An automatic visual Flora – segmentation and classification of flower images

Maria-Elena Nilsback

Robotics Research Group
Department of Engineering Science
University of Oxford

Hilary Term, 2009

Maria-Elena Nilsback
Brasenose College

Doctor of Philosophy
Hilary Term 2009

Abstract

This thesis investigates the problem of flower classification from images. Given an image and only the knowledge that it contains a flower, the task is to design a system that can determine the species of that flower despite changes in viewpoint, scale, illumination and natural deformations of the flower. We divide the problem into two challenges: segmentation and classification.

For segmentation, we develop an iterative segmentation scheme which is able to segment a flower given only a general colour model for flowers and no information about the flower in a particular image. This is done by iteratively fitting a geometric model and updating the colour model. The segmentation at each iteration is obtained by minimizing a Conditional Random Field using graphcuts.

For classification, we investigate how combining different features, which are carefully honed to describe different aspects, can improve performance. The aspects we describe are colour, texture and shape. In addition, we use the geometric model introduced for the segmentation to develop affine invariant geometric layout features.

We evaluate our method on a 17 category and a 102 category flower database. Despite significant amount of intra-class variation, small inter-class variation and the scale of the database, we demonstrate good classification performance and very good retrieval performance.

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Maria-Elena Nilsback, Brasenose College

Acknowledgements

This thesis would not have been possible without the enthusiasm and guidance of my supervisor Professor Andrew Zisserman. I am also grateful for the support and encouragement he has given me throughout these years.

Thanks to all the people in the Visual Geometry Group for making it such an enjoyable and fun environment. My very special thanks go to Mukta Prasad for travelling on this long journey with me and for being a good friend throughout. It would not have been the same without you. Thanks to Patrick Buehler, Oliver Woodford, Florian Schroff and James Philbin for their friendship and support. I would also like to thank Rob Fergus, Josef Sivic and Pawan Kumar. I am very grateful to Varun Gulshan for proofreading this thesis.

I would like to thank Dr. Barbara Caputo for leading me into the field of Computer Vision and Marie Curie Research Training Network Visiontrain for sponsoring this work. I am indebted to everyone who helped annotate the flower images.

Most importantly, I would like to thank my parents for letting me choose my own path, and my brother for always being sensible and supportive.

My final thanks go to Andrew Grieve for his support throughout these years and Marie Danielsson for always being there.

Contents

Chapter 1

Introduction

The objective of the work is to classify flowers from images. Flowers and the ability to identify them has been fascinating humans for hundreds of years. In the 18th century, Carl Linneaus proposed a hierarchical classification system for plants [63], which is still widely used. The taxonomy originally contained approximately 8000 plants, but has since been extended to encompass the more than $250,000$ flower species around the world. For flowers, classification may require dissection. However, even when an image is sufficient, classifying a flower may still need a guidebook, e.g. *Flora*, at hand, because of the level of expertize required. This is particularly frustrating now because with advances in digital and mobile technology it is easy to take pictures of flowers, but it is still difficult to find out what they are. Once you know the name you can find more information about a flower on the web, but the link between obtaining an image of a flower and acquiring its name is missing. Therefore, our aim is to create an automatic guide that classifies an image of a flower given its visual content, i.e. we aim to create *an automatic visual Flora*.

In addition to flower classification being an interesting problem, per se, it is also an interesting problem from a visual object classification point of view. Object classification, is a challenging problem within the field of computer vision that has seen much progress recently. As image based classification systems are improving, the task of classifying objects is moving to datasets with hundreds of categories. However, most work is focused on a large number of disparate categories e.g. faces, motorbikes, cows, dogs etc. In this thesis, we investigate the problem of recognizing a large number of classes *within* one category. Flower classification poses an extra challenge over categories such as bikes, cars, airplanes etc. because of the large similarity between classes, and also due to being non-rigid objects that can deform in many ways.

We aim to develop a framework that – although applied to the task of flower classification – can be easily extended to other similar classification tasks, i.e. classification of a large number of similar objects which require specialist knowledge. There are many such challenges including birds, butterflies, mushrooms, trees, cars, airplanes.

1.1 Problem Definition and Challenges

We will address the problem of flower classification. By flower classification, we mean identifying flowers as members of different flower species as determined by established flower taxonomy. Given an image of a flower, the task is to assign a species label to the image. We will do this by analyzing the visual content of the image. Not all species of flowers can be distinguished visually, hence we will limit ourselves to ones that can be. We also limit ourselves to flowers that commonly

occur in the United Kingdom.

The main problem for an object recognition system for any task is to be unaffected by irrelevant variations in the appearance of the object. For object classification, including flower classification, these appearance variations can be due to variations in imaging conditions, object deformations and variations between different object instances of a category.

Imaging Conditions:

- *Light variations* – can be due to different intensity or colour of the light of the light falling from a different angle. Ambient light also changes with the time of day. These changes in illumination can cause shadowing effects and intensity changes. In addition, flowers are often transparent to some degree. This means that the perceived flower colour will be different if light comes from behind or from in front of the flower, for example. Compare the images of Daffodils in figure 1.1. Although the flowers should have a very similar colour it is imaged quite differently.

Figure 1.1: Images of Daffodils taken under different illumination conditions. The flowers in all images should have very similar colour.

- *Viewpoint variations* – may result in significant changes of an object's appearance. It can lead to changes in the perceived size, shape, pose and rotation of the object. Figure 1.2 shows images of Crocuses from various viewpoints.

