Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Yuning Chai

Recognition between a Large Number of Flower Species

Master Thesis

Winter 2010/2011

‘

Supervisors: Dr. Victor Lempitsky
Prof. Dr. Andrew Zisserman

Abstract

The goal of this thesis is to design and build a system which automatically classifies an image of a flower for hundreds of flower species. The classification should be done within a reasonable time so that it is usable for a real-time computer vision application.

The classification performance of the system is improved if only the flower region (foreground) of the image is considered, and the background region is ignored. To this end we present a superpixel-based flower foreground segmentation method, which has a processing speed 10 times faster than state-of-art flower segmentation algorithms, while maintaining the segmentation quality as evaluated on the 849 images of the Oxford Flower 17 Data Set.

For the actual classification task, we use a standard linear support-vector-machine in combination with the Bag-of-Words representation for the features. We investigate several different descriptors, kernels and heuristics. By combining our new segmentation and improved recognition methods, we achieve a class-average recognition accuracy of 81.4% over the state-of-art 76.3% on the Oxford Flower 102 Data Set.

In the final part of the thesis, we propose a simple, scalable and unsupervised co-segmentation method (to simultaneously segment a set of related images into foreground/background) called BiCoS. This is able to automatically produce training segmentations for our superpixel-based foreground segmentation. The algorithm outperforms several recent co-segmentation methods on both Oxford Flower data sets. It also gives competitive results on other benchmark data sets, such as the Weizmann horses and Caltech-101 categories.

Acknowledgements

This project is completed during my stay in the Visual Geometry Group (VGG) at the University of Oxford. Many people have contributed in the success of the my project, and I like to thank in particular:

Prof. Andrew Zisserman for the continuous advices and encouragements and as well the kind hospitality.

Dr. Victor Lempitsky for the valuable knowledge inputs throughout the entire project.

Prof. Luc Van Gool who supported my stay in Oxford and made this exchange possible in the first place.

I would like to thank all members of the VGG, in particular Dr. Andrea Vedaldi for the kind support with his great software VLFeat and Varun Gulshan who helped me a lot setting up a new home in Oxford.

Disclaimer

Although all experiments are done by myself, Dr. Victor Lempitsky greatly contributes in both the direction of the research and the writing of some reports. Such, some text in Sec. 2.3 and Chap. 5 is taken from our joint report with minor changes.

Contents

Chapter 1

Introduction

Recognition is one of the major applications in computer vision. The majority of the research effort has been on the classification between different categories, such as between cars and airplanes as posed in the PASCAL [15] and Caltech data sets [27]. Many recent approaches have been successful in this area. However, we aim to use computer vision techniques to distinguish between visually similar objects. In particular, we investigate the possibility to separate flower species from each other.

Flowers from different species may look very similar both in shape or color *inter-class similarity*, e.g. lotus and water lily as shown in Fig. 1.1, and flowers from the same species may look different *intra-class dissimilarity*, e.g. Sword Lily as shown in Fig. 1.2. Together, these facts result in a very interesting and challenging computer vision problem. But besides our ambition to tackle with difficult problems, there are also application-related interests in solving it. Given an image of a flower, a non-professional person cannot tell which species that flower belongs to due to lack of flora-specific knowledge. With only the image and no other information, there is no way he or she can obtain further details about the flower unless going to a botanist. Our system can help people interested in flora to identify the flowers in order to get further information about their species. This idea can be easily extended to a more general vision of a system which takes images and provides additional information about it, like a Wikipedia using image instead of text search.

We note a very extensive work on flower species recognition has been done by Nilsback [32, 33, 34, 35, 36]. This thesis can be considered as an extension and improvement over her work.

A state-of-art recognition system often takes the whole input image and extracts features either

(a) Petunia (b) Silverbush (c) Water Lily (d) Lotus

(e) Globe Thistle (f) Artichoke (g) Cyclamen (h) Magnolia

Figure 1.1: Visually similar flower species. As shown in pairs, images from different flower species may even look indistinguishable for non-trained human eyes.

in a dense (using grid points) or sparse (using interest points) fashion. We note that the difficulty of our problem, the high similarity between flower species, implies a high similarity in the feature space as well, which lower the classification performance. We believe that this similarity can be reduced if we first localize the flower inside the image and only extract features from the flower and discard the background clutter. This idea is reflected as an additional foreground-background segmentation stage in our approach.

The use of foreground-background segmentation in a visual classification system is a controversial issue in modern computer vision. On the one hand, the background contains information that is irrelevant in separating one class from others. This information may be converted into noise during the feature extraction and causes problem in the end classification. Several studies have reported improved classification/detection performance once accurate segmentation masks are given [30]. However on the other hand, the background may contain information that is unique for a specific class, in this case referred to as *context*, which can be exploited to increase the recognition performance. In our case, due to the nature of our problem as stated above, we believe that discarding the background tips the balance to our favor. We decided for a recognition system with help of foreground segmentation, as shown in Fig. 1.3 as an example of Daffodil.

The first contribution of this thesis is therefore a fast foreground-background segmentation

Figure 1.2: Visually dissimilar flowers within the same species as shown in the example of sword lily. Sword lilies have a high variation in both color and shape, which makes the recognition problem very challenging.

pipeline, since our ultimate goal is to provide a real-time system for flower recognition. The proposed algorithm consists of three parts. We first divide the input image into smaller subregions (the *superpixels*), then we classify each superpixel as foreground or background using a discriminative classifier. Finally, we refine the result using GrabCut [38]. This new segmentation method is roughly 10 times faster than the state-of-art model-based approach by Nilsback [36], while maintaining the same segmentation accuracy in our overlap measure.

After the segmentation, we can use common classifier for the flower species recognition task. A well-acknowledged choice is a multi-label support-vector machine (SVM). However, there is a large play-room for feature selection, kernelization and the choice over linear or non-linear SVM. After plenty of experiments, we decided for a linear SVM that takes kernelized feature vectors. Our choice of features consists of the color inLab space and various SIFT descriptors that are extracted only from the foreground segmentations given by the previous stag. The features of each descriptor are then vector-quantized, linearly pooled to create a single feature vector of that descriptor of the input image. By finally concatenating them, we get the final feature representation of that image.

We show that our new feature selection, kernelization and the choice of SVM improves the recognition accuracy over the state-of-art work by Nilsback [32] by 5%, which constitutes to the second contribution of this thesis.

The human effort in the proposed pipeline is limited mainly to the training of the discriminative

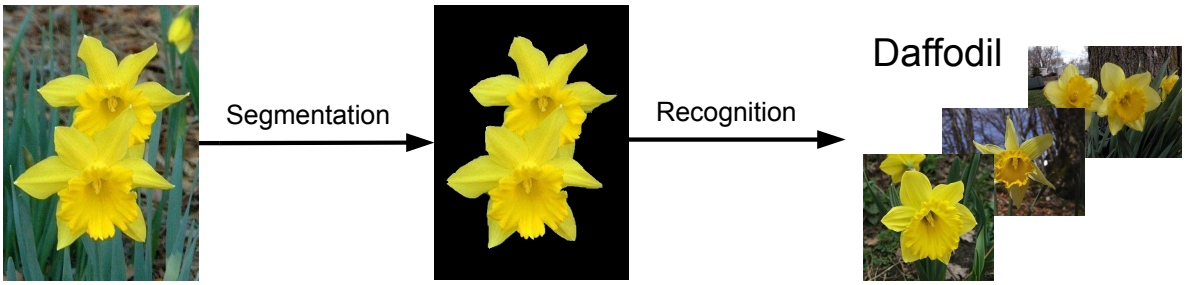


Figure 1.3: Two-stage approach of our system. The input image is first segmented, and then classified.

classifiers (*supervised learning*), one during the segmentation and another one during the recognition. Obviously, while annotating ground truth for recognition is easy, providing ground truth for segmentating foregrounds can be a long and painful process, even with proper tools like graphics tablets. In order to avoid this process, we propose BiCoS - Bi-Level Co-Segmentation, which is a very scalable, unsupervised learning algorithm to provide ground truth segmentation for the segmentation step. The high scalability is achieved by alternating between two levels of representation: at the bottom level, it treats every image separately and segments using the well-known GrabCut on RGB values, whereas at the top level, a discriminative classification is performed on high-dimensional descriptors of superpixels. A work flow of BiCoS is demonstrated in Fig. 1.4. Experimental results have shown that BiCoS outperforms state-of-art unsupervised and supervised algorithm on the Oxford Flower data sets, and also yields competitive results on other benchmark data sets. For our purpose, we run BiCoS on each flower species individually in order to obtain ground truth segmentations for the supervised learning.

Until now, the assumption has been that a better foreground segmentation according to an overlap ratio with hand-annotated ground truth will improve the final recognition accuracy. Although this assumption is true for the most time in our data set, we want to further enforce this assumption. Instead of finding individual classifier hyperplanes for each flower species, we exploit the high scalability of BiCoS and take images from all species at the time, and train all hyperplanes together. This new assumption yields a multi-task modification of the BiCoS algorithm, BiCoS-MT. BiCoS and BiCoS-MT constitute to the third contribution of this thesis.

To finally summarize the contributions. We provide:

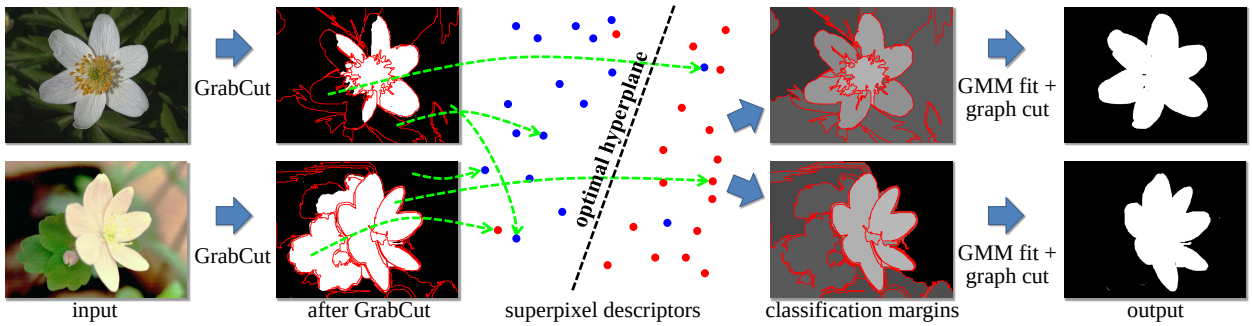


Figure 1.4: The diagram of our method (BiCoS) shown for 2 (out of 15) training images from one class of Oxford Flowers-17 dataset. BiCoS starts with the GrabCut at the pixel-level applied to each image independently. Each superpixel (superimposed) is then assigned to foreground or background and mapped to a descriptor space, where the optimal separating hyperplane is found. The pixels in each image are then assigned values according to the margins of the respective superpixels, and such soft segmentation map is then used to fit color Gaussian mixtures and apply pixel-level graph cut to each image.

- a fast flower foreground segmentation algorithm that is roughly 10 times faster than the state-of-art performance while achieving the save accuracy.
- a new combination of feature selection, kernelization, SVM choice and heuristics that improves the recognition performance by 5%.
- BiCoS and BiCoS-MT, two unsupervised co-segmentation algorithm that yields better results on the flower data sets than state-of-art unsupervised and supervised algorithms. They also achieve competitive results on benchmark co-segmentation data sets, such as the Weizmann horses.

The rest of this report is organized as follows: in Chap. 2, we first describe the state-of-art algorithms in foreground segmentation, recognition (in particular between *subordinate* categories) and co-segmentation, followed by main chapters for each of the three contributions in Chap. 3, 4, 5. We then finalize this report with a discussion and conclusion in Chap. 6.

Chapter 2

Related Work

Both segmentation and recognition are two major fields in computer vision, there are over hundreds of new publications every year. In the following overview of the related works, we try to cover the works that are mostly related to ours to our best knowledge. Co-segmentation is a relatively new topic. But the idea of unsupervised foreground segmentation has made it increasingly important.

2.1 Segmentation

Actually, both flower segmentation and recognition have been extensively research during the PhD study by Nilsback. While very good performance was achieved, the algorithm was too slow to be run in real-time.

Nilsback et al. [33] implemented a two-stage approach for flower segmentation, which propagates information between color and shape by alternating between a color dependent conditional random forest (CRF)-based graph-cut and a generic shape fitting. The iterative method requires an initialization, which is given by a CRF based on a generic color model trained from all flower species. Graph-cut is then applied to the image with a unary term from that color model and a binary term from the contrast between neighboring pixels. The generic flower shape model is then fitted to this initial segmentation in order to detect petals. The model selects petals which have a loose geometric consistency using an affine invariant Hough-like procedure. A second CRF is generated using a foreground color model extracted from the foreground from the last step and a

corresponding background color model. The iteration then continues between the shape model fitting and graph-cut. While the graph-cut on CRF can be computed very efficiently, fitting a generic shape model can be very costly, as the algorithm first tries to find the center of the flower and then the petals. Also, not every flower has petals, which may cause problems in few species.

Another approach focusing on flower segmentation is proposed by Saitoh et al. [41]. It is based on "Intelligent Scissors", which find the path between two points that minimizes a cost function dependant on image gradients. It is assumed that the object is in the center and such, the cost between two points on the flower is smaller than the one between one on and another one out of the flower. One can imagine that the assumption is not true in many cases and path finding can be computationally expensive.

Now let us leave flower for a moment and discuss about existing recognition methods that resembles to our proposed one:

The work by Hoiem et al. [21], which our proposed algorithm takes the idea of superpixels classification from, aims at multi-label image segmentation based on superpixels. The algorithm first divides the input image into multiple superpixel segmentation (multiple hypothesis), and then classify each pixel to either background or one of the geometric directions, as the algorithm is designed to find the geometric context of the object in the image. A generative classifier is applied on a very high dimensional feature vector consisting of the color, the texture, the location, the shape and the geometry.

However, superpixels are not perfect, as it is shown later. We are aware of graph-cut-based segmentation methods which have been proved to be very efficient and accuracy in segmentation. Ever since graph-cut was introduced to computer vision by Boykov and Jolly [7] , there have been several publications which focused on segmentation using graph-cut. We however focus on the GrabCut by Rother et al. [38], which is described in a more detailed way in Sec. 3.3.

A standard graph-cut algorithm converts the segmentation problem into an energy function:

$$E = \sum_i U_{x_i}(y_i) + w \sum_{i,j} V_(x_i, x_j) \tag{2.1}$$

where x_i is the i-th pixel and y_i the label of it. The so called unary term $U_{x_i}(y_i)$ is often defined as the probability of the pixel x_i having the label y_i. And the binary term is $V_(x_i, x_j)$ is the contrast between two pixels x_i and x_j. w is a scalar value for balancing. There is a fast solution (and implementation) by Boykov and Kolmogorov [8].

Another method using graph-cut for image segmentation is OBJCUT by Kumar et al. [25]. The binary term in graph-cut is extracted based on pairwise similarity between two pixels. A class-specific cues are trained using the so called layered pictorial structures. These are a small number of template for each part of the object. The unary term is extracted by fitting those templates onto the object that is to be segmented.

2.2 Recognition

The flower recognition pipeline by Nilsback et al. [35] uses a bag-of-words (BoW) approach followed by a non-linear support-vector-machine (SVM). The authors use a feature combination consisting of HSV, internal dense SIFT (scale-invariant feature transform [29]), boundary SIFT and histograms of gradients (HOG) [13] covering the color and the shape histograms of the flower, and an average recognition accuracy of 72.8% is achieved.

