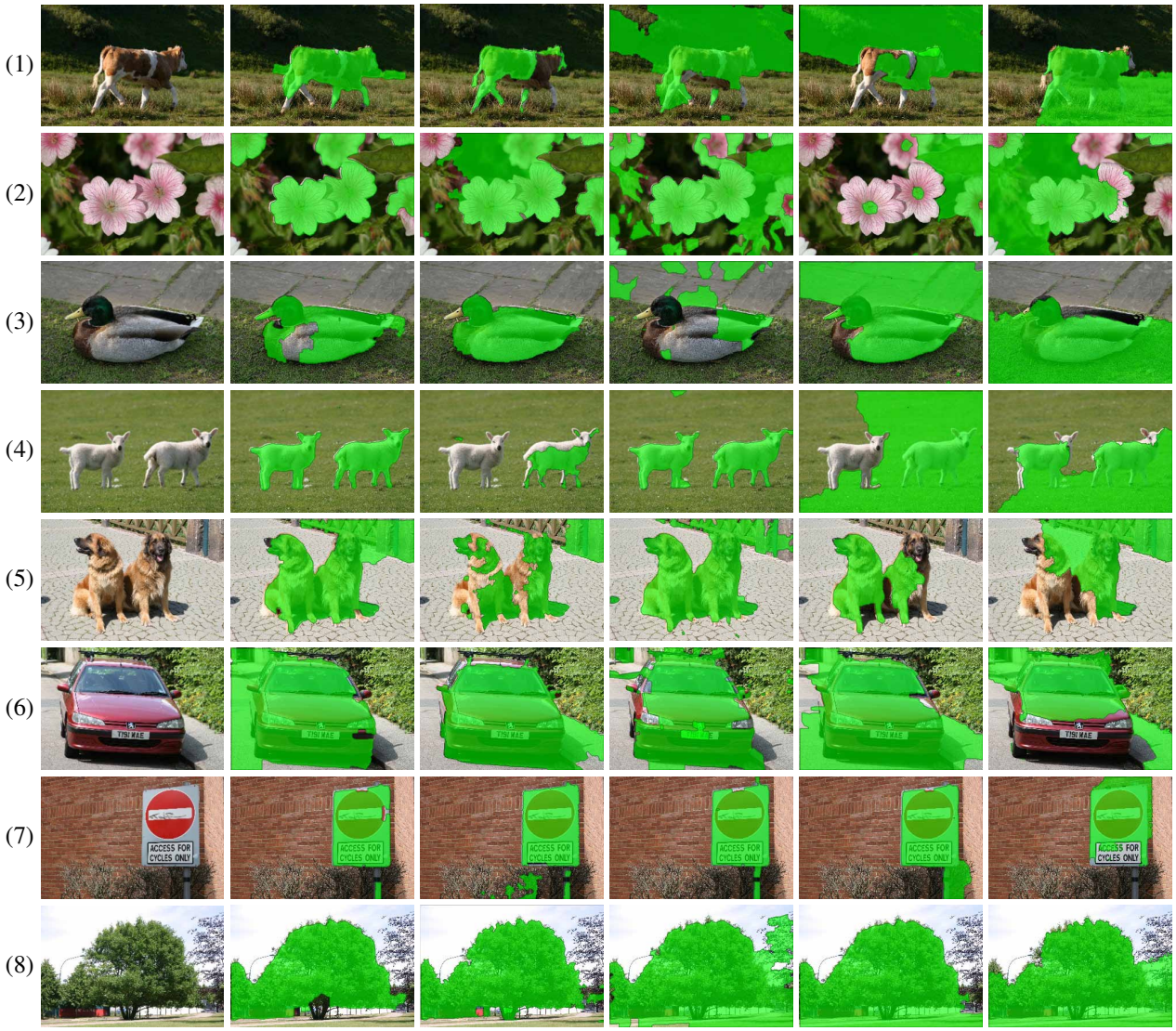

Figure 4. Original image shown in column (a). Column (b), (c), and (d) are results from multi image cosegmentation methods, where (b) is our method, (c) is from [16] and (d) is [14]. For these results, we cluster the MSRC dataset into $k = 5$ groups, resulting in an average of 6 images per group. These groups are cosegmented, and a random image from one of the groups is displayed here. In (e) & (f) we show results from single image spectral decompositions where our result is in (e) and the baseline algorithm, MNcut [9], is shown in (f).

ter performing these steps, the cosegmentation of a group of 10 clustered images can be performed in 30-60 seconds. Several examples can be seen in Figure 5.