Figure 1.2: Images of Crocuses taken from different viewpoints. Viewpoint variations can render discriminative information invisible, such as the yellow stamen.

- *Occlusion* – occurs when part of an object is hidden by either itself or another object. Figure 1.3 shows some examples of typical occlusions in flower images. If characteristic parts, for example texture patterns, are occluded it might be difficult to classify a flower.

Figure 1.3: Images of flowers which are partly occluded by other objects.

- *Clutter* - An image often contains some background clutter. In flower images the backgrounds are often similar, hence the background clutter can mislead the classifier. Figure 1.4 shows some examples of background clutter. For

example, in the image of the sunflower bouquet the clutter is made up of other flower species. In this work we limit ourselves to images containing only one flower species.

Figure 1.4: Flower images with cluttered background.

Object deformations: A non-rigid object can deform in many ways that can change its appearance, for example the size of the object might change or a part might go missing. For flowers, petals might fall off or the flower might be closed or opened, for example. Figure 1.5 shows some natural flower deformations.

Figure 1.5: Natural deformations of flowers.

Intra-class vs inter-class variations: Intra-class variations are variations between objects within a category and inter-class variations are variations between objects across different categories. Large intra-class variation and small inter-class

variation makes the classification task more difficult. The natural object deformations and variations in imaging conditions all contribute do the variations within a class. In addition, the intra-class variations also depends on the natural variations within a class, i.e. the diversity of the species. For example, some flowers occur in many different colours, the petals of a flower might vary in shape and number, and some instances of a species might have distinctive texture whereas others do not. Figure 1.6 shows three flowers, where two of the flowers belong to the same category. One might just be able to tell that the left and right ones are Dandelions and the middle one is a Colt's Foot. Nevertheless, this example illustrates a case where the intra-class variation is large and the inter-class variation is small, and that even for a human this can be challenging task.

Figure 1.6: Three images from two different categories. The left and right images are both dandelions and the middle one is a colt's foot. The intra-class variation for the two dandelions is possible greater than the inter-class variation between each of the dandelions and the colt's foot.

1.2 Contributions and thesis outline

In this thesis we develop and evaluate a flower classification scheme. Figure 1.7 shows the general scheme we employ. The main contribution can be divided into three parts: *i*) segmentation and geometric model *ii*) classification *iii*) datasets.

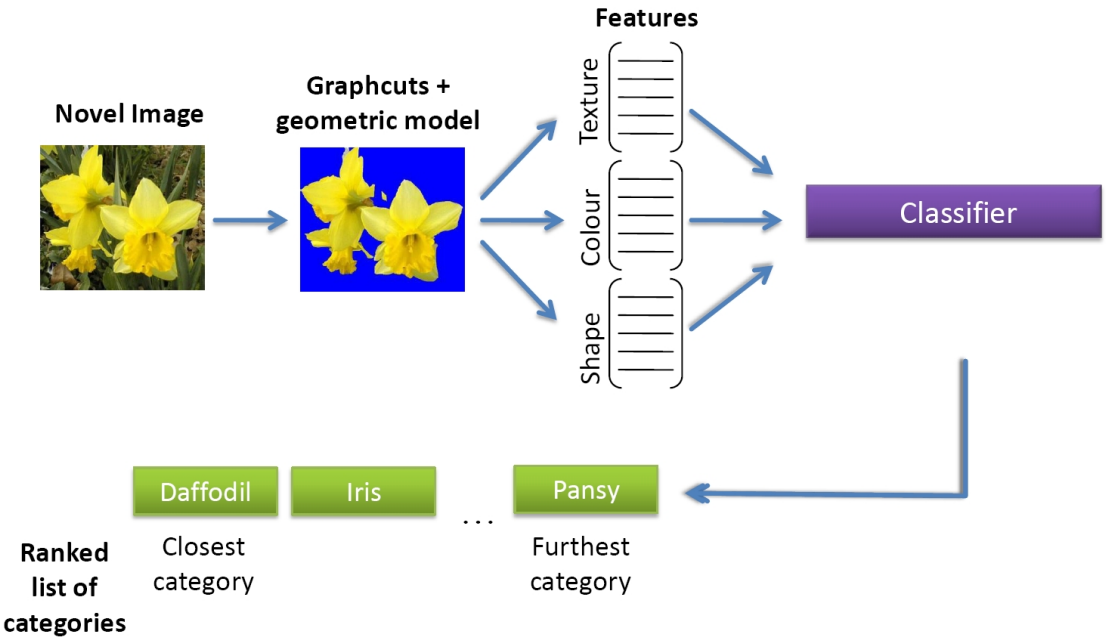


Figure 1.7: Flower classification scheme. Given a novel image, the flower is first segmented, features are then extracted, and finally the flower is classified.