In Nilsback's later PhD thesis [32], she improved the score by roughly 4% by introducing geometric layout features. Flowers are first normalized in a sense of being rotated and stretched in order to have the same size and orientation. The idea of the layout feature is then to have multiple histograms instead of only one for one descriptor on one image.

The so called bag-of-words (BoW) is very popular in computer vision. One of the first works (if not the first) using it in conjunction with support-vector-machine for classification task, as it is done in our work, can be found in the publication by Csurka et al. [12]. The authors suggest a classification pipeline using vector-quantization and pooling of point-wise features from an image, which gives a vector representation of the image. The vector is then used in a SVM for classification.

Conventional BoW approaches completely discard the structural information within the image. The work by Lazebnik et al. [26] however introduces the spatial pyramids, which partitions the image into increasingly fine subregions. The BoW are then extracted locally from those subregions instead from the whole image. By concatenating those BoW representations, the final feature vector implicitly inherits some structural information from the pyramid. In our approach, we however did not consider the spatial pyramid for time efficiency.

Another work the success of this thesis thanks to is the work on kernel maps by Vedaldi and

Zisserman [46]. As training non-linear SVM can be very computationally costly, replacing it with a linear SVM using a kernel map on the feature vector can be beneficial. The authors derived explicit feature maps for several additive kernels and compared their performance on benchmark data sets.

2.3 Co-Segmentation

Co-segmentation of image collections has recently become a topic of active research [20, 23, 31, 39, 47]. Co-segmentation methods consider sets of images where the appearance of foreground and/or background share some similarities, and try to leverage these similarities to obtain accurate foreground/background segmentations either totally-unsupervised, or with a small amount of interactive supervision [4]. Most of the proposed co-segmentation methods (with the exception of [23]) assume close similarity of the foreground color histograms essentially requiring foreground objects to be the same through the image set.

Among the unsupervised co-segmentation algorithms, discriminative learning on superpixels is used by Joulin *et al.* [23]. The optimization framework of [23] simultaneously enforces spatial smoothness within each image as well as finding the foreground/background boundary in the superpixel space. Unlike [23], BiCoS decouples spatial smoothness enforcement and classification of superpixels, so that these two steps are performed consequently rather than simultaneously. We demonstrate experimentally that, despite the sub-optimality that such alternation-based approach might bring, BiCoS consistently attains higher segmentation accuracies, while being applicable to much bigger image sets than [23].

Alexe et al. [2] is another very recent work closely related to ours, as their system also uses superpixels to propagate information across multiple images. Such propagation is however achieved through binary-label Conditional Random Field (CRF) with unary potentials derived in a generative fashion as opposed to discriminative learning (SVM) used within BiCoS. Despite the use of explicit geometric modeling within [2], the experimental comparison revealed that BiCoS is able to achieve similar or higher segmentation accuracy for several viewpoint-constrained datasets, where their loose geometric model is appropriate.

Another co-segmentation work that adopts an alternation strategy similar to the bi-level architecture of BiCoS is Batra et al. [4]. Their appearance models are however based purely on color

(and hence are too limited for many scenarios). Their focus is also on interactive user supervision, rather than the fully unsupervised scenario used by most other co-segmentation works, as well as ours.

Chapter 3

Foreground Segmentation

Foreground segmentation describes the process to localize the flower in the input image and separate it from the background clutter. Although the model-based approach by Nilsback et al. [33] yields very good performance (93% in overlap ratio), its processing time over 1 minute is not well suited for real-time applications. Our research achieves to provide an approach that is as accurate as the one by Nilsback et al., but significantly faster.

GrabCut [38] is a very accurate and fast segmentation method. However, as part of an iterative method, it requires an initialization which is meant to be provided by the user in the original publication. We aim to provide the initialization automatically. Basically, for each pixel in the input image, we have to give a probability of it being foreground or background. It may be difficult and time inefficient to do the entire processing on the pixel-level. Such, we want to process on units that are larger than one single pixel, but small enough that pixels belong to that unit can be meaningfully assigned to the same probability. There are several recent approaches [14] which simply divide the whole image into grid elements and classify the grid points using a discriminative classifier. Instead, we use fast *superpixel* algorithms such as graph-based or min-shift-based, since natural images rarely possess rectangular shapes and the size of discriminative parts in a flower image may vary a lot. Once superpixels are extracted from the input image, we assign one probability to each superpixel according to a discriminative classifier.

To summarize, the proposed method first splits the original image into small segments (*the superpixels*), which are individually classified as foreground or background. In the final stage, the global information within the whole image is taken into account in order to create a smoother

segmentation result. The outputs of each stage are shown in Fig. 3.1.

(a) original image (b) superpixels (c) classification (d) final segmentation

Figure 3.1: Stages of foreground segmentation

3.1 Superpixel Segmentation

Every superpixel segmentation algorithms can be observed as either clustering-based or graph-based.

Clustering-based approaches, including Mean-Shift [11] and QuickShift [45], are originated from the idea by treating every pixel as a 5-tuple consisting of its 3-dimensional representation of color and its 2-dimensional position coordinates in the image, and applying clustering algorithm in the resulted 5-dimensional space. The different approaches mainly differentiate in the choice of the used clustering algorithm, its speed of convergence and resilience against outliers.

Graph-based methods, including Min-Cut and Normalized-Cut [43], form a graph using all pixels as its nodes. The nodes are then connected to each other with edge weights depending on the similarity of the two pixels (*affinity*). The graph is then cut to provide segments. Here, approaches differentiate in the definition of affinity, as well as the choice the cutting algorithm.

In this thesis, we chose one representative from each family (chosen by the availability of the code and processing speed). Graph-cut-based approach by Felzenszwalb and Huttenlocher [17] can segment images under 1 second. Although its counterpart, QuickShift by Vedaldi and Soatto, requires several seconds to process in general, the authors are in possession of a GPU version of the algorithm which is enough fast.

3.1.1 Graph-Based Image Segmentation

The algorithm by Felzenszwalb and Huttenlocher first blurs the input image with a strength defined by the parameter σ. Every pixel is first treated as an individual node in a graph, links are created between neighboring pixels and weights are assigned to each of the links. One then starts with the link with the smallest weight and start to merge pixels according to an energy function. Such, one goes through all links in a greedy way.

and based on an energy function and a threshold given by the input parameter k, nodes get merged with each other. The last input parameter θ defines a minimum area of each segment. Some example images are shown in Fig. 3.4.

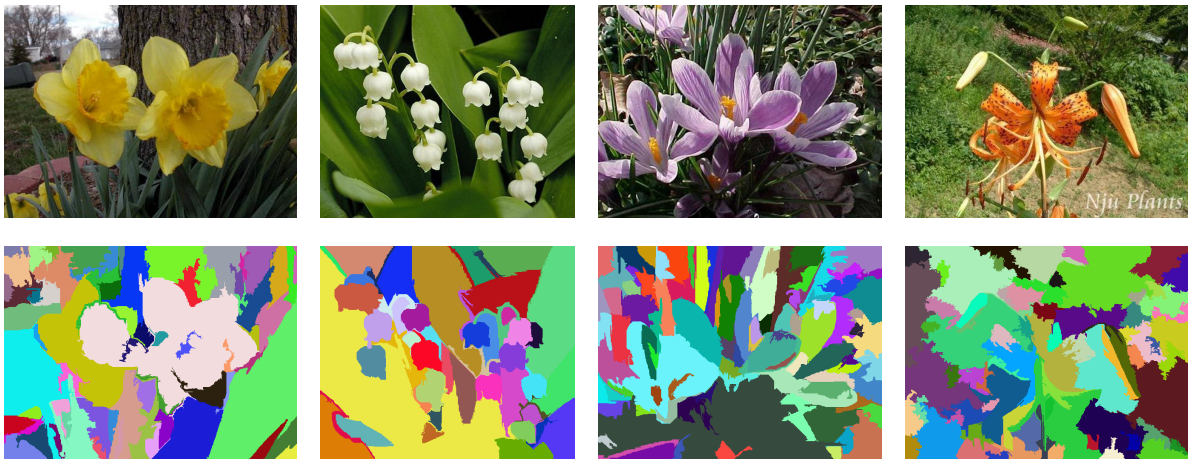


Figure 3.2: Examples using the graph-based segmentation method. Note the noisy superpixel boundaries.

3.1.2 Quick Shift

Quick shift is an image segmentation algorithm with strong relation to mean-shift and medoid-shift. Given a gray-level image, the goal of those algorithms is to convert the image to a tree. The segmentation is achieved by cutting off the tree's branches. In opposite to the original mean-shift, medoid-shift can create multiple trees and gain a speed improvement at the same time. However, it may also end up with under- and over-fragmentation. Quick shift on the one hand inherits the speed bonus from medoid-shift and avoids under- and over-fragmentation.

From the experience on graph-based segmentation, we could assume that around 50 to 100 segments per image is a good number to start. The quick shift implementation provided by [44] requires about 10 seconds to do so. However, since this algorithm has a GPU implementation which can reduce the processing time by a factor of 10, we tried this method nonetheless. Some quick shift examples are shown in Fig. 3.3.

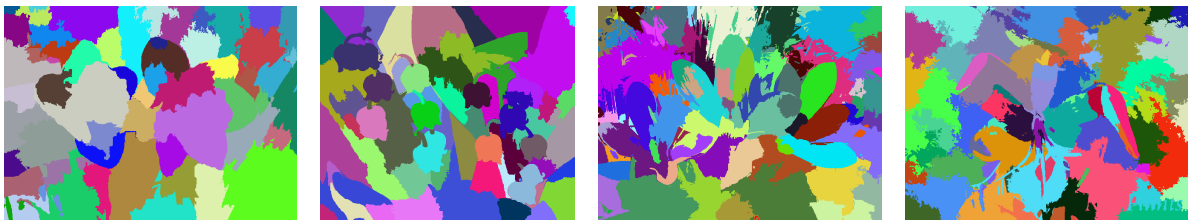


Figure 3.3: Examples using Quick Shift. Same images are used as in Fig. 3.4. Quick Shift produces much cleaner boundaries compared to the graph-based algorithm, however, it takes much longer processing time without GPU.

3.2 Superpixel Classification

3.2.1 Feature Selection

We like to train a discriminative classifier to give each superpixels a probability being foreground or background. As we apply the classifier to a binary problem, we do not want the feature vectors to be unnecessarily large. We considered the following features:

RGB Color Obviously, the color of a superpixel is an important feature, as flowers are seldom or even never to be grey, black or green. For each superpixel, we simply extract the mean and variance of the RGB value of all pixels in the that superpixel.

Location As the data sets constrain the flowers to be in the focus of the images, the location information of each superpixel becomes informative. We design two binary indicator and a feature vector to describing this important property. The former indicator tells whether the superpixel touches the image boundary, where as the latter describes it covering the image center. The feature vector is a 36-dimensional descriptor obtained by taking the image-size binary mask indicating pixels belonging to the superpixel and linearly down-sampling it to a 6×6 size.

Shape The shape is described in a similar manner as the 36-dimensional descriptor for location, where two changes are made. First, we only consider the bounding box instead of the whole image. And the binary mask is replaced with the boundary pixels of the binary mask. Again this descriptor is of 36 dimensions.

Area The area of the superpixel is covered by a single scalar number corresponding to the proportion of the superpixel area in the image.

As we like to keep the framework simples as we address it to a binary problem, the features are combined using simple concatenation.

3.2.2 Classifier Selection

Several common discriminative classifiers are used on the combined feature vector in our experiments:

k-Nearest-Neighbors (kNN) kNN is one of the most primitive classifiers. During training time, each feature vector from the training images is simply mapped onto the feature space. And for each test image, the "nearest k neighbors" from the training samples in the feature space are found and the test image takes the label of the majority of the neighbors.

Logistic Regression (LR) and Support-Vector-Machine (SVM) LR as well as the SVM (below) try to find the optimal hyperplane separating two categories. This problem can be described to a convex optimization problem:

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N} \ell(y_i \cdot \mathbf{w}^T\mathbf{x}_i) \longrightarrow \min_{\mathbf{w}}, \tag{3.1}$$

with the $\mathbf{w}$ being the representation of the hyperplane, N the set of training images, $\mathbf{x}_i$ being the feature vector from the i-th image, y_i its label and C the regularizer constant. The loss function is the logistic function $\ell(t) = \frac{1}{1+exp(t)}$ for LR and the hinge loss $\ell(t) = \max(0, 1-t)$ in case of SVM.

In order to use graph-cut-based algorithms in the final refinement stage, an output in form of a confidence image on the pixel level is more desirable than a binary classification. In case of LR and SVM, this confidence is given by the algorithm already. However, it becomes a more complicated in kNN. That is why we only considered 1NN, where the confidence of the pixel of

one segment being foreground is the ratio between the distance to the nearest background neighbor and the distance to the nearest foreground neighbor.

3.3 GrabCut

GrabCut by Rother et al. [38] is a powerful tool for interactive foreground-background segmentation. It combines two components: a binary-label random field defined on image pixels, and a generative foreground/background classifier that takes pixel RGB values as features. GrabCut proceeds as follows: starting from an initialization which is designed to be given by the user and simulated by the superpixel-wise classification in our approach, it alternates between **(1)** The (re)estimation of foreground and background probability densities via Gaussian mixtures given foreground/background pixel labels [5], and **(2)** The graph cut inference of foreground-background labels in the random field with unary terms defined according to the evidence from the Gaussian mixtures and the pairwise terms defined according to local image gradients cues [7].

The assumption behind GrabCut and in particular behind the step (1) in its alternation scheme, is that the RGB distributions of foreground and background are shared across the entire image (so-called *global color modeling*). As demonstrated in [38] and subsequent evaluations, GrabCut is capable of producing accurate segmentation even when initialized in a very crude way (e.g. a rectangular mask overlapping the true foreground).

3.4 Experiments

Data Set We use the Oxford Flower 17 Data Set [33] as the it provides human-annotated ground truth segmentations. The data set contains 17 different flowers with 80 images each, which results in totally 1360 images. However, the data set does not have ground truth segmentation for all of the images, but for only 849 of them, distributed unequally among all classes. Note that one class was left without any ground truth segmentations at all. There is also no data split provided for the segmentation, we initialized one by equally dividing the 849 images into training, validation and testing sets.

Segmentation Accuracy Measure Before we continue with the research steps, we would like

to introduce our accuracy measure P: Given A a binary mask of the foreground segmentation (foreground=1, background=0) and its ground truth binary mask $\tilde{A}$, P is defined as the average ratio of the intersection area of A and $\tilde{A}$ and their union:

$$P = mean(\frac{A \bigcap \tilde{A}}{A \bigcup \tilde{A}})$$

We will referred to P as **segmentation accuracy**, **overlap ratio** or **seg I** in the rest of this report, or simply accuracy within this section. The numbers are shown in percent.

Hardware The tests are processed on a laptop with Duo Core II 3.0 GHz CPU, where the implementation is a mixture of Matlab and C code. A typical processing time is between 3-4 seconds, whereas two significant times are worth mentioning. On the one hand the feature extraction, where avoiding shape features can save around 20% of total processing time. On the other hand is the number of iterations and GMM kernels during Grabcut.