5. Conclusion

We presented an efficient method for image cosegmentation. We introduced a hierarchical framework that effectively captures local image information from a single image. We represent these intra-image connections in an affinity matrix constrained by superpixel parent/child relationships. In fact, one could even think of our intra-image layer representation as a single image cosegmentation method (each layer as its own image). Furthermore, we proposed novel inter-image connections between images in a small cluster that exploit our hierarchical framework. From our quantitative and qualitative results, we show that our multi image cosegmentation method is effective and robust for large datasets with intra-class heterogeneity.

Table 2. Results on several synsets from the ImageNet dataset. We compare the overlap score of our method to CoSand and MNcut.

	Multi image	CoSand [16]	Single image	MNcut [9]
cannon[1]	42.9	**43.1**	36.7	33.2
chihuahua[2]	**50.3**	49.0	46.2	45.3
hammer[3]	**45.9**	42.8	43.2	41.1
pineapple[4]	**52.5**	48.6	42.7	40.3
stingray[5]	46.1	**52.9**	44.2	40.5
tennis ball[6]	**48.2**	44.6	38.2	35.1

synsets: [1]*n02950826,* [2]*n02085620,* [3]*n03481172,* [4]*n07753275,* [5]*n01498041,* [6]*n04409515*

(a) (b) (c)

Figure 5. (a) Original images from ImageNet with superpixel segmentations (b) and final results (c). These pairs of images belonged to the same clustered group.

References

[1] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. *CVPR*, pages 2294–2301, 2009. 3, 6

[2] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. icoseg: Interactive co-segmentation with intelligent scribble guidance. In *CVPR*, pages 3169–3176, 2010. 2

[3] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *ECCV*, pages 404–417, 2006. 3

[4] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. *CIVR*, pages 401–408, 2007. 3

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE PAMI*, 23(11):1222–1239, 2001. 2

[6] Y. Chai, V. Lempitsky, and A. Zisserman. Bicos: A bi-level co-segmentation method for image classification. *ICCV*, pages 2579–2586, 2011. 2

[7] D. Cheng and M. Figueiredo. Cosegmentation for image sequences. In *Image Analysis and Processing*, pages 635–640, 2007. 2

[8] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *CVPR*, pages 1–8, 2007. 2

[9] T. Cour, F. Benezit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *CVPR*, pages 1124–1131, 2005. 2, 4, 5, 6, 7, 8

[10] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. 5

[11] V. Ferrari, F. Jurie, and C. Schmid. From images to shape models for object detection. In *IJCV*, volume 87, pages 284–303, 2010. 2

[12] C. Gu, J. Lim, P. Arbelaez, and J. Malik. Recognition using regions. In *CVPR*, pages 1030–1037, 2009. 2

[13] D. Hochbaum and V. Singh. An efficient algorithm for co-segmentation. In *CVPR*, pages 269–276, 2009. 2

[14] A. Joulin, F. Bach, and J. Ponce. Discriminative clustering for image co-segmentation. In *CVPR*, pages 1943–1950, 2010. 2, 4, 6, 7

[15] H. Kang, M. Hebert, and T. Kanade. Discovering object instances from scenes of daily living. In *ICCV*, pages 762–769, 2011. 2

[16] G. Kim, E. Xing, L. Fei-Fei, and T. Kanade. Distributed cosegmentation via submodular optimization on anisotropic diffusion. *ICCV*, pages 169–176, 2011. 2, 3, 6, 7, 8

[17] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume 2, pages 2169–2178, 2006. 3

[18] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE PAMI*, 31(12):2290–2297, 2009. 6

[19] C. Rother, T. Minka, A. Blake, and V. Kolmogorov. Cosegmentation of image pairs by histogram matching-incorporating a global constraint into mrfs. In *CVPR*, pages 993–1000, 2006. 1, 2

[20] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE PAMI*, 22(8):888–905, 2000. 2

[21] J. Sun, S. Kang, Z. Xu, X. Tang, and H. Shum. Flash cut: Foreground extraction with flash and no-flash image pairs. In *CVPR*, pages 1–8, 2007. 2

[22] A. Toshev, J. Shi, and K. Daniilidis. Image matching via saliency region correspondences. In *CVPR*, pages 1–8, 2007. 2

[23] S. Vicente, V. Kolmogorov, and C. Rother. Cosegmentation revisited: models and optimization. *ECCV*, pages 465–479, 2010. 2

[24] S. Vicente, C. Rother, and V. Kolmogorov. Object cosegmentation. In *CVPR*, pages 2217–2224, 2011. 2, 3

[25] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *CVPR*, volume 2, pages 1800–1807, 2005. 5

[26] S. Yu and J. Shi. Segmentation given partial grouping constraints. *IEEE PAMI*, 26(2):173–183, 2004. 2, 5