1.2.1 Segmentation and Geometric Modelling

The first contribution addresses the problem of segmenting flower images. Flowers often occur in similar settings and we want to make sure that we are classifying the flower and not the background. We employ the segmentation as part of a classification pipeline and we want to be able to segment a flower from an image without any prior notion of which class it belongs to. Due to the sheer variety of flower classes, with varying shape, pose etc. we can not rely on shape based methods such as [51]. Even if we knew the class, segmentation using a class-based model could result in a bad segmentation for classes where the intra-class variation is large and/or not modelled properly. Instead, we propose an iterative segmentation scheme which starts off using a general colour distribution to propose a foreground/background

segmentation and uses this to initialize a loose geometric model. This can then be used to obtain an image-specific estimate of the colour distribution. The geometric model is applicable across a number of classes and viewpoints (see chapter 4). We thoroughly evaluate each step of the model and compare it to a segmentation obtained by minimizing a Conditional Random Field (CRF) using graphcuts and a baseline segmentation scheme. The geometric model obtained for segmentation is used in chapter 6 to develop rotation and scale invariant geometric layout features for flower classification.

1.2.2 Classification

The second contribution is the classification scheme. Previous work on flower classification has dealt with a small number of classes [91] (of the order of 30 different species). In chapter 5 we develop and thoroughly evaluate a classification scheme on around 20 classes. We then use our findings to scale our algorithm to classify approximately 100 classes. The challenge lies in determining suitable features to represent different flower aspects and then combining these in a flexible manner, such that the right features are used to discriminate between appropriate classes. For example, some flowers are characterized by their colour, e.g. Bluebells, some by their shape, e.g. Irises, and some by a combination of these, e.g. Dandelions. We investigate how using different features in a multiple kernel learning framework with a Support Vector Machine classifier can boost the performance. In chapter 6 we develop the geometric layout features and evaluate how these improve classification.

1.2.3 Datasets

The third contribution is the datasets. Although flower images are widely available on the internet, there is no site where there is a large collection of multiple images of several categories that can be used straight-forwardly for classification or segmentation. We have created two datasets: a smaller 17 species database and a large 102 species database, which both consists of common flowers in the UK. We have also created trimaps for 753 images which can be used for evaluation of segmentation schemes. A trimap is a pre-segmented image consisting of three regions: foreground, background and unknown. The datasets are presented in chapter 3.

In chapter 2 we review related work in the area, and in chapter 7 we draw conclusions of our work and propose future directions.

Chapter 2

Literature Review

The objective of recognizing objects has been a challenging task within computer vision since the 1960's [88]. Initially, the main focus was to recognize specific objects under varying viewpoints, and problems such as illumination variations, clutter, occlusion were avoided by inducing fairly constrained settings. Early work on object categories started in the 1980's on a limited number of classes, such as digits [56, 55] and faces [5, 50, 90, 95, 108], and in controlled settings. During the past decade the classification tasks undertaken have become more and more challenging, emphasis has been put on classifying object in natural settings [86, 105, 120] and the ability to classify many categories [11, 32, 41, 54, 112, 120, 121]. This increase in challenge has led to the development of more robust features descriptors and classifiers, and it has also become necessary to remove background clutter. The latter often requires some notion of what part of the image constitutes the object.

The following literature review is split into three parts: the first part (section 2.1) reviews ways of extracting features, the second part (section 2.2) reviews the

most relevant work on object classification and the third part (section 2.3) reviews the most relevant work on object segmentation.

2.1 Feature extraction

Features are used for describing the characteristics of objects [31, 101]. There is a lot of irrelevant data in images, for example the background. Good features describe discriminant properties of a category thus making it easier to distinguish between categories and the background [31]. Different features are useful for different tasks, for example, when distinguishing between lemons and bananas a shape describing feature could be useful, but when distinguishing between lemons and oranges a colour describing features would be more appropriate. Features can be either global, i.e. computed over the entire image or foreground region, or local, i.e. computed over interesting local regions of an image. Global features, for example colour histograms [103] and eigenspaces [81], are usually more sensitive to image variations, such as clutter, occlusion and viewpoint changes [110]. In general, extracting local features is a two step process; determining interesting regions and subsequently describing these regions. Traditionally they are computed around interest points, e.g. corner [44] or edge points [19], but local features can also be used to describe edges, curve-segments and contours [46]. Good detectors should be able to detect an interest point independent of imaging conditions [94]. An alternative is to densely sample the image [59, 117]. This could generate an excessive amount of feature points and is likely to be more sensitive to clutter, but there also no guarantee that an interest point detector will find points that are discriminative for a class [82]. Sliding window approaches have been show to work well for rigid objects and articulated objects

[26, 78, 115]. There are a wide variety of features available for object recognition. Here, we will limit ourselves to describing some of the more significant ones.

Feature detectors

The aim of the feature detector is to find interesting points and/or regions in an image. Given several images of object belonging to the same category, a good feature detector finds the same features in all images (where they exist) despite variations between images. When finding interest points the feature detector can also detect a patch surrounding the interest point from which the feature is to be extracted. It is desirable that this patch also is invariant to variations between images.

Canny Edge Detector. A Canny edge detector [19] is used for finding edges. It uses image gradients to find edge pixels, edgels, which are then linked into edges, Before computing the image gradient the image is smoothed with a Gaussian to remove noise and small edge segments.