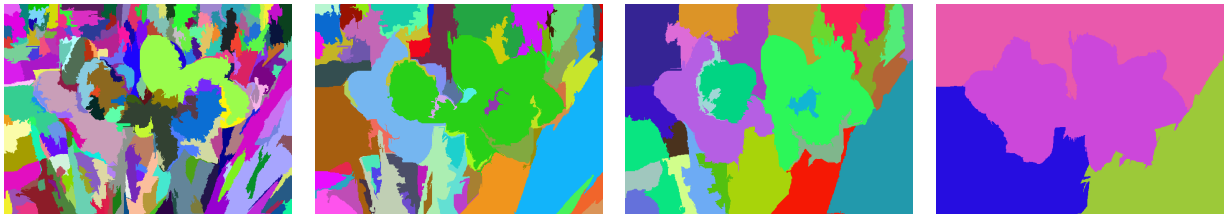
3.4.1 On Superpixel Segmentation

Superpixel segmentation is, like clustering, an ill-posed problem, and therefore it is hard to measure the performance of a particular method by only looking at the superpixels. Although it is possible to do so in a later stage, the high complexity would take too much time. Therefore, we assume two empirical criteria.

First, we are interested in the overlap ratio of a segmentation in the best case (ORBC). For each superpixel, we assign it to be foreground if it has more than 50% overlap with the ground truth foreground. However, this criterion has an obvious maximum if all pixels are assigned to individual segments, which gives 100% accuracy in this measure. To avoid this, the second criterion is the number of segments in the superpixel image, which should be kept as low as possible. With these two criteria, the superpixel segmentation becomes an multi-objective optimization problem.

The graph-based method by Felzenszwalb and Huttenlocher [17] is optimized using grid search with $4 \times 6 \times 7 = 148$ grid points tested on 85 images from the Oxford Flower 17 Data Set. Four segmentation results using different parameters are shown in Fig. 3.4.

A graph showing the average ORBC against the average number of segments is plotted in Fig. 3.5. Red points indicate non-Pareto property, meaning that it is worse in both measures than



(a) 202 segments (b) 67 segments (c) 29 segments (d) 4 segments

Figure 3.4: Graph-based segmentation on flowers.

(*dominated by*) at least one of other points. We discard those setups first in narrow down the parameter choices. Further experiments finally result in the following setup: the blurring parameter $\sigma = 1$, the merging threshold $k = 200$ and the minimum segment size $\theta = 640$. While experiments of the Felzenszwalb method show satisfying results, the algorithm seems to have problems along the boundaries between segments, as can be seen in Fig. 3.1 (b). The code provided by Felzenszwalb [17] needs under 1 second for each image.

A proper grid search is not possible due to the long processing time of Quick Shift. Therefore, we set the parameters so that the resulted segmentations have the same number of superpixels as the segmentations by the optimized graph-based segmentation approach. As being observed in Fig. 3.3, it produces much cleaner results than the graph-based approach. However, as the noisy boundaries are corrected by the final GrabCut algorithm, it turns out that the two segmentation algorithms give approximately the same overlap accuracy in the end. As Quick Shift takes longer time to process, we decided to leave it and only focus on the graph-based approach.

3.4.2 On Superpixel Classification

The power of discrimination of each element in the proposed feature vector is observed using the accumulated histograms $f(X)$ of positives (red) and negatives (blue):

$$f(X) = Pr[x < X] \tag{3.2}$$

Intuitively, the area between the two curves in accumulated histograms tells how discriminative a specific element in the feature vector is. Such histograms for each element of the discussed features are shown in Fig. 3.6, 3.7, 3.8 , 3.9 and 3.10.

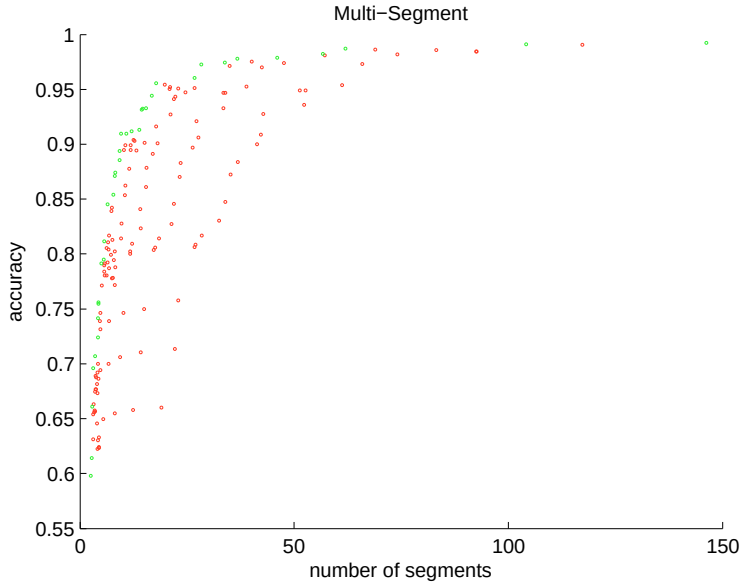
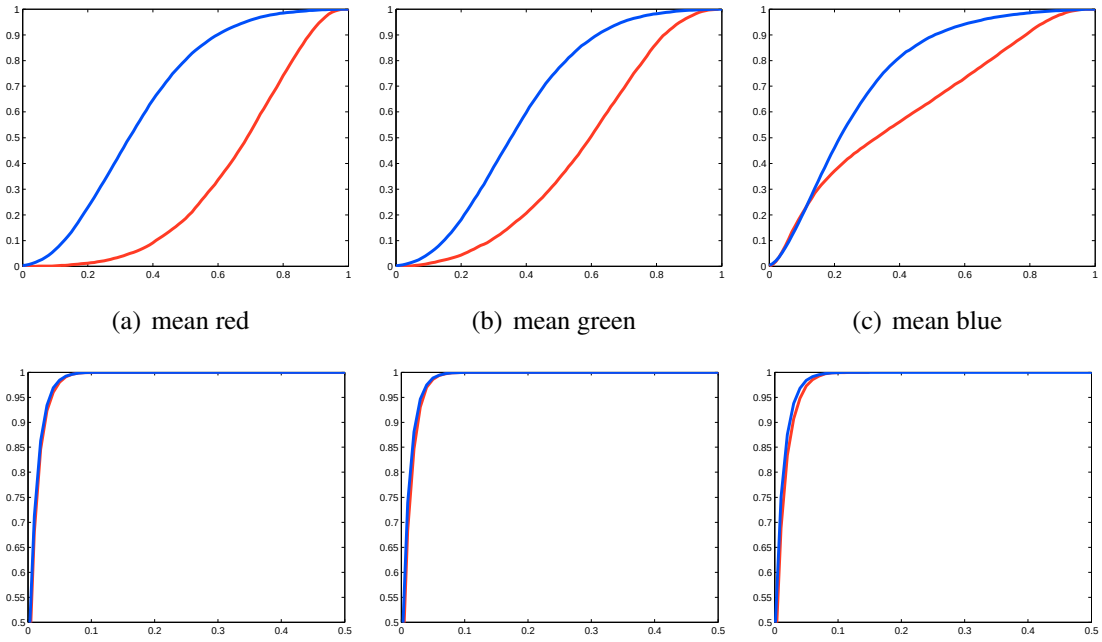


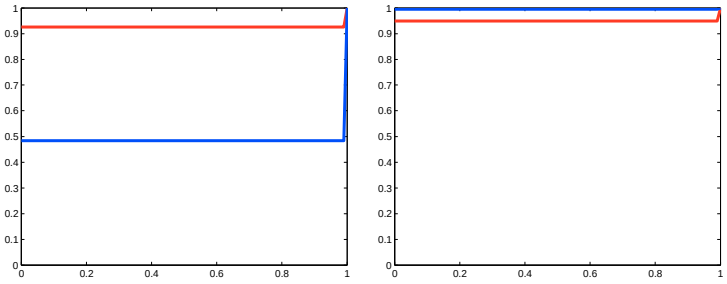
Figure 3.5: Average overlap accuracy versus average number of segments. Green points denote Pareto points, whereas red points show non-Pareto property.



(a) mean red (b) mean green (c) mean blue

(d) variance red (e) variance green (f) variance blue

Figure 3.6: Histograms of RGB color feature. The separability is equivalent to the area between the two curves. E.g. the means contributes more to the separability than the variances.



(a) touching the image boundary (b) covering the image center

Figure 3.7: Histograms of the two location indicators. We like to keep consistent with the other graphs, although these two binary histograms are difficult to read in this form. The left one shows one background superpixel having roughly 50% chance to touch the boundary, while one foreground one only has 10% to do so. The right one shows that one background superpixel rarely covers the image center, while 8% of the training foreground superpixels does so.

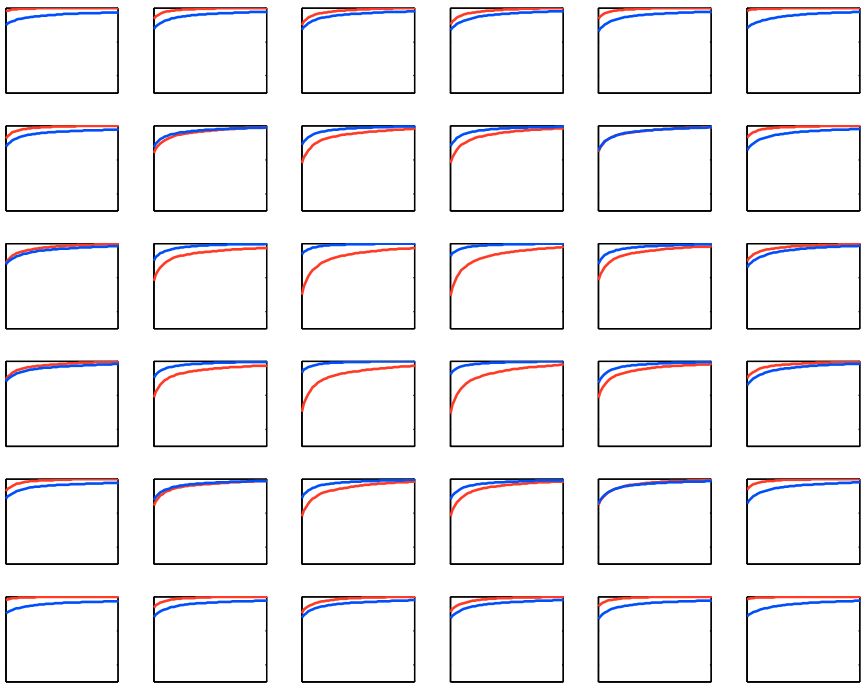


Figure 3.8: The 36-dim. location feature vector. Note that a rough center symmetry can be observed. Also note that the axis are x=[0 ... 0.5], y=[0.5 ... 1]

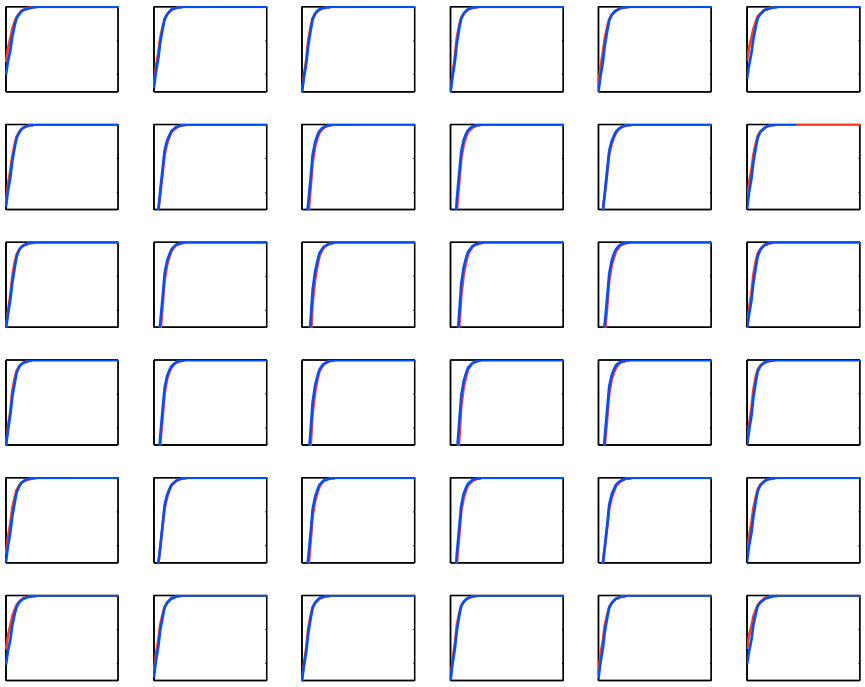


Figure 3.9: The 36-dim. shape feature vector. Although the difference between the curves are slim, including the shape vector indeed helps the segmentation. Note that the axis are also x=[0 ... 0.5], y=[0.5 ... 1]

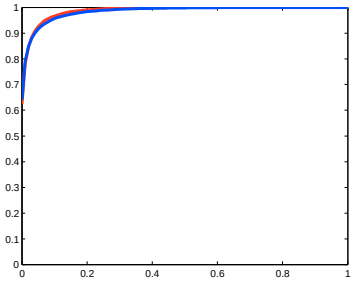


Figure 3.10: Histogram of superpixel areas

Features	Classifier	GC Initialization	# GC Iterations	# GMM Modes	Accuracy
All but shape	SVM	prob. map	1	3	92.0
All	SVM	prob. map	1	3	**93.9**

Table 3.1: On the contribution of the shape feature.

Features	Classifier	GC Initialization	# GC Iterations	# GMM Modes	Accuracy
All	LR	prob. map	1	3	93.7
All	SVM	prob. map	1	3	**93.9**

Table 3.2: On the performance between logistic regression (LR) and support-vector-machine (SVM).

It is shown in Fig. 3.9 that the histograms of shape features only differ slightly, but our experiments show a relatively big improvement due to that feature as shown in Tab. 3.1. We decided to include this feature as the computational overhead is still acceptable. The choice of the discriminative classifier is carried out at Tab. 3.2, where SVM outperforms LR slightly.

3.4.3 On GrabCut

As mentioned previously, our aim is to provide an initialization which can be ported into the GrabCut algorithm. One could argue that the superpixel classification already gives a binary mask for foreground-background segmentation, why do we need GrabCut? Tab. 3.3 and Fig. 3.11 give clear indication that a final GrabCut stage is required.

We investigate two ways to do so using the information provided by the superpixel-wise classification. 1) We put the binary classification of all superpixels of one image together to create a binary mask. The initial GMM for GrabCut then samples the foreground color model from the

Features	Classifier	GC Initialization	# GC Iterations	# GMM Modes	Accuracy
All	SVM	-	-	-	84.1
All	SVM	prob. map	1	3	**93.9**

Table 3.3: On the performance between with or without GrabCut. GrabCut clearly helps the segmentation a lot.