Harris Corner Detector. The Harris detector [44] finds a corner by locating points where the image gradient is large in two directions. This is done by first computing the second moment matrix M, which for an image I is defined by:

$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, \tag{2.1}$$

where I_k is the derivative of the image I in the kth direction. The eigenvalues, λ_1 and λ_2, of M give an indication of the gradient variation at a point. The relationship between the eigenvalues can be used to describe the 'cornerness' of a point. Corner

points are determined by finding local maximas of the cornerness. The eigenvalues are independent of rotation. Thus, the Harris corner detector is *rotation invariant*. *Scale invariance* can be achieved by searching the scale space for local maximas [61]. This can be done by computing the difference of Gaussian (DoG) for different scales [64] or Laplacian of Gaussian (LoG) [61, 74]. The scale of the feature is given by the scale of the maxima. *Affine invariance* can be obtained by convolving the image with a non-uniform Gaussian kernel. The kernel is given by

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{2\pi\sqrt{|\Sigma|}} \exp -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}). \tag{2.2}$$

The kernel is defined by three parameters, which makes the search space very complex and additional constraints are therefore required. Lindeberg and Gårding [62] and Baumberg [4] explore affine adaptation of local regions by iteratively computing a non-uniform second moment matrix. Instead of convolving with a non-uniform Gaussian, an affine invariant region is found by iteratively adapting the region surrounding an interest point. The region is stretched along the dimension of the smaller eigenvalue and transformed to a uniform region. This approach generates an affine co-variant region and in order to make it affine invariant the rotation needs to be determined. Baumberg [4] uses training data to determine the rotation, whereas Schaffalitzky and Zisserman [92] use linear filter responses to estimate the invariants. The maxima might move a bit when doing this so Mikolajczyk and Schmid [75] propose redetecting the maxima at each iteration before updating the second moment matrix. This process is repeated until the ratio of eigenvalues is larger than a threshold. In [76] a thourough evaluation is given of these detectors.

Maximally Stable Extremal Regions. The Maximally Stable Extremal Regions (MSER,[72]) are obtained through a watershedding algorithm. The regions obtained are stable as a threshold varies across a range of intensity values. These regions are therefore brighter or darker than their surroundings. An ellipse is fitted to the region to make it affine invariant. Extension using other measures have also been proposed, for example colour [38].

Kadir-Brady Saliency Detector. Kadir and Brady's saliency detector [47] detects regions of maximum entropy for some image attribute (e.g. intensity or colour). Maxima of the entropy in the image are determined for a range of scales by using circular regions of multiple radii. A measure of saliency is obtained by weighting the entropy of the peak with a measure corresponding to the change of entropy over scales, such that regions which exhibit larger change in entropy over scale are given a higher measure of saliency. In [48], the descriptor is made affine invariant by iteratively adapting the circular region detected in order to maximize the entropy and saliency over scales.

Scale Invariant Feature Transform. In [64] interest points are detected by computing the convolution of the difference of Gaussian with an image in an efficient way. This involves generating a scale-space pyramid and detecting extrema by comparing each pixel to its neighbours in image and scale (figure 2.1).

2.1.1 Feature descriptors

This section explains how features are described given a point and/or a region. An invariant region descriptor describes the same region in two different images in the

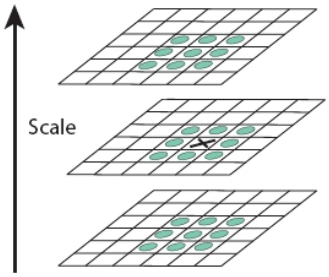


Figure 2.1: Scale space pyramid – image taken from [65]. Extrema of the DoG images are detected by comparing a pixel (marked with X) to its neighbours at the current and adjacent scales (marked with circles).

same way regardless of certain variations between images.

Shape-Context. The Shape-Context descriptor describes the shape of an object. Given a set of points (usually edges) in an image, the Shape-Context descriptor [6] can be used for describing the relationship between these points. This is done by creating log-polar histograms of the points relative to a reference point, for example the interest points (figure 2.2). In most applications the Shape-Context is sensitive to scale, but scale-invariance can be achieved by using multiple radii for the descriptor. Typically, the Shape-Context is computed from edges in the image, so it relies on robust edge detection.

Scale Invariant Feature Transform. Scale Invariant Feature Transforms (SIFT, [65]) descriptors are histograms of local gradient orientations. Given a patch, a histogram of gradient orientations is computed on a 4×4 grid (figure 2.3). Typically 8 orientations bins are used, resulting in 128 dimensional histogram for each patch. The gradient magnitudes are weighted by a Gaussian. The invariance of the descriptor depends on the invariance of the feature detector. The affine-invariant Harris

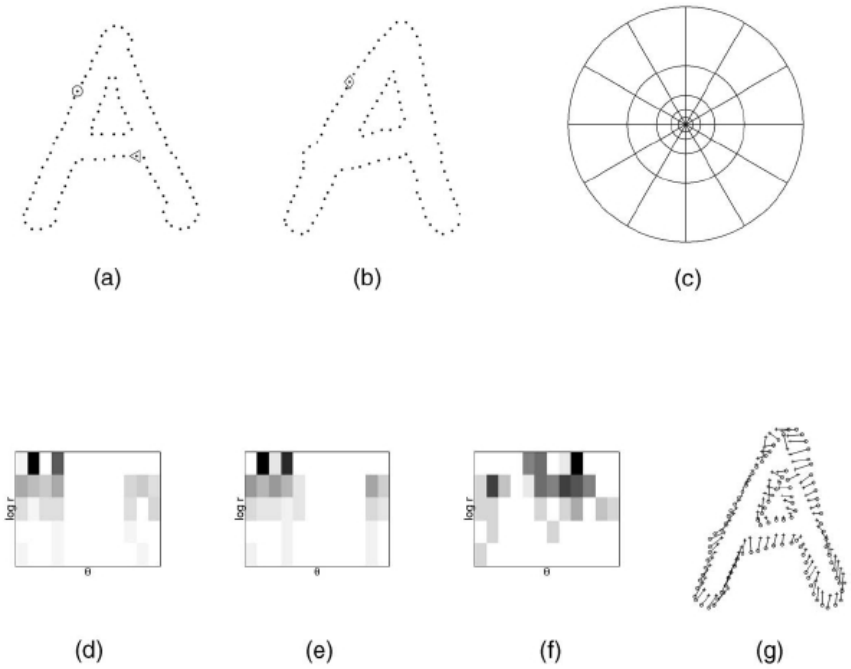


Figure 2.2: Shape Context computation and matching – figure taken from [6]. a) and b) Sampled edge points of two shapes. c) Diagram of log-polar histogram bins used in computing the shape contexts. d), e) and f) Example shape contexts for the points marked in a) and b). g) Correspondences.

detector is a good choice of detector as the scale and rotation of the descriptor can be determined by the detector, which results in an affine-invariant SIFT feature. Alternatively, one can compute the SIFT-features on a circular region, thus making the descriptor rotation and scale invariant. Circular regions are commonly used when the features are densely sampled.

Textons Textons are used to describe the texture of an object. Leung and Malik [60] compute textons using a filter bank. The response to 48 different filters are computed for each pixel and concatenated into a 48 dimensional vector. The vectors are then clustered using K-means clustering. Varma and Zisserman [113] use a much

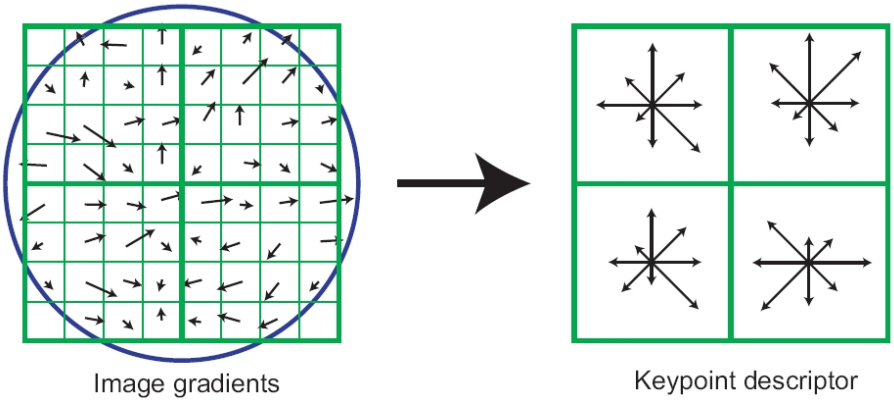


Figure 2.3: SIFT descriptor– image taken from [65]. A keypoint descriptor is created by first computing the gradient magnitude and orientation at each point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4×4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitude in that direction within the region. The figure show a 2×2 descriptor array, whereas most implementation use a 4×4 array.

smaller filter bank of so called Maximum Response (MR) filters. The MR4 filter bank consists of 4 filters: 2 rotationally invariant (Laplacian and Gaussian filters) and 2 directional ones (bar and edge filters) as shown in figure 2.4. The bar and edge filters are computed at 6 different orientations but only the maximum response is kept. The MR8 filter bank computes the directional filters at 3 different scales, and thus consists of 8 filters. In [114] the authors show that large filter banks are not necessary and textons consisting of local neighbourhoods can perform equally well for classification.

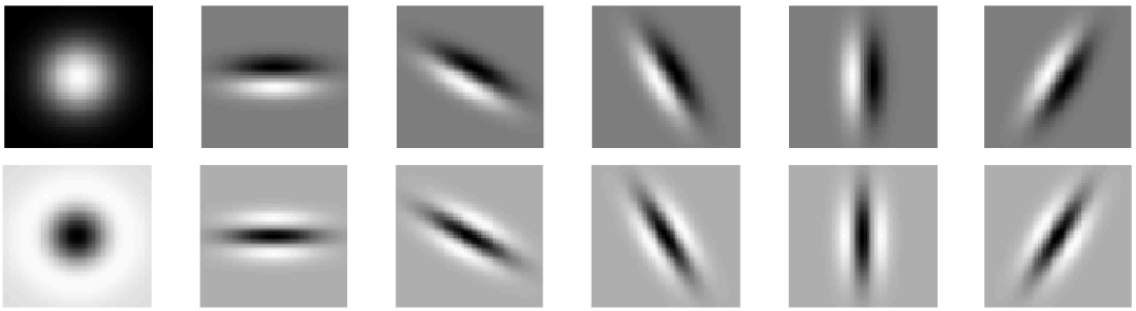


Figure 2.4: MR4 filter bank – figure taken from [114]

2.2 Classification

Suppose we have K categories. Given a set of features, $\boldsymbol{x}$, from a novel image, we want to determine the class label of the image, c^*. The probability for each class is $P(c_k|\boldsymbol{x})$. The class labels is given by the highest probability, i.e.

$$c* = \underset{k}{\operatorname{argmax}}\, P(c_k|\boldsymbol{x}), k = 1, 2, ...K. \tag{2.3}$$

We do not know the probability $P(c_k|\boldsymbol{x})$ *a priori*. The task of the classifier is to estimate this probability, which requires learning a model from some training data [7, 31]. Classifiers can roughly be divided into generative and discriminative classifiers.