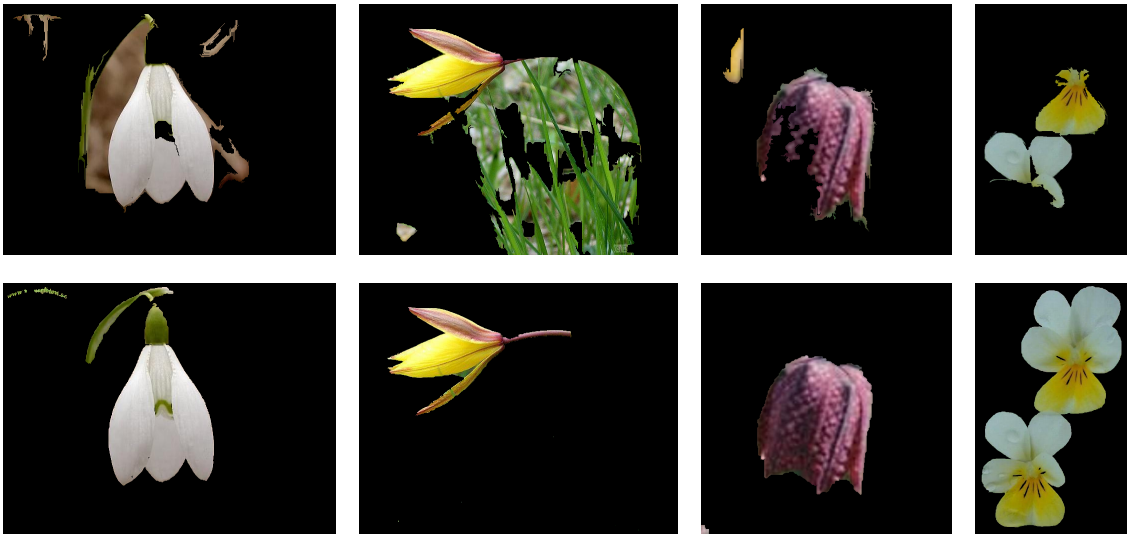


Figure 3.11: Examples of comparisons between having (bottom) or not having (top) GrabCut.

Features	Classifier	GC Initialization	# GC Iterations	# GMM Modes	Accuracy
All	SVM	binary	1	3	92.5
All	SVM	prob. map	1	3	**93.9**

Table 3.4: On the performance between the choice of GrabCut initialization. It either uses a binary or a probability map equal to the SVM margin.

ones in the mask and the background model from the zeros. 2) instead of a binary mask, we make a probability map based on the margin given by the discriminative classifier. The GMM then takes the probabilities as weights into account. Tab. 3.4 shows the improvement of using the probability map over the binary mask.

In the end, we determine the optimum number of GrabCut iteration and the optimum number of GMM Modes (number of GMM centers for each of the categories) in Tab. 3.5 and Tab. 3.6.

Features	Classifier	GC Initialization	# GC Iterations	# GMM Modes	Accuracy
All	SVM	binary	1	3	93.9
All	SVM	binary	5	3	**94.5**

Table 3.5: On the performance change between different number of GMM iterations.

Features	Classifier	GC Initialization	# GC Iterations	# GMM Modes	Accuracy
All	SVM	prob. map	1	2	93.4
All	SVM	prob. map	1	3	**93.9**
All	SVM	prob. map	1	4	93.5
All	SVM	prob. map	1	5	93.4

Table 3.6: On the performance change between different number of GMM modes.

For comparison, the state-of-art work by Nilsback [32] has achieved 93.0% average overlap accuracy on the Oxford Flower 17 Data Set. Her processing time can be assumed to be unsuitable for real-time applications. Also, her supervisor Prof. Andrew Zisserman recalled her processing time to be around 1 minute. Since our processing time is around 4 seconds using one GrabCut iteration, we can safely assume a large gain in processing speed.

More examples of our segmentation pipeline are shown in Fig. 3.12-3.14.

Figure 3.12: Example results of our proposed segmentation method on the Oxford Flower 17 Data Set. Part I.

Figure 3.13: Example results of our proposed segmentation method on the Oxford Flower 17 Data Set. Part II.

Figure 3.14: Example results of our proposed segmentation method on the Oxford Flower 17 Data Set. Part III.

3.5 Discussion

The proposed algorithm performs as well as the method by Nilsback, but using only a fraction of its processing time. It greatly demonstrates the power of appearance-based approach over model-based ones. However, we should remind ourselves of some limitations of this approach and discuss about some other points:

Heavily Relying on Color There is basically no structural information in the algorithm. Some connectivity is explicitly demanded in GrabCut, and implicitly in the superpixel segmentation and classification steps. But overall, the color is an important cue in all steps of the algorithm and therefore constitutes to the major feature of the system. We assume the color histograms of foreground to be different than background. This is often the case, as flowers only take some distinct colors, e.g. flowers are seldom or never green, grey or black. However, the color ”White” is a color that often appears in foreground and background, which causes both false-positives and false-negatives, as shown in Fig. 3.15. This problem can be mitigated using either better features, if they exist, or including structural information which may take long time to process.

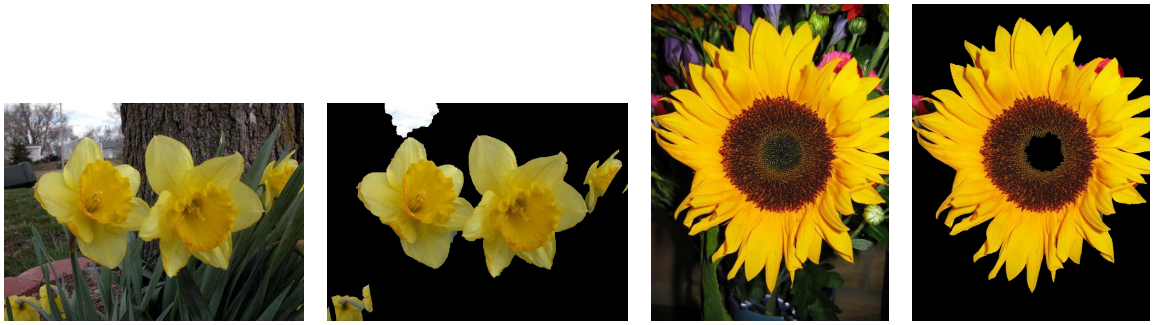


Figure 3.15: Two comparisons showing problems caused by shared color among foreground and backgrounds. In the left case, ”White” is a dominant color in flowers but also appears in background segments such as cloud, wall. In the right case, the center of sunflowers has a very distinct color compared to the petals, and it is also almost black or dark brown which resembles to the color of dirt.

GrabCut Related Drawbacks Our algorithm heavily relies on the final GrabCut method, therefore, also inherits its drawbacks. One of them is the predefined number of GMM modes. As we set the number of foreground GMM modes to be equal as in the background GMMs, GrabCut tends

to balance the color distributions between foreground and background. This property would cause the segmentation of an image containing a very colorful foreground and a uniform background color to fail. Gladly, this problem has not been detected in our data sets so far.

Annotation Required Hand-annotation is the most painful process in machine learning. Unlike recognition tasks where only a label is needed, an entire foreground mask has to be manually cut out for each image in the training set. And when we use another data set, as we often have to do with our collaborators, we would have to make new annotations to get the best results. This fact makes the algorithm difficult to scale to larger problems (See Chap. 5 for our proposed co-segmentation algorithm in order to bypass the hand-annotations.)

Unbalanced Data Set As mentioned before, the Oxford Flower 17 Data Set only has 849 hand-annotated ground truth segmentations out of 1360 images. Unfortunately, the annotated images are not randomly selected as shown in Fig. 3.16. In fact, they are the images that are easy to annotate, which causes an unbalance when using the rest images to test.

Figure 3.16: Unbalance in the appearance of hand-annotated ground truths. As the images on the top have hand-annotations, the images at the bottom, which belong to the same flower species as the top ones, do not.

Better Segmentation equal to Better Recognition? As our aim is to improve the recognition performance with better localization. However, if we use images generated by hand-annotated training images, we implicitly assume that human-recognized foreground also contains the discriminative

information, while the human-recognized background does not. This does not have to be true, but the assumption is approximately valid in our case of flowers. In Chap. 5, we introduce a training data generating process which enforces that the discarded segment to be non-discriminative.

Chapter 4

Classification

We chose a discriminative classifier such as a support-vector-machine (SVM) as opposed to a generative classifier, which is often significantly slower. Nonetheless, there are plenty of variations within the SVM. Similar to the binary classification in the segmentation stage, choice of descriptors (feature selection), kernelization and the type of SVM are to be determined.

4.1 Feature Selection

As being said in the beginning of this report, pairs of the flower species may be only separable from each other in very slight aspects, whereas we want to do our best to cover these marginal properties. We note that the species are originally defined by the botanists, such we want to map natural images to something artificial (like a species name). Before we go on with the research, we want to have a better understanding of how botanists defined the different species and how this affects the recognition process:

- Shape alone is in the most cases sufficient to separate classes from each other. But taking into the account that flowers are highly deformable and may look totally different from different views, the shape property is less reliable due to high intra-class variation. An example of red roses is shown in Fig. 4.1.
- Color on the other hand is more reliable since the reflectance is low in case of flowers, and images are normally taken under similar conditions. If all flowers of the same species have

the same composition of color, it can be classified very well, see Fig. 4.2. However, some flowers have a very large range of different colors, e.g. sword lily can be white, yellow, orange, red, pink or many combination of them, which can push the false-positive ratio of that class higher, see Fig. 4.3.

- A small subset of the flower species have very unique patterns, which can be extracted very reliably. However, they mostly have a unique shape and color composition as well.
- If a pair is only distinguishable by the size, e.g. Vinca major vs. Vinca minor, it is not possible to determine the class based on the flower image without focal length only.

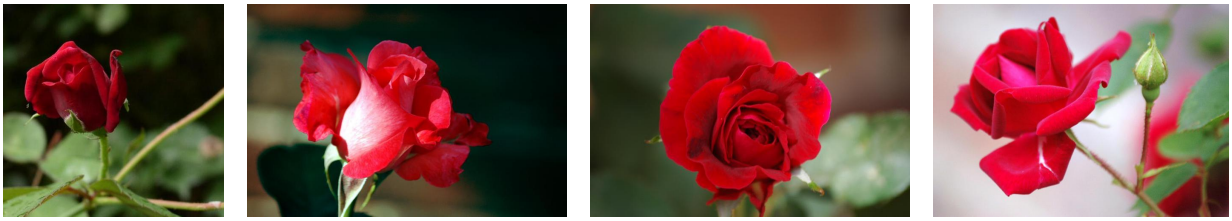


Figure 4.1: Petals of flowers are highly deformable, as shown in the example of roses. They are intentionally chosen to be red to emphasize the shape difference.

Figure 4.2: Color feature is sometimes very consistent within the same species, which makes classification easy. Here an example of pink-yellow dahlias.

Backed by this knowledge and taking the choice by Nilsback into consideration, we decided for 4 descriptors:

Lab Color In the Lab color space, color is represented as a point (a,b) in the color-opponent space, whereas the third dimension is the light strength. We chose this color space as opposed to RGB or HSV based on past experiments. It can be however assumed that the difference in color space does not affect the final results significantly.

Figure 4.3: In some cases, color histograms of flowers from one species vary a lot, where the color feature only slightly helps the classification process . Here an example of hibiscus.

Multi-Scale Dense SIFT Scale-invariant feature transform (SIFT) [29] is a common choice for the shape property. We use an implementation from VLFeat which can extract SIFT from different scales very efficiently. As the nature of dense SIFT, features are extracted on the points of a grid which is overlaid with the image. The fact that later this descriptor is the most powerful single descriptor in our experiments, which agrees with our observation of the importance of the shape.

Interest Point SIFT Also called as sparse SIFT, this descriptor first runs an interest points detector on the image and then extracts SIFT features on these particular points.

Multi-Scale Boundary SIFT This is actually the same descriptor as Multi-Scale Dense SIFT, only extracted along the boundaries of the foreground segmentation mask. It is questionable whether it is worth it including this additional descriptor, since there is already another two SIFT-based descriptors in the system. Later experiments show a slight recognition accuracy decrease if dropping this descriptor.

4.2 Bag-of-Words and SVM

Each of the descriptors discussed above is extracted on multiple points inside the image, which results in a N D-dimensional vectors with N being the number of feature points and D being the feature dimension. In order to get a single feature vector for the final discriminative classifier, we need a mechanism to transform the vector collection into a single vector.

The state-of-art way to do so is the so-called bag-of-words (BOW) approach. First, we build a dictionary of D-dimensional vectors based on the raw feature vectors extracted from all images of the training data using clustering algorithms such as K-Means. Each of the vectors in the dictionary is a *word* in the dictionary. During both training and test time, we vector-quantize every feature

vector according to the pre-built dictionary and linearly pool the output of the vector-quantization to get a single feature vector of the size of the dictionary (number of words). A ℓ^1-normalization is finally applied to ensure the system to work in case of more than one flower in the image.

Past experiments have shown that the choice of the dictionary does not affect the final results significantly as long as it is meaningfully extracted from a subset of the data set. There choose K-Means for its simple implementation and the dictionary is generated by applying K-Means onto a subset of all feature vectors from the training data.

The vector-quantization can be done in either hard or soft way:

Hard Assignment Each feature vector is assigned to its nearest neighbor word in the dictionary. The representation of the input image becomes a simple histogram over all words in the dictionary.

Soft Assignment More than one word is taken into consideration, meaning that one feature vector can be assigned to several nearest neighbors words with different weights depending on their distances. Soft-assignment is a very active research topic in machine learning, we decide for a very recent kernel - the locality-constrained linear coding [48].

We then concatenate the resulted vectors of all descriptors for one image in order to get the final representation of the image. We note that in our case, we have a very high (77440) dimensional vector.

Unlike in the segmentation classifier, a linear SVM is not able to give good results because of the sheer size of the feature vector and the number of labels. Training a kernelized SVM can however take a long processing time, as it requires a complexity of $\mathcal{O}(n^3)$ as opposed to $\mathcal{O}(n)$ for linear SVM, which has a high impact on our large-sized problem. We want to exploit the fast training speed in linear SVMs by first mapping the original feature vector onto an appropriate feature vector before training and testing such we can use a linear SVM instead of a non-linear one. This map is extensively discussed in the work by Vedaldi and Zisserman [46].

We considered the Hellinger's and the χ^2 kernel maps for our approach. According to Vedaldi and Zisserman, the Hellinger's kernel is applied by taking the square root of each element in the feature vector. And we use the χ^2 kernel map implementation from VLFeat [44].

4.3 Introducing Heuristics

Flowers have some shared but specific properties that we did not take into consideration in our more generalized classification pipeline. In this section, we investigate several heuristics to exploit these properties.

Center Masks Flowers mostly consists of a center part (the stigma and the anther) and relatively large petals surrounding it. We investigate the possibility to treat the center and the surrounding petals separately. As localizing the center in a very precise manner would be another segmentation task, which we like to avoid due to time-efficiency, we simply assume the centric ellipse of the foreground segmentation to be the flower center and the rest of the foreground segmentation to be the petals. The pipeline is as follows: we find the largest segment in the foreground segmentation and locate its convex hull, to which an ellipse is fitted according to Fitzgibbon et al [18]. We then take the center part of the ellipse to be the flower center. Some examples are shown in Fig. 4.4.

Mirroring Training Images For the majority of the flowers, it is hard to find enough training images for the recognition accuracy to converge. Therefore, we exploit the fact of flower's vertical symmetry and mirror the training images along the vertical axis in order to create more training images. The fact of a good gain in the recognition accuracy as the result indicates the non-sufficiency of training images in the data set.

Images Always Contain Flowers Sometimes, the segmentation gives no or too few foreground. In that case, we assume that the segmentation method has failed and we use the whole image as foreground area. This heuristic is applied to all experiments in the following section.

4.4 Experiments

Data Set We use the Oxford Flower 102 Data Set [35], which contains 8189 images unequally distributed among all 102 flower species. The largest species contains over 250 images while the smallest ones only have 40. There is no hand-annotated ground truth foreground segmentations available. Such, we used a segmentation pipeline which is trained on the Oxford Flower 17 Data Set.