Generative classifiers: aim to model what is common between object of the same category. Instead of modelling $P(c_k|\boldsymbol{x})$ directly, generative models estimate the likelihood $P(\boldsymbol{x}|c_k)$ and classify images by obtaining $P(c_k|\boldsymbol{x})$ using Bayes' rule. Gaussian mixture models (GMMs, [7]) is an example of a generative model. Given

M Gaussians, the mixture of Gaussians can be written as

$$P(\boldsymbol{x}|c_k) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m), \tag{2.4}$$

where the mixing coefficient π_m must satisfy $0 \leq \pi_m \leq 1$ and $\sum_m \pi_m = 1$. The expectation-maximization algorithm (EM, [28, 73]) is a maximum likelihood algorithm that can be used to fit the model to the training data.

Discriminative classifiers: aim to model the difference between categories. This means that they try to find discriminant surfaces separating object classes. Discriminative classifiers model the probability, $P(c_k|\boldsymbol{x})$, directly. Simple linear classifier and nearest neighbour classifiers are both examples of discriminative classifiers.

Support Vector Machines (SVMs, [24, 96, 111]) are popular discriminative classifiers that learn the decision boundary with the largest margin to the training data. This makes it more robust to outliers. The decision function is given by

$$f(\boldsymbol{x}) = sgn(\sum_{i=1}^{m} \alpha_i y_i K(\boldsymbol{x_i}, \boldsymbol{x}) + b), \tag{2.5}$$

where $K(\boldsymbol{x_i}, \boldsymbol{x})$ is the value of a kernel function for the training sample $\boldsymbol{x_i}$ and the test sample $\boldsymbol{x}$. The kernel is a way of computing a dot product in a high dimensional space. The training samples with $\alpha_i > 0$ are the support vectors.

An alternative to the SVM classifier is the AdaBoost classifier [39]. Boosting [93] is the process in which a strong classifier, H, is created by combining M weak classifiers, h_m.

$$H(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m h_m(\boldsymbol{x}) \tag{2.6}$$

The weights are learnt in an iterative manner. At each iteration the classifier with the smallest error is selected and the weights for the misclassified samples are increased. This approach was first introduced in Computer Vision by [115] for face detection. For multiclass problems each strong classifier is learnt independently for each class. In [107] a method is proposed for speeding up learning by sharing features across the classifiers.

Random Forest classifiers [2, 15] are decision tree based classifiers. The key idea is to generate multiple decision trees where the initial set of features is chosen at random from a very large pool of features. Multiple trees are constructed for different random samples from the feature pool. Each tree is learnt by choosing the feature which maximizes information gain at each node. In [58] the tests performed at the nodes are simple binary tests based on intensity differences between two pixels, $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ (figure 2.5). At each leaf node the posterior probabilities $P_{\eta(l,\boldsymbol{p})}(Y(\boldsymbol{p}) = c)$ are stored, where c is the class and $\eta(l, \boldsymbol{p})$ is the leaf node of tree T_l reached by the patch $\boldsymbol{p}$. The trees are combined by averaging the posterior probabilities over all trees. They have been show to perform as well as SVMs [12] and have been used for many classification tasks [22, 29, 68, 79, 98, 118, 119].

Multiple Kernel Learning (MKL, [3, 53, 102]) is an algorithm for obtaining an optimum linear combination of kernels. Given a set of symmetric positive semidefinite kernels $\boldsymbol{K}_1, ..., \boldsymbol{K}_m$ the task is to find the optimal combination $\boldsymbol{K}_{opt} = \sum_j d_k \boldsymbol{K}_k$.

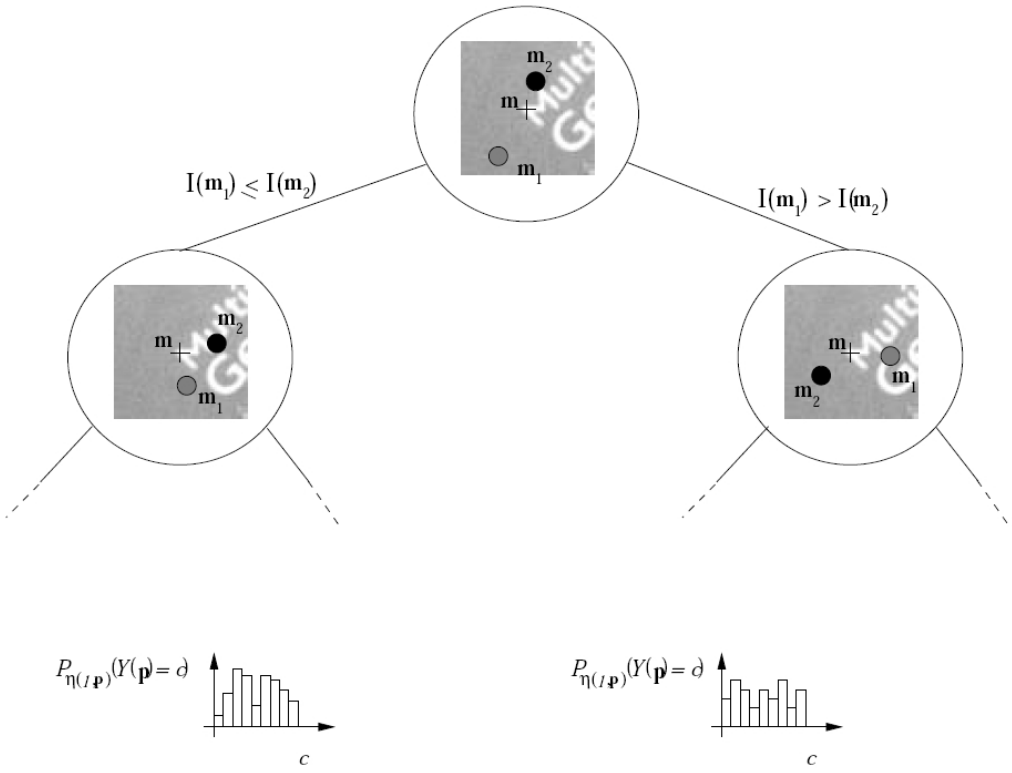


Figure 2.5: Random trees – figure taken from [58]

Varma and Ray have applied MKL to object classification [112]. In their work the weights are optimized in an SVM framework. The primal cost function is given by:

$$\min_{\boldsymbol{w},\boldsymbol{d},\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i + \langle \boldsymbol{\sigma}, \boldsymbol{d} \rangle \tag{2.7}$$

$$\text{s.t.} \quad y_i \left(\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i) + b \right) \geq 1 - \xi_i, \quad \forall i \tag{2.8}$$

$$\boldsymbol{\xi} \geq 0, \ \boldsymbol{d} \geq 0, \ \boldsymbol{A}\boldsymbol{d} \geq \boldsymbol{p} \tag{2.9}$$

$$\text{where} \quad \boldsymbol{\phi}(\boldsymbol{x}_i) = \sum_k d_k \phi_k(\boldsymbol{x}_i) \tag{2.10}$$

The objective function (2.7) is the same as the standard l_1 C-SVM objective

function with the added l_1 regularization term on the weights $\boldsymbol{d}$. The parameter σ encodes prior preferences for the descriptors. The constraints on $\boldsymbol{d}$ are added to restrict the weights to take on interpretable values and also to allow us to encode prior knowledge. In addition, the final condition is added to encode the linear weighted combination $\boldsymbol{K}_{opt} = \sum_j d_k \boldsymbol{K}_k$.

We focus next on 'bag-of-words' methods, which are methods based on forming a visual vocabulary of local features. These methods preserve no spatial information. We also review approaches incorporating spatial layout as an extension to the 'bag-of-words' methods.

2.2.1 Bag-of-words

Bag-of-words models are based on a representation from statistical text analysis, where a document, d_j, is represented by a set of words, w_i, taken from some vocabulary. One aim of text classification is to determine the topic of a document given the words therein. The bag-of-words model refers to the fact that the documents are represented by a collection of words with no information of where in the document they come from.

The visual analogy of this is to determine the class of an image given a set of visual words. A vocabulary of codewords is obtained by clustering visual words from a set of training images. The codewords are the cluster centres. Each feature in an image can then be vector quantized to words in the vocabulary. Hence, each image can be represented by a frequency histogram denoting the number of occurrences of each codeword in the vocabulary [100].

Csurka *et. al.* [25] use a bag-of-words model for classification. The authors com-

pare using a bag-of-words approach with a discriminative classifier using SVMs and a generative classifier using Bayes' Rule. They show that the SVM outperforms the Bayesian classifier on a seven category database of largely different categories, such as faces, buildings, trees, cars etc. SVMs have been very successful in conjunction with bag-of-words model [112, 120, 121].

Object classification usually involves labelling the images and sometimes further manual preprocessing, for example segmentation of the image. With bag-of-words models the data labelling only has to involve assigning one class label per image. No object segmentation or bounding box is necessary, but because we do not know whether the features are coming from the background there is a risk that we might be classifying an object based on the background. Sometimes this might not be a bad thing, as the context can give useful information about an object [121]. For example, if we have sky in the background we might be much more likely to be looking at an airplane than a car. However, for a wide variety of categories the context is a very unreliable indicator of the object.

There have been several approaches to automatically remove words and features not belonging to the object [30]. Sivic *et al.* [100] apply a bag-of-words model to object retrieval of particular objects from videos. In this work the authors normalize each bin in the frequency histogram by the frequency of the corresponding word in the training set. In text retrieval, this is known a 'term frequency-inverse document frequency', *tf-idf*. The authors also incorporate a stop list to suppress words that occur very frequently. The approach increases the influence of less frequently occurring words and is based on the idea that these words will be more discriminative. This is not necessarily true. For example, objects like flowers share a lot of similar features, and although the presence of these features is not discriminative, the

relative frequency of these can be.

Another approach is to use generative models to automatically discover object categories. In natural language processing Probabilistic Semantic Analysis (pLSA, [45]) and Latent Dirichlet Allocation (LDA, [8]) have been used for this purpose. The idea behind both methods is that the occurrence of a word w_i in a document d_j depends on a latent topic variable z_k. A category could consist of a mixture of topics.