Recognition Accuracy Measure The measure we use for the recognition accuracy is the mean per-class accuracy, the mean value of the precisions of all species. As the ultimate goal is to design

an application which displays a list of results based on their confidence, we are interested in the top K performance as well, which is defined as the chance a flower being shown within the first K entries of that list. The notation [a/b/c] ($[50/60/70]$) indicates the top 1 performance to be a %, top 5 to be b % and top 10 to be c %.

Hardware We use around 25 nodes on our cluster computers. The whole process takes roughly 0.5 to 2 hours depending the features used for classification.

Segmentation The proposed segmentation methods are used for the experiments. Parameter setting: graph-based segmentation with $\sigma = 1$, $k = 200$ and $\theta = 640$, all features described, linear SVM with $C = 1$, GrabCut with probability map initialization, 5 GMMs each and 1 iteration.

4.4.1 On Features

Each of the descriptors has few hyper-parameters. As it is impossible to do grid search for all of them, we fixed may of them them according to past experiments, common choice and Nilsback's report. The feature choices are:

Lab Color (Color) The Matlab implementation for transforming images from RGB to Lab space is used. We then apply dense sampling onto the transformed image, as taking the Lab value of all pixels would slow down the processing unnecessarily. A step size set to 3 pixels is chosen. The dictionary size for BoW is 800.

Multi-Scale Dense SIFT (MSDS) The image is first transformed to a grey-level image, which is valid for the all SIFT descriptors. Here, we use an implementation from VLFeat. The implementation first smooth the image with a Gaussian kernel with a magnification equal to 6. The step size for the dense sampling is 5 pixels and the different scales are $4, 6, 8$ and 10. The dictionary size for BoW is 8000.

Interest Point SIFT (iSIFT) As interest point SIFT detects the pixels where feature vectors are extracted, we do not have to specify a step size here. The initial Gaussian kernel blur is down with a magnification equal to 3. The dictionary size for BoW is 8000.

Multi-Scale Boundary SIFT (mbSIFT) The boundaries of the foreground segmentation are extracted using the Matlab processing toolbox. We then walk with step size equal to 10 pixels along the boundaries. Again, the image is blurred with a Gaussian kernel with strength equal to 3. The dictionary size for BoW is 8000.

Feature Selection	Assignment Type	Kernel	Heuristics	Accuracy
Color	Soft	χ^2	mirrored	48.7 / 69.3 / 76.0
MSDS	Hard	χ^2	mirrored	75.1 / 89.4 / 93.4
iSIFT	Soft	χ^2	mirrored	61.3 / 80.2 / 87.2
mbSIFT	Soft	χ^2	mirrored	42.0 / 66.0 / 76.0

Table 4.1: On Feature Selection: single descriptors. We try to keep all other factors fixed while changing the feature selection. However, only hard assignment is computationally possible for MSDS due to a very high number of extracted feature vectors from one single image. "mirrored" indicates to the heuristic including vertically mirrored training images

Color	MSDS	iSIFT	mbSIFT	Accuracy
+	+			77.2 / 90.9 / 94.7
	+	+		76.3 / 90.4 / 94.0
	+		+	74.3 / 89.8 / 93.6
+	+	+		80.2 / 92.3 / 96.0
+	+		+	79.6 / 92.3 / 95.7
	+	+	+	77.7 / 91.1 / 94.7
+	+	+	+	**81.4 / 93.5 / 96.4**

Table 4.2: On Feature Selection: combined descriptors. As MSDS is the strongest feature, we keep it chosen in all cases. We note that although there are three SIFT descriptors in the choice, ignoring either of them seems to drop the accuracy by at least 1%.

Tab. 4.1 shows the performance of single descriptors, while Tab. 4.2 shows combination of them. The latter ignores the case without MSDS, as it is most likely that the performance drops significantly if MSDS is not included.

4.4.2 On BoW and Kernels

A comparison between soft and hard assignment is shown in Fig. 4.3, and one between kernels is shown in Fig. 4.4.

Feature Selection	Kernel	Heuristics	Hard Assignment	Soft Assignment
Color	χ^2	mirrored	44.4 / 64.4 / 72.1	**48.7 / 69.3 / 76.0**
MSDS	χ^2	mirrored	**75.1 / 89.4 / 93.4**	-
iSIFT	χ^2	mirrored	47.0 / 70.2 / 79.6	**61.3 / 80.2 / 87.2**
mbSIFT	χ^2	mirrored	32.8 / 56.7 / 67.8	**42.0 / 66.0 / 76.0**

Table 4.3: On the assignment type during BoW. Note that it is computationally impossible to soft assign MSDS features.

Feature Selection	Assignment Type	Heuristics	Hellinger	χ^2
Color	Soft	mirrored	36.9 / 60.5 / 70.4	**48.7 / 69.3 / 76.0**
MSDS	Hard	mirrored	73.7 / 88.7 / 92.6	**75.1 / 89.4 / 93.4**
iSIFT	Soft	mirrored	60.1 / 79.5 / 86.5	**61.3 / 80.2 / 87.2**
mbSIFT	Soft	mirrored	41.9 / 64.2 / 74.4	**42.0 / 66.0 / 76.0**

Table 4.4: On SVM kernel type. Note the significantly larger drop of performance by switching from χ^2 to Hellinger kernel in case of Lab color. We reason this behavioral with the fact that the feature vector for Color is denser than in other three descriptors.

Feature Selection	Assignment Type	Kernel	Without	With
Color	Soft	χ^2	47.5 / 69.2 / 76.7	**48.7 / 69.3 / 76.0**
MSDS	Hard	χ^2	70.0 / 86.4 / 91.6	**75.1 / 89.4 / 93.4**
iSIFT	Soft	χ^2	57.9 / 77.4 / 84.7	**61.3 / 80.2 / 87.2**
mbSIFT	Soft	χ^2	39.4 / 63.2 / 73.0	**42.0 / 66.0 / 76.0**

Table 4.5: On the effect of including mirrored training images into the training set on single descriptors. Note the slight difference in color, which is caused by different sampling in a mirrored image pair.

Feature Selection	Without	With
All	78.4 / 92.0 / 95.6	**81.4 / 93.5 / 96.4**

Table 4.6: On the effect of including mirrored training images into the training set on all descriptors combined. "With" and "Without" are referred to including mirrored images in the training set.

4.4.3 On Heuristics

It is very surprising how the heuristic including mirrored training images in the training data improves the performance, as being shown in Tab. 4.5 and Tab. 4.6. We suspect the problem being the small training data size given by the data set itself (20 images per class) causes non-saturation. The experiment in Tab. 4.7 shows the gain of the performance against the number of training images per class. As the smallest flower species only has 40 images, we can only afford to go up to 30 with the training set, which leaves merely 10 test images for some categories.

# Tr. Images	With	Without
10	67.9 / 87.7 / 92.7	71.6 / 88.8 / 93.2
15	75.1 / 90.2 / 94.7	78.4 / 91.4 / 95.3
20	78.4 / 92.1 / 95.6	81.3 / 93.2 / 96.4
25	81.0 / 93.4 / 96.6	83.0 / 94.2 / 96.9
30	81.9 / 94.0 / 97.0	84.5 / 95.1 / 97.2

Table 4.7: On the effect of growing size of the training data. The setup is the same as in Tab. 4.6, "With" and "Without" are referred to including mirrored images in the training set.

α	Color	MSDS	iSIFT	mbSIFT
without	48.7 / 69.2 / 76.0	73.7 / 88.7 / 92.6	60.1 / 79.5 / 86.5	41.9 / 64.2 / 74.4
0.01	45.4 / 67.0 / 74.9	58.3 / 79.1 / 86.1	48.1 / 71.3 / 80.0	42.4 / 64.6 / 74.5
0.04	48.1 / 69.3 / 76.8	63.6 / 83.0 / 89.2	55.0 / 76.1 / 84.0	46.9 / 69.0 / 77.8
0.09	50.0 / 70.7 / 78.4	69.1 / 85.5 / 90.7	57.9 / 78.9 / 86.2	**48.8 / 71.1 / 80.2**
0.16	51.0 / 72.0 / 79.1	71.9 / 87.8 / 92.0	61.1 / 80.6 / 87.5	48.1 / 70.3 / 79.7
0.25	51.6 / 72.9 / 79.3	74.0 / 88.5 / 92.6	**62.5 / 82.0 / 88.8**	46.5 / 69.9 / 79.0
0.36	**51.9 / 72.8 / 79.4**	**75.2 / 89.2 / 93.1**	62.3 / 83.0 / 88.9	44.9 / 69.0 / 78.3
0.49	51.3 / 72.3 / 79.4	74.5 / 88.9 / 93.2	61.8 / 81.3 / 88.2	43.2 / 65.9 / 76.0
0.64	50.5 / 71.9 / 78.4	72.4 / 88.0 / 92.6	60.2 / 80.2 / 87.6	39.8 / 63.1 / 73.0
0.81	49.3 / 69.8 / 77.4	67.7 / 86.2 / 91.1	55.6 / 78.2 / 85.2	36.7 / 61.5 / 72.6

Table 4.8: Grid Search for the alpha parameters. We notice that the heuristic indeed improves the performance of individual descriptors. The optimum α value varies for different descriptors. The first line show the corresponding performances without the center mask heuristic.

For the center mask heuristic, we first did grid searches on individual descriptors and combined them in the end. The varying parameter α is the proportion of the center area of the ellipse that is assumed to be the flower center. Please note that with this assumption, the lengths of the feature vectors become double of their original size. Therefore, the experiments on the 3 SIFT descriptors are done with the Hellinger kernel, as χ^2 kernel with our parameters would produce triple the vector size, which would make our final vector to be too long as it would be 6-times of original size. The heuristic using mirror training images is included here. And the assignment types are the same as in Tab. 4.4.

The performance of individual descriptors using the center mask is shown in Tab. 4.8. However, by combining them, we were unable to best the results from Tab. 4.6 as we only get [79.4 / 93.0 / 96.1] , most likely because of the using of Hellinger instead of χ^2 kernel.

4.5 Discussion

Without the mirroring image heuristic, we beat the state-of-art performance on the Oxford Flower 102 Data Set by Nilsback by roughly 2% in the average recognition accuracy. Since this differ-

Segmentation	Recognition	Accuracy
Nilsback	Nilsback	76.3 / 91.5 / 95.5
Nilsback	Ours	76.8 / 91.7 / 95.6
Ours	Ours	**78.4 / 92.1 / 95.6**

Table 4.9: In comparison with Nilsback's results. Note that the top 10 performances are almost the same, likely to be saturated due to the relatively small size of the data set. As we work toward over 400 flower species, this number will differ

ence consists of a different segmentation pipeline and a different recognition algorithm, we like to narrow down what exactly makes that 2 %. We then take the segmentation masks provided by Nilsback and port it to our recognition pipeline and get the results in Tab. 4.9.

It seems that the proposed segmentation method (Chap. 3) does not only improve the speed by a factor of 10, but also improves the recognition performance within our proposed recognition framework by 1.5%.

Although our proposed recognition pipeline does not dramatically improve the results, we however believe the geometric layout features by Nilsback are computationally expensive, as the geometric layout features in that algorithm require ellipse fitting, rescaling and rotating to align the flower. Without the geometric layout features and only with Color, dense SIFT, boundary SIFT and Histograms of Gradients (HOG), the algorithm would give 73.4% instead of 76.3%.

Besides the comparison to state-of-art algorithms, we have also discovered some other evidences:

LLC Soft Assignment The LLC [48] soft assignment dramatically improves the recognition performance. Its authors claim that LLC is even good enough that no other kernels (such as χ^2) is required, this property cannot however be proved in our setup.

Heuristics Help! We are surprised ourselves by the huge improvement using a heuristic such as including mirrored training images into the training set. The roughly 3% improves the recognition performance by more than we could do with proper scientific research. There are some other similar heuristics, e.g. rotating the image, which unfortunately could not be tested due to time limitation.

Figure 4.4: Some examples of the Center Masks. 1st column: input images. 2nd column: foreground segmentation. 3rd column: center of the flower. 4th column: petals of the flower. The center of the flowers are separated from the rest in the majority cases. But due to the simplicity of the heuristics, many errors can occur. The majority of the error occurs in case of composite flowers, e.g. adjacent flowers as shown in the 4th example. Another major error source is the lack of a visible flower center at all.

Chapter 5

BiCoS: Bi-Level Co-Segmentation

The segmentation method as described in Chap. 3 requires plenty of training images. Whenever we move to another data set, we have the following options: a) using the classifier trained on the Oxford Flower 17 data set, which is sub-optimal for other data sets. b) we annotate new ground truths for new data sets, which results into a long and painful task.

In order to avoid either two of the mentioned solutions, we investigate a relatively new field in computer vision - co-segmentation, in order to generate the required training data. As already being pointed out in Sec. 2.3, we consider sets of images where the appearance of foreground and/or background share some similarities, and try to leverage these similarities to obtain accurate foreground/background segmentations. Toward our problem, we developed a new co-segmentation method, *BiCoS*, which outperforms recent co-segmentation approaches on the Oxford Flower 17 and 102 data sets. We also evaluate BiCoS on a number of viewpoint-constrained datasets, e.g. Weizmann Horses and Caltech 101, finding out that, despite the lack of geometric modeling, the performance of BiCoS is rather competitive with many of the recent geometry-based methods.

We introduce the BiCoS method that is based on the combination of GrabCut and discriminative learning in the superpixel descriptor space. After that, we demonstrate how the proposed method may be modified to work with a dataset of multiple image sets with shared background patterns (a scenario typical for image classification as discussed above).

5.1 Class-Level Co-Segmentation

Given a set of images of the same class, co-segmentation algorithms attempt to improve the segmentation accuracy by assuming the similarity of the visual appearance of foreground and/or background across the images in the dataset. A number of approaches [20, 31, 39, 47] work with pixel RGB values to model and propagate the distributions of visual appearance. However, in realistic scenarios for the vast majority of classes, RGB distributions of foreground and background pixels across the entire set of images overlap too much for such approaches to be useful. In other words, the color-based representation which works very well at the level of individual images is often unsuitable at the dataset level.

Rather than share a simple descriptor (RGB) at the level of individual pixels across the dataset, we share a richer descriptor at the level of super-pixels. This richer descriptor (a high dimensional feature vector) is suitable for discriminative learning at the dataset level.

Thus, each image is partitioned into a set of superpixels (via the graph-based method [17], preferred for its time efficiency) and described by a real-valued descriptor (with the dimensionality $D = 1076$) stacked from five fairly standard sub-descriptors, representing the superpixels' color distribution, SIFT [29] distribution, size, location within the image and shape (more details below).

The super-pixels are then classified into foreground and background using standard linear support vector machine (SVM) training. Assume that a set $\{\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_N\}$ of N descriptors corresponding to all superpixels are given. Let also $\{y_1, y_2 \dots y_N\}$ be the binary labels with $y_i = +1$ ($y_i = -1$) corresponds to superpixels with the majority of pixels assigned to foreground (background). Then, the separating hyperplane $\mathbf{w}$ in the descriptor space may be obtained by solving the standard SVM (convex quadratic) optimization program:

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N} \ell(y_i \cdot \mathbf{w}^T\mathbf{x}_i) \longrightarrow \min_{\mathbf{w}}, \tag{5.1}$$

where $\ell(t) = \max(0, 1-t)$ is the hinge loss function, and C is the regularization constant set to 10 in all our experiments.

Our co-segmentation method then integrates this discriminative learning step that propagates the information about appearance distributions across images with GrabCut steps that propagate the information within images. Overall, BiCoS has the following steps:

1. **Initialization at the image level.** BiCoS starts with GrabCut being applied to each image independently. GrabCut here is intialized with the rectangle in the center (50% of the image size, unless noted otherwise) assigned to the foreground and the rest to background. The output of this step is a pixel level segmentation of each image.

2. **Propagation at the dataset level.** Each superpixel is assigned in each image to either foreground or background depending on the label of the majority of its pixels. Given the set of superpixels and their labels (over all images), the separating hyperplane $\mathbf{w}$ is then learnt using a linear SVM. Each superpixel is then re-classified according to its sign w.r.t. to the hyperplane (so that superpixels on the "wrong" side of the hyperplane effectively switch label). The output of this step is a linear classifier and a classification of all super-pixels across the dataset.

3. **Update at the image level.** Within each image independently, the Gaussian mixture distribution are reestimated from the new foreground and background regions using the pixels that belong to superpixels classified as foreground or background in the previous step. Each pixel is weighted according to the distance from the separating hyperplane, i.e. the margin $\mathbf{w}^T\mathbf{x}_i$. Given the new mixtures, a single graph cut [7] is then applied with the potentials defined as in GrabCut. The output of this step is again a pixel level segmentation of each image.

One can reiterate the sequence of steps 2 and 3. For the efficiency reasons, we however adopted the single-pass version of BiCoS for our experiments (unless noted otherwise). Overall, scalability to large datasets is an attractive property of the method. Essentially, the scaling is linear in the number of images: the SVM in step 2 can be trained in linear time in the number of superpixels [42] and all other steps are done independently within each image.

We have used a discriminative classifier. Due to the high dimensionality of superpixel descriptors and obvious interdependencies in their dimensions, generative modeling and classification of foreground and background distributions would be problematic, although naive-Bayes approximation could be used [2] as well as topic models as in [40, 9].

Implementation details. The superpixel descriptor is almost identical to the one introduced in Sec. 3.2, with three differences to make it more similar to our classification features: (1) the mean and variance of color is replaced by a 200-dimensional color histogram that is obtained by vector-quantizing and pooling the densely-sampled Lab color values. (2) the scalar area feature is replaced

by an encoding as a tiny 2 bins-histogram with the 1 value placed depending on whether the superpixel is "big" or "small" (the meaning of "big" and "small" is obtained by clustering the sizes of superpixels in the training set into two clusters). (3) we add a 800-dimensional SIFT histograms that are obtained in a similar way to the color, but using multi-scaled dense SIFT descriptors. The resulting descriptor is quite high-dimensional (1076-D), but is relatively cheap to compute and permits the similarity computation by a simple dot product (hence no need for the kernelization of the algorithms). The vector-quantization is done with locality-constrained linear coding [48].

5.2 Co-segmenting multiple class image sets

The GrabCut algorithm achieves a remarkable accuracy by making foreground and background regions of an image to share the same appearance distribution. Co-segmentation algorithms including BiCoS are capable of improving segmentations by enforcing the sharing of such distributions across the images of the same class. In this subsection, we make one step further and consider the scenario when one is given a dataset that contains multiple sets of images $\mathcal{S}_1, \mathcal{S}_2 \dots \mathcal{S}_K$ corresponding to K classes, with each set $\mathcal{S}_k$ containing N_k images ($\mathcal{S}_k = \{\mathcal{I}_1, \mathcal{I}_2, \dots \mathcal{I}_{N_k}\}$), where N_k may be as small as 1 and as big as several hundreds.

Clearly, such datasets may be segmented by running co-segmentation algorithm for each set S_k independently. However, one may attempt to improve the segmentation even further by enforcing appearance sharing across classes. As discussed in the introduction, we assume that the ultimate goal in this scenario is the improvement of the recognition accuracy for classifiers trained from the respective training sets after background removal. The consequence of this observation is that a "good" co-segmentation process should have a tendency to assign regions with the appearance patterns that are ubiquitous across multiple image sets to background (as these patterns would not be discriminative but rather confusing for class discrimination).

Our second approach (**BiCoS-MT**) enforces background sharing by modifying the SVM formulation in BiCoS. For each class-specific dataset S_k, the algorithm finds a separate hyperplane that discriminates between the background and the foreground appearances typical for that class. The hyperplanes for different classes are however not computed independently. Instead, for the class k, the separating hyperplane is found as the sum of two vectors $\mathbf{w}_k + \mathbf{w}_-$, where $\mathbf{w}_k$ is the class-specific part and $\mathbf{w}_-$ is the component shared across all classes. The objective of the joint

SVM formulation is then naturally defined as:

$$\frac{\mu}{2}||\mathbf{w}_-||^2 + \sum_{k=1}^{K}\left[\frac{1}{2}||\mathbf{w}_k||^2 + C\sum_{i=1}^{N_k}\ell\left(y_i^k\cdot(\mathbf{w}_k+\mathbf{w}_-)^T\mathbf{x}_i^k\right)\right] \rightarrow \min_{\{\mathbf{w}_1\ldots\mathbf{w}_K,\mathbf{w}_-\}}. \tag{5.2}$$

Thus, (5.2) is defined as a sum of K independent SVM objectives for each class along with the vector component $\mathbf{w}_-$ (and its regularization term) shared across the SVMs. (In (5.2), $\mathbf{x}_i^k$ and y_i^k denote superpixels within the dataset $\mathcal{S}_k$; N_k denotes the number of superpixels in image set $\mathcal{S}_k$; μ is an additional regularization constraint set to $\frac{1}{2}$ in all our experiments.)

Since we want to encourage the sharing of background and discourage the sharing of the foreground appearance patterns, we impose additional constraints on (5.2):

$$\mathbf{w}_k^j \geq 0, \quad \forall j = 1\ldots D \tag{5.3}$$

$$\mathbf{w}_-^j \leq 0, \quad \forall j = 1\ldots D \tag{5.4}$$

An intuition behind the constraints (5.4) is that they make the separating hyperplanes for different classes to share negative coefficients (that are indicative of background superpixels), while positive coefficients can be inferred for each class independently. With respect to the original SVM formulation, it is easy to show that for the case of a single class $K = 1$ and equal regularization on $\mathbf{w}_1$ and $\mathbf{w}_-$ (i.e. when $\mu = 1$), the standard SVM formulation (5.1) and the new formulation (5.2)–(5.4) are exactly equivalent and produce the same separating hyperplane.

Based on the augmented SVM formulation, we devise the BiCoS-MT algorithm that uses the joint program (5.2)–(5.4) to estimate the separating hyperplane $\mathbf{w}_k+\mathbf{w}_-$ for each class k on the step 2. The rest of the steps are unchanged from BiCoS. We note that the joint SVM formulation can be naturally interpreted as an instance of multi-task learning [10, 16] (hence the name BiCoS-MT). Within BiCoS-MT, each task corresponds to learning a superpixel classifier for a particular image set $\mathcal{S}_k$. We also note that "softer" version of background sharing may be implemented, where each class possesses both positive component $\mathbf{w}_k^+$ and negative component $\mathbf{w}_k^-$ specific to that class, whereas the extra term $\nu\sum_k||\mathbf{w}_k^- - \mathbf{w}^-||^2$ penalizes the deviation of the negative components from their joint mean vector $\mathbf{w}^-$ that does not enter in any other terms (c.f. [16]).

The program (5.2)–(5.4) may be rather large (many tens of thousands of variables in some of our experiments), and we therefore implemented the modification of a popular stochastic gradient

descent-based method (Pegasos [42]) to handle the task. This allows to scale (5.2)–(5.4) to large number of classes and/or large number of images per class.

5.3 Experiments

Performance measures. During the evaluation of co-segmentation systems, we are interested in two measures: the segmentation accuracy and the accuracy of the recognition achieved by a visual classifier trained on the segmented dataset, and we report these measures for three image classification datasets and several methods.

Given a ground truth foreground mask, the segmentation accuracy can be expressed in several ways. The one is the ratio between the size of the intersection area between the estimated foreground and ground truth foreground over the size of their union, as already used in Chap. 3 (**Seg. I**). The other is the percentage of pixels classified correctly as either foreground or background (**Seg. II**). We report both numbers for our methods, and whichever is published for the competitors. For the average recognition accuracy (**Rec.**), we take the mean value of the relative numbers of correctly classified images of each category by a state-of-the-art visual classifier briefly described below.

Segmenting images at test time. We use the foreground segmentations from our algorithms and other co-segmentation algorithms as training data. As we don't know the label of test images, co-segmentation algorithms may not be applied during test time. However, we propose a segmentation method for test time which is similar to our algorithms. For each training image, we again divide it into superpixels, where this time, we train a *universal* discriminative classifier (an SVM) based on all superpixels extracted from the training data independent of their category labels. During test time, this universal classifier is applied to superpixels of the test image to classify its superpixels into foreground or background. We then apply step 3 of BiCoS to each image (i.e. we put superpixels each with their SVM margin together and make 1 step of GrabCut).

Classification algorithm. The features for each image are obtained by concatenating the Bag-Of-Words (BoW) histograms of Lab color and SIFT descriptors, which are extracted solely from the foreground area given by the segmentation. We use three different histograms for SIFT feature corresponding to dense sampling at several scales, sampling at interest points, and sampling along the

foreground boundary. Similar to the superpixel descriptors, the vector-quantization uses locally-constrained linear coding [48]. The vocabulary sizes are 800 for LAB and 8000 for each of the 3 SIFT descriptors. Before concatenating the four descriptors together, we apply the homogeneous kernel map with approximation order 1 and sampling step 0.7 [46] onto it, which results in a 74400 dimensional feature vector, that is used within the linear SVM (with $C = 1$). The implementation uses VL-Feat [44].

5.3.1 Oxford Flowers 17

We first compare our algorithms to the recently published co-segmentation algorithm by Joulin et al. [23]. We use their provided code and their suggested parameter settings for this dataset. Unfortunately, this algorithm uses an excessive amount of memory, so that even our computer with 16 GB memory cannot handle the amount of memory needed for the predefined training split. We then made a new data split having 15 training and 65 test images per category. We also split it so that those 15 training images from 16 of 17 categories have ground truth segmentations. Thus, for the new split we can compare both segmentation and recognition accuracies. We note that other published co-segmentation methods either do not scale to even this number of images or require user supervision [4], making [23] the natural choice for the comparison.

The numerical comparison between our own methods and method [23] is shown in Tab. 5.1, while some of the segmented images are shown in Fig. 5.1. In Tab. 5.1 and thereafter, we also provide the accuracies for the cases when no pre-segmentation of training and test images is performed, and when the training dataset is pre-segmented with GrabCut.

We also compare the recognition accuracy of our full system that uses BiCoS to co-segment training image with the state-of-art recognition results using the predefined data splits. We show our recognition accuracies in Tab. 5.2 along with the results provided by Gehler and Nowozin [19] and Nilsback [32].

5.3.2 Oxford Flowers 102

Among all recent approaches evaluated on this dataset [22, 24], the best performance is reported in [32]. The authors proposed a model-based foreground segmentation approach which segments

Methods	Seg. I	Seg. II	Rec.
All foreground	32.8	32.8	62.1
Joulin et al. CVPR'10 [23]	75.8	86.6	74.1
GrabCut [38]	89.3	96.3	73.3
BiCoS (this work)	94.2	98.3	79.3
BiCoS-MT (this work)	94.3	98.2	80.5

Table 5.1: Performance on Oxford Flowers 17 (with the alternative data split – 15 images per class). Our algorithms achieves the highest results in all three measures. For this number of training images, BiCoS-MT also outperforms BiCoS in terms of the recognition accuracy.

Methods	Rec. Accuracy
Gehler and Nowozin ICCV'09 [19]	85.5 ± 3.0
Nilsback' thesis [32]	88.1 ± 1.9
BiCoS (this work)	91.1 ± 1.5
BiCoS-MT (this work)	90.4 ± 2.3

Table 5.2: Performance on Oxford Flowers 17 (with the original data splits – 60 training images per class). Our algorithms once again lead to highest recognition accuracies. For 60 images per class, BiCoS-MT no longer perform better than BiCoS.

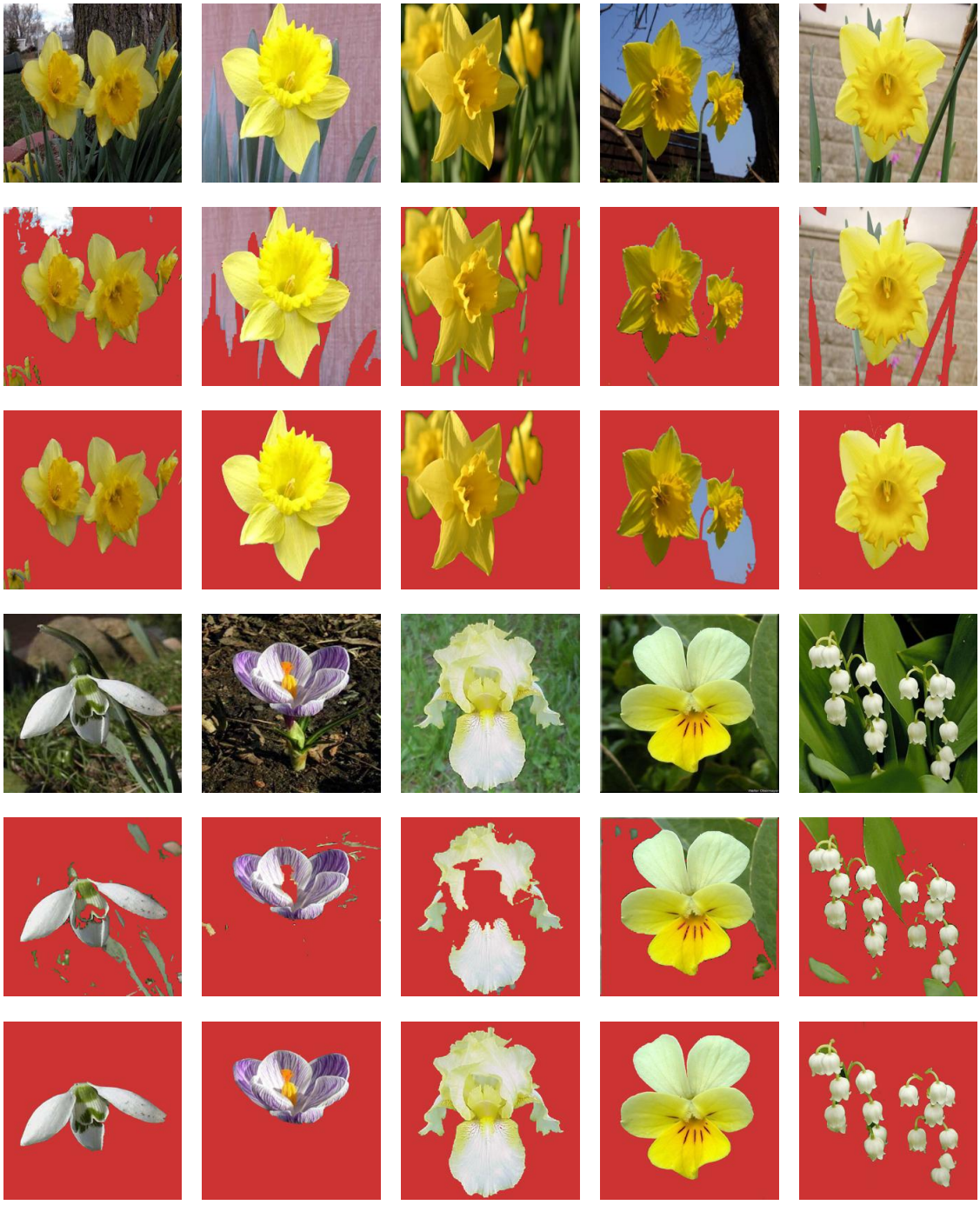


Figure 5.1: Flower segmentations: (top) original images, (middle) segmentations by Joulin et al. [23] and (bottom) the segmentations of BiCoS-MT. The results of BiCoS are very similar to BiCoS-MT for these particular images.

Segmentation	Classification	Rec. Accuracy
no segmentation	ours	64.7
Nilsback [32]	Nilsback [32]	76.3
Nilsback [32]	ours	76.8
Kanan and Cottrell [24]	Kanan [24]	72.8
Ito and Cubota[22]	Ito [22]	74.8
GrabCut [38]	ours	77.0
BiCoS	ours	79.4
BiCoS-MT	ours	80.0

Table 5.3: Recognition performance for different combination of segmentaiton and classification approaches on Oxford Flowers 102. For a number of methods, we vary the (co-)segmentation approach while keeping the classification approach fixed. The proposed methods once again lead to highest recognition accuracy.

images independently. In the following experiments, we can show that this baseline can be surprisingly outperformed in our experiment framework even if the images are segmented independently using GrabCut alone. We also show that the accuracy can be pushed even further with the proposed methods (Tab. 5.3). Note that since the segmentations used in [32] are also available online, we were able to evaluate the combination of their segmentation system and our image classifier. The improvement obtained by BiCoS and BiCoS-MT over [32] in this experiment is all the more important, as the system [32] required around 800 hand-annotated segmentation masks as training data.

5.3.3 Caltech-UCSD Birds 200

The recently published Caltech-UCSD Birds 200 [49] contains 200 bird categories and 6033 images in total. There are *rough* segmentation masks provided with the dataset, which allowed us to compare segmentation accuracies. However, we should note that those values are only approximated because of the rough segmentation ground truth obtained with MTurk. The dataset is extremely challenging, and its authors report only 19% recognition accuracy *when using ground truth masks*. We provide the segmentation and classification accuracies (using the suggested 20

Methods	Seg. I	Seg. II	Rec.
no segmentation	17.0	17.0	6.7
GrabCut [38]	38.8	73.5	13.6
BiCoS	41.1	78.3	15.7
BiCoS-MT	39.9	77.3	16.2
using ground truth seg.	-	-	23.3

Table 5.4: Performance on Caltech-UCSD Birds 200. BiCoS algorithms outperform GrabCut in all measures. BiCoS-MT leads to slightly higher recognition accuracy. The bottom line provides the performance of our classification approach given ground truth segmentation.

training images per class split) of GrabCut, BiCoS, and BiCoS-MT paired with our recognition approach Tab. 5.4. We hope that they may serve as a baseline for the further research on segmentation and co-segmentation. Some examples for BiCoS and BiCoS-MT are shown in Fig. 5.2.

5.3.4 Weizmann Horses and Caltech-4

We also evaluate BiCoS on datasets with fixed viewpoint orientation and classes that have well-defined geometric shape. This scenario is quite different from our main application and it is inevitable that methods interleaving segmentation and construction of class-specific geometric models are likely to have an advantage over BiCoS or any other method not having a geometric model. Still, it is interesting to see how well can BiCoS fare against such methods, in particular against Alexe et al. [2] to which it is most similar. We therefore evaluated BiCoS on Weizmann Horses [6] and 3 out of 4 Caltech-4 classes [27] (the grayscale 'cars' class was left out). We keep all the parameters fixed from the Flowers and Birds experiments, except that GrabCut is initialized with the central foreground rectangle occupying 25%(rather than 50%) of the image area.

The comparison with state-of-the-art is in Tab. 5.5. Here we also give results for the iterated BiCoS (repeating step 2 to step 3 for 5 times). In the experiments, BiCoS performed poorly on the motorbike class, where GrabCut often fails due to unnatural framing within many photographs. Apart from that, the performance of BiCoS is surprisingly competitive, in particular outperforming Alexe et al. [2] on the remaining 4 categories. The top performing method of [28] outperforms BiCoS on all categories except airplanes where BiCoS (1 iteration) is better.

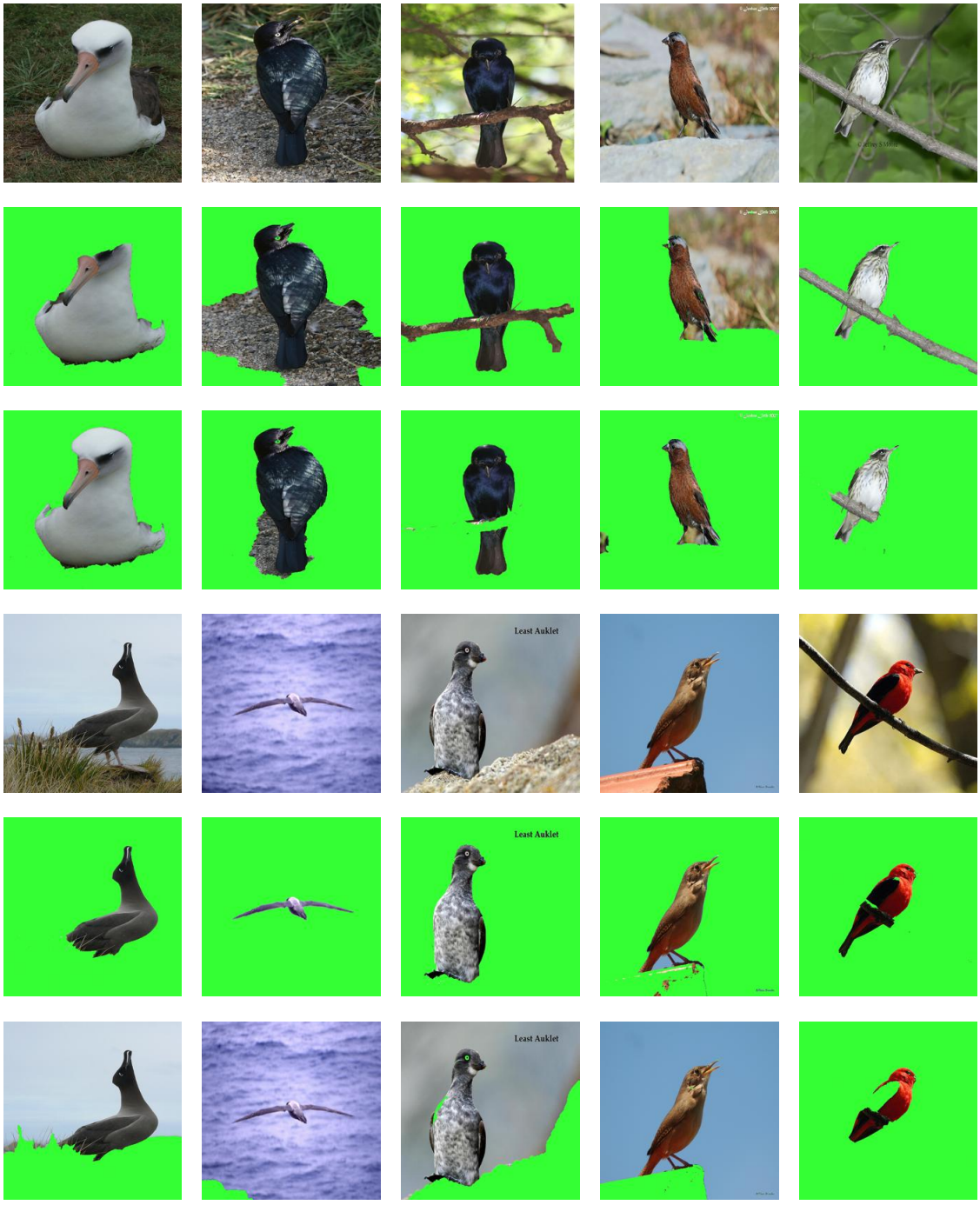


Figure 5.2: Bird segmentations: (top) original images, (middle) using BiCoS and (bottom) using BiCoS-MT. In the top 5 images, BiCoS-MT performs better because it takes into account background elements (e.g. tree branches) in the background of images from other classes. But due the existence of confusing foreground elements (e.g. blue birds) in other classes, sky can be misclassified using BiCoS-MT.

Methods	horse		airplane		face		motorbike	
	Seg I	Seg II	Seg I	Seg II	Seg I	Seg II	Seg I	Seg II
All background	0.0	71.0	0.0	83.1	0.0	80.0	0.0	72.9
Joulin et al. [23]	-	80.1	-	-	-	-	-	-
Cao&Fei-Fei [9]	-	81.8	-	-	-	-	-	-
Alexe et al. [2]	-	86.2	-	89.8	-	89.0	-	90.3
Arora et al. [3]	-	-	-	93.1	-	**92.4**	-	83.1
LOCUS [50]	-	93.1	-	-	-	-	-	-
Liu et al. [28]	-	**95.9**	63.9	-	-	-	**71.6**	-
GrabCut [38]	60.6	86.3	59.2	90.7	60.1	86.2	41.1	80.6
BiCoS (1 iter.)	64.5	89.4	**64.6**	93.0	66.2	89.3	44.7	82.8
BiCoS (5 iter.)	68.7	90.0	63.9	**93.2**	68.9	91.1	41.9	82.4

Table 5.5: (Co-)segmentation performance on Weizmann Horses and Caltech-4 datasets reported in the literature. Despite the lack of the geometric model, BiCoS is competitive with the methods that have such models (see text for more discussion).

5.4 Discussion

We presented BiCoS: a co-segmentation method that is arguably very simple, yet is scalable and performs remarkably well in our experimental comparisons. BiCoS outperformed GrabCut that treats images independently and the state-of-the-art co-segmentation [23] in all comparisons. Image classification systems trained on datasets co-segmented with BiCoS achieved state-of-the-art accuracy on both Oxford Flowers datasets. For the small number of images per class (Tab. 5.1, 5.3) the multitask version of BiCoS lead to even higher recognition accuracy. Many co-segmentation methods attempt to segment images and infer the foreground and background appearance models at the same time. BiCoS follows a different, perhaps less principled way, and performs within-image and across-image appearance propagation consecutively rather than jointly. Such two-step approach, however, allows to use proper feature representation at each level (color-based at the level of individual images, discriminative classification of rich superpixel descriptors at the dataset level). The success of BiCoS in our experiments therefore seems to concur with the well-known fact that devising appropriate feature representations has a higher impact on the performance of computer vision systems than the choice of a principled optimization algorithm.

Chapter 6

Conclusion

The contribution of this project can be concluded in three scientific and one application-specific parts:

Segmentation We designed a foreground-background segmentation method that is as accurate as the state-of-art work by Nilsback [32] (overlap ratio is 94%), but improves the speed by a factor of roughly 10 to around few seconds. The proposed algorithm first divides the input image into superpixels, which are classified to either foreground or background with a pre-trained discriminative classifier. Then, GrabCut is applied in order to remove the artifacts by the superpixels and apply global information onto the image. We demonstrated the power of GrabCut on our problem. We assume that the fact that flowers have relatively distinct color models, and the fact that the flowers are often in the center and in focus as they are taken into the images, are the main reason for the successful usage of GrabCut in our framework.

Recognition By combining our segmentation method with the recognition method, we could achieve an accuracy of 83% of predicting the correct result, compared to 76.9% by Nilsback. The number even goes to 96.9% if only ensuring the correct result to be among the top 10 proposals, as the final application is meant to show that number of proposals. Our method uses a bag-of-words (BoW) approach with features including the Lab color, multi-scale dense SIFT, boundary SIFT and sparse SIFT. We show that despite three SIFT descriptors are used, abandoning each of them will result a notable drop in recognition accuracy. Moreover, we show the power of the locally-constraint linear coding [48] over hard assignment during our pooling step in BoW. Finally, we demonstrated in our case, where training images of many flowers are rarely found, mirroring

images and including them into the training set helps the recognition significantly.

Co-Segmentation We presented BiCoS: a co-segmentation method that is arguably very simple, yet is scalable and performs remarkably well in our experimental comparisons. BiCoS outperformed GrabCut that treats images independently and the state-of-the-art co-segmentation [23] in all comparisons. Image classification systems trained on datasets co-segmented with BiCoS achieved state-of-the-art accuracy on both Oxford Flower data sets. For the small number of images per class, the multitask version of BiCoS lead to even higher recognition accuracy. Many co-segmentation methods attempt to segment images and infer the foreground and background appearance models at the same time. BiCoS follows a different, perhaps less principled way, and performs within-image and across-image appearance propagation consecutively rather than jointly. Such two-step approach, however, allows to use proper feature representation at each level (color-based at the level of individual images, discriminative classification of rich superpixel descriptors at the dataset level). The success of BiCoS in our experiments therefore seems to concur with the well-known fact that devising appropriate feature representations has a higher impact on the performance of computer vision systems than the choice of a principled optimization algorithm.

Applications We built a system for our recognition engine according to the common design pattern Model-View-Control (MVC), which provide interfaces to both WWW and mobile platforms.

6.1 Outlook

We first discuss what we could still try on the existing framework, followed by a proposal of using attributes to discriminate between flowers. We then finalize the thesis with the idea of ”Thinking Large”.

Speed Although our existing system needs roughly 10 seconds to process, which is a common processing time for mobiles phone application (e.g. Google Goggles [1]), we still want it to be faster. Although superpixels contain information about which pixels belong together, the fact that the two different superpixel algorithms (graph-based and Quick Shift) performed roughly the same after GrabCut may indicate that the initial superpixel segmentation does not matter that much. Such, we could use a simple grid instead of superpixel. By doing so, we save the time of dividing superpixels and the feature pooling becomes easier as well. Microsoft Windows 2010 has shown that GrabCut can be done within a tenth of a second, as opposed to the few seconds that our

implementation takes. We are aware of several approximations and variations that increases the speed, but decreases the performance, e.g. using histogram instead of Gaussian mixture as the color model. During the recognition step, we can experiment about the steps size and sample rate for SIFT and color, respectively.

Attributes The categories, as in our case of flower species, are primarily separated by biologists based on human-recognized attributes. Those attributes can be visible attributes, like the color or the number of petals, or non-visible attributes, like the smell or the interior structure that is not visible in images. This kind of separation between the categories can be easily modeled into a tree. In other words, if we are able to extract the attributes properly, the recognition ratio can be nearly 100% (as there are categories which only differ in properties that are not captured by images). Another advantage of using attributes is that it enables user interaction. If our system is not sure about a specific attribute, it can ask the user. Such, in the next step of our work, we would like to assess from this completely different angle.

Thinking Large How many people can profit from our research and application? To be honest, I can only imagine a very small population who are not professionally botanists but are interested in what flowers are growing around them. However, this fact of a low number of potential users does not make our research worthless. In contrary, the idea embedded in our system is to provide users with information based solely on images. By thinking in a large scale, we imagine our system to provide extra knowledge based on any kind of images. It would be a Wikipedia based on image instead of text search. Perona [37] formulated this idea in his ”Vision of Visipedia”. And we are eager to make this vision into reality.

Appendix A

System Design

Research is worthless as long as there is no application. Therefore, we designed a demo system of our flower recognition that can be accessed from both the the world-wide-web (WWW) or and mobile platforms. Moreover, we designed a simple API for our collaborators (see Chap. B). As our collaborators have their own data sets which should put onto the demo, we need different back-ends in our system. To summarize, our goal is to design a server which can handle multiple types of front-ends and multiple back-ends. As shown in Fig. A.1, we use the standard concept of model-view-control (MVC) design pattern.

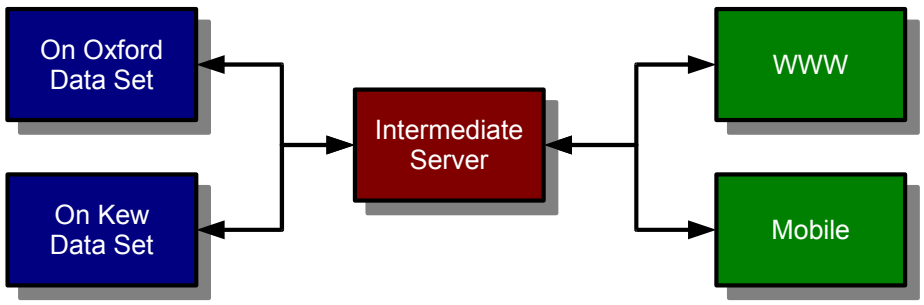


Figure A.1: An overview of the demo system. The model-view separation is done using an intermediate server, which allows extensions in both front-end and back-end

Back-End At the moment, the available code runs on multi-thread Matlab with bottleneck functions implemented in C. The recognition process takes around 10 seconds. We intend to re-

implement it in native languages but have been unable to do so due to time limitations.

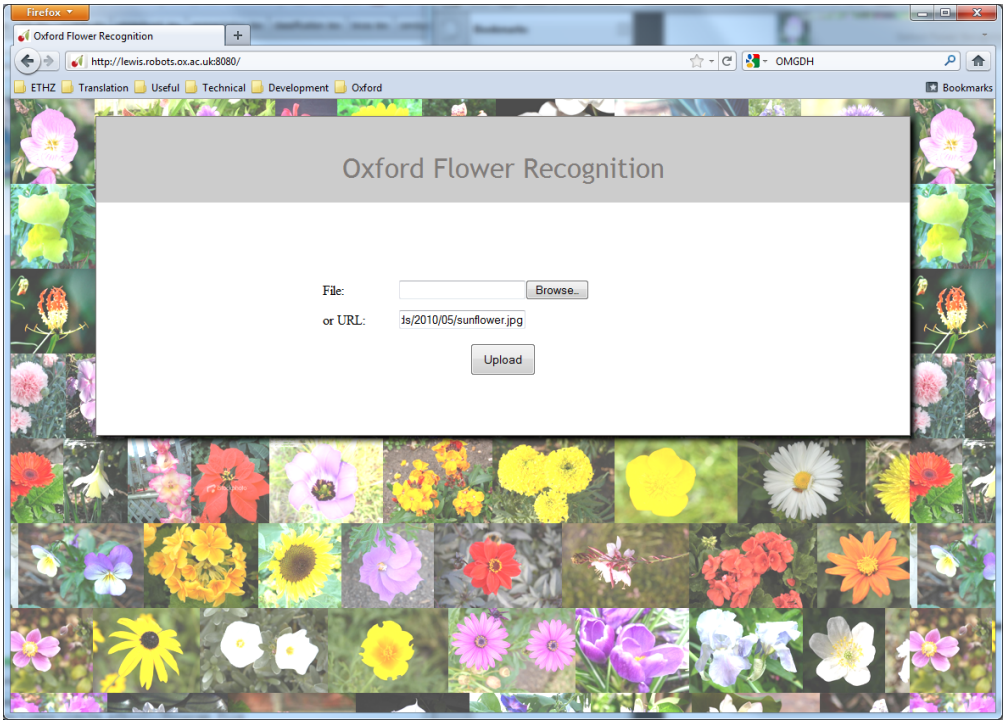
Intermediate Server Although the intermediate server is not visible from outside of the computer, it collects queries from Web server and the mobile server, and forward them to the according back-ends.

WWW Server As one would expect, the server takes an image as either a file or a URL link and gives a list of possible candidates as output.

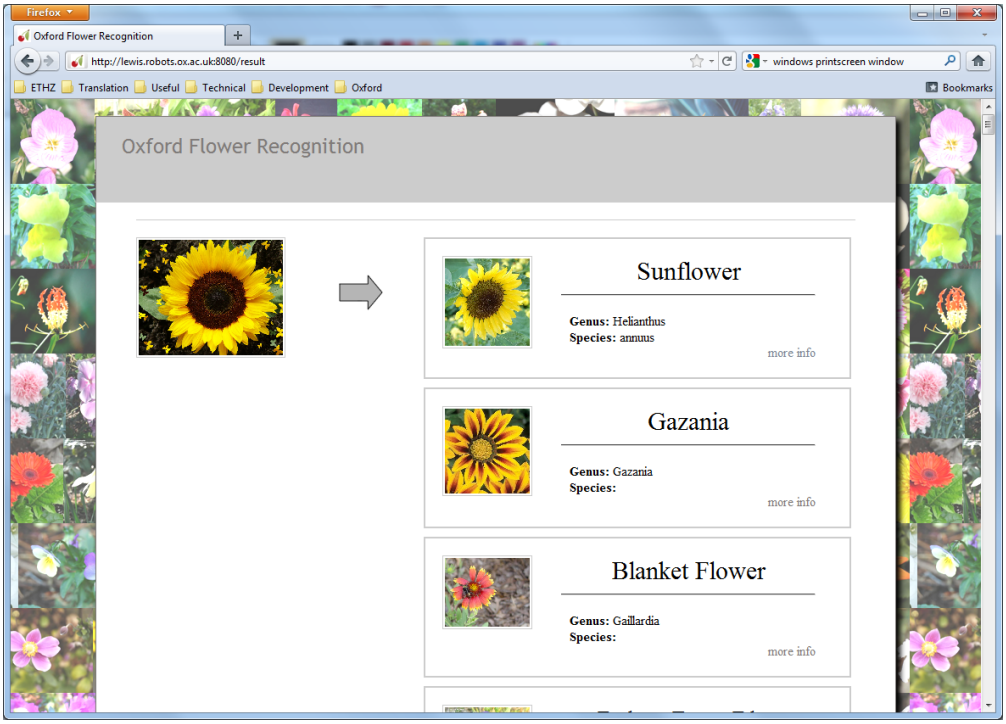
Mobile Platform With the available time so far, we are only able to develop a demo for the Google Android platform. The simple demo is not more complicated than letting the user to take an image with an Android phone, and the system sends back the results back which is displayed as shown in Fig. A.3

The API At the moment, the input is a simple HTTP GET function, while the output is a file containing a list of the 2-tuple $\{$species ID, confidence$\}$.

Security To our best knowledge, our system might be vulnerable to two cyber-attacks. *Code Insertion* deals with the fact that any parameters can be sent to the server, which can wipe off the whole disks if it is not designed properly. In order to prevent this, sanity checks are used on each server. *Denial-of-Service (DoS)* is another common attacks, however, the only solution is to put the attacking IP addresses into the firewall as soon as possible. DoS itself cannot destroy stored data. As the interest of denying our free flower recognition service is low, we do not apply any techniques to prevent DoS.



(a) Start page



(b) Results

Figure A.2: An overview of the web demo. Either a file or a URL can be passed into the system in order to assign a flower to a species.

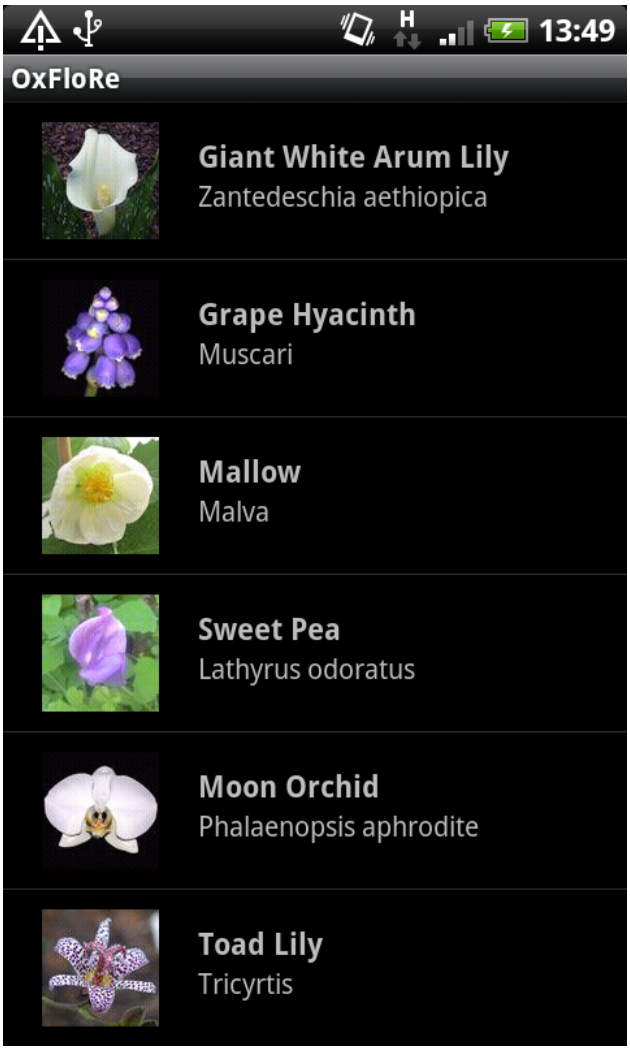


Figure A.3: The result as a short-list displayed on an Android phone

Appendix B

Collaboration with the Royal Botanic Gardens Kew

The Royal Botanic Gardens Kew, which is located just outside of London, is our only collaborator. The garden is interested in developing its own App for flower recognition, with an initial goal of over 400 common British flower species and over 1000 world-wide, which poses a real challenge to our system.

They have a continuously growing data base. At this time, 115 flower species with 8054 images in total are included. Unfortunately, we are unable to show sample images due to copy right issues.

Appendix C

Task Description

The goals, as they were defined in the beginning of this project, consists of the following points:

- Segmentation: Increase the segmentation performance and speed using a newer approach instead of the model-based approach.
- Recognition: Increase the recognition performance and speed.
- Demo: A web demo for flower recognition.

The time period starts on the 4th of October and ends on the 31st of March.

Bibliography

[1] Google goggles. `http://www.google.com/mobile/goggles/#text`, 2008.

[2] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Classcut for unsupervised class segmentation. In *ECCV*, 2010.

[3] Himanshu Arora, Nicolas Loeff, David A. Forsyth, and Narendra Ahuja. Unsupervised segmentation of objects using efficient learning. In *CVPR*, 2007.

[4] Dhruv Batra, Adarsh Kowdle, Devi Parikh, Jiebo Luo, and Tsuhan Chen. icoseg: Interactive co-segmentation with intelligent scribble guidance. In *CVPR*, 2010.

[5] Andrew Blake, Carsten Rother, M. Brown, Patrick Pérez, and Philip H. S. Torr. Interactive image segmentation using an adaptive GMMRF model. In *ECCV*, 2004.

[6] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *ECCV*, 2002.

[7] Yuri Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *ICCV*, 2001.

[8] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2004.

[9] L. L. Cao and L. Fei-Fei. Spatially coherent latent topic model for concurrent segmentation and classification of objects and scenes. In *ICCV*, 2007.

[10] Rich Caruana. Multitask learning. *Machine Learning*, 28(1), 1997.

[11] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell*, 2002.

[12] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cdric Bray. Visual categorization with bags of keypoints. In *ECCV*, 2004.

[13] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR (1)*, 2005.

[14] A. Ess, T. Mueller, H. Grabner, and L. J. Van gool. Segmentation-based urban traffic scene understanding. In *BMVC*, 2009.

[15] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 2010.

[16] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *KDD*, 2004.

[17] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2), 2004.

[18] A. W. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least square fitting of ellipses. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1999.

[19] Peter V. Gehler and Sebastian Nowozin. On feature combination for multiclass object classification. In *ICCV*, pages 221–228, 2009.

[20] Dorit S. Hochbaum and Vikas Singh. An efficient algorithm for co-segmentation. In *ICCV*, 2009.

[21] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Geometric context from a single image. In *ICCV*, 2005.

[22] Satoshi Ito and Susumu Kubota. Object classification using heterogeneous co-occurrence features. In *ECCV*, 2010.

[23] Armand Joulin, Francis R. Bach, and Jean Ponce. Discriminative clustering for image co-segmentation. In *CVPR*, 2010.

[24] Christopher Kanan and Garrison W. Cottrell. Robust classification of objects, faces, and flowers using natural image statistics. In *CVPR*, 2010.

[25] M. Pawan Kumar, Philip H. S. Torr, and Andrew Zisserman. OBJCUT: Efficient segmentation using top-down and bottom-up cues. *IEEE Trans. Pattern Anal. Mach. Intell*, 2005.

[26] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[27] F. F. Li, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(4), 2006.

[28] Guangcan Liu, Zhouchen Lin, Xiaoou Tang, and Yong Yu. A hybrid graph model for unsupervised object segmentation. In *ICCV*, pages 1–8, 2007.

[29] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[30] Tomasz Malisiewicz and Alexei A. Efros. Improving spatial support for objects via multiple segmentations. In *BMVC*, 2007.

[31] Lopamudra Mukherjee, Vikas Singh, and Chuck R. Dyer. Half-integrality based algorithms for cosegmentation of images. In *CVPR*, 2009.

[32] M-E. Nilsback. *An Automatic Visual Flora – Segmentation and Classification of Flowers Images*. PhD thesis, University of Oxford, 2009.

[33] M. E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *CVPR*, 2006.

[34] M-E. Nilsback and A. Zisserman. Delving into the whorl of flower segmentation. In *BMVC*, 2007.

[35] M. E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, 2008.

[36] M-E. Nilsback and A. Zisserman. Delving deeper into the whorl of flower segmentation. *Image and Vision Computing*, 2009.

[37] P. Perona. Vision of a visipedia. *Proceedings of the IEEE*, 2010.

[38] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3), 2004.

[39] Carsten Rother, Thomas P. Minka, Andrew Blake, and Vladimir Kolmogorov. Cosegmentation of image pairs by histogram matching - incorporating a global constraint into mrfs. In *CVPR*, 2006.

[40] Bryan C. Russell, William T. Freeman, Alexei A. Efros, Josef Sivic, and Andrew Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, 2006.

[41] T. Saitoh, K. Aoki, and T. Kaneko. Automatic recognition of blooming flowers. In *ICPR*, 2004.

[42] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated subgradient SOlver for SVM. In *ICML*, volume 227, 2007.

[43] Jianbo Shi and Jitendra Malik. Normalized cut and image segmentation. Technical Report CSD-97-940, University of California, Berkeley, 1997.

[44] A. Vedaldi and B. Fulkerson. Vlfeat – an open and portable library of computer vision algorithms. In *ACM int. conf. on Multimedia*, 2010.

[45] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *ECCV*, 2008.

[46] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010.

[47] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Cosegmentation revisited: Models and optimization. In *ECCV*, 2010.

[48] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas S. Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

[49] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[50] John M. Winn and Nebojsa Jojic. Locus: Learning object classes with unsupervised segmentation. In *ICCV*, 2005.