In pLSA the probability $P(w_i|d_j)$ is given by marginalizing over the topics z_k, i.e.,

$$P(w_i|d_j) = \sum_{k=1}^{K} P(z_k|d_j)P(w_i|z_k) \tag{2.11}$$

$P(z_k|dj)$ is the probability of topics z_k occurring in document d_j and $P(w_i|z_k)$ is the probability of word w_i occurring in a topic z_k. These probabilities can be estimated by maximizing the function

$$L = \prod_{i=1}^{M}\prod_{j=1}^{N} P(w_i|d_j)^{n(w_i,d_j)}, \tag{2.12}$$

where M is the size of the vocabulary, N is the number of documents and $n(w_i|d_j)$ is the frequency of a word w_i in a document d_j. Sivic *et al.* [99] use pLSA to classify 4 different categories. The pLSA is trained using an additional background class. The authors show that given a topic, the visual words with high probability, are largely coming from the object corresponding to that topic. In [10] pLSA is used for scene classification. One of the drawbacks of pLSA is that the number of topics need to be fixed.

LDA is an extension of pLSA, where the mixture weights $P(z_k|d_j)$ are determined

by a latent random variable θ sampled from a Dirichlet distribution. It was used in [33] for scene classification.

2.2.2 Spatial Layout

So far we have dealt with bag-of-words model which do not preserve any spatial information. We have also looked at methods that implicitly remove/reduce influence of words belonging to the background. Now, we will look at methods that explicitly model spatial information.

In the 1970's Fischler and Elschlager [36] introduced the concept of Pictorial Structures, which is a collection of parts held together by springs. The matching cost is determined by both the patch appearance and the deformation of the springs. Felzenszwalb and Huttenlocher [34] have extended this work by defining an efficient matching algorithm and introducing a Bayesian formulation for the matching problem. In the late 1990's, Burl *et al.* [16, 18, 17] used a constellation of parts to model faces. In their work, manual labelling of parts in the training images is required. Weber *et al.* [116] extend this work by learning a model for the shape and and a model for the appearance of the parts. Fergus *et al.* [35] improve this work by simultaneously modelling the appearance, shape, relative scale and location of parts by a Gaussian distribution. The parameters for the Gaussian distributions are learnt from each category automatically without the need to manually segment the training images, but the number of parts need to be supplied. Classification is done by first finding hypothetical locations of parts and then determining whether given this hypothesis the object is more likely to be foreground or background. This method can handle clutter and occlusion by modelling each part separately, but is

sensitive to pose variations.

Leibe *et al.* [57] incorporate a Hough-like voting scheme to localize an object in an image. In this work a visual vocabulary is generated for each class. The authors then use a probabilistic framework to learn the probability of the location of each word relative to the centre of the object. This requires that a bounding box is supplied for the training images. During testing each detected patch votes for the centre of the object. It can handle scale changes and cluttered background, but is sensitive to view-point variations. Shotton *et al.* [97] and Opelt *et al.* [83] propose two similar methods which extend this work. Instead of patches they use boundary fragments together with relative information about the centroid position and then learn class-specific boundary fragments using boosting.

The pyramid match kernel was introduced by Grauman and Darrell [40] as a way of matching histograms of features at multiple resolutions. The basic intuition is that tighter correspondences are given stronger weight. Lazebnik *et al.* [54] extended this work to the spatial domain by subdividing into grid cells and computing the bag-of-words frequency histogram for each cell. In a similar manner to [40] matches in coarser grids are given less weight than matches in finer grids. The approach does not require any manual segmentation of training data, but it is sensitive to scale, since the grid is applied to the entire image and thus depends on the image size rather than the object size.

Marszalek and Schmid [69] build on both the ideas of patches giving information about the location of the object and spatial weighting. They use ground-truth segmentation as training data and train their system only on features from the foreground. Each training feature is associated with a segmentation mask. Classification is done by matching features to the N closest features from the training

data and then transforming the segmentation mask for each training feature to the orientation and scale of the feature in the test image. Finally, a score mask map is obtained that describes the likelihood of each pixel belonging to the object. A bag-of-words frequency histogram is then created by weighting the features according to the likelihood for their location, such that foreground features have stronger influence on the segmentation.

Another way of preserving spatial layout is using features such as Histogram of Gradient Orientations (HOGs, [26]). HOGs are computed by dividing the image into cells and computing a histogram of gradient orientations for each cell. The local histograms are then concatenated into one feature vector. In [26] the local histograms are contrast normalized. The authors developed the descriptor for pedestrian detection where there is not much pose variation, other than for the extremities. It is not straightforwardly able to deal with changes in rotation and scale. That would require some preprocessing of the image, such as pre-segmentation.

2.3 Segmentation

In this thesis we address the problem of automatically segmenting flowers using colour as a step towards improving classification. We are interested in segmentation that results in a foreground (object) and background segmentation. Here we review the most relevant work on image segmentation, and describe related work on flower segmentation. We will start by reviewing simple pixel labelling methods using only pixel appearance to assign a label to a pixel, including the use of Gaussian Mixture Models and recent work on flower segmentation by Das *et al.* [27]. We will then review contour-based methods which try to find the boundary of an object by locally

minimizing an energy function so that the segmentation boundaries align with strong gradients (i.e. edges) in the image. These include Snakes [49], level sets [84] and Intelligent Scissors [80]. Finally, we will review graph-based pixel labelling methods, where a global energy function is defined depending on both appearance and image gradients. We will focus on work using Conditional Random Fields by Boykov and Jolly [13], and the extensions of this method provided by GrabCut [89] and `ObjCut` [51].

2.3.1 Appearance based methods

The aim of appearance based methods is to determine whether a pixel is more likely to belong to the foreground or the background given its appearance alone. The simplest appearance based model would simply partition an image into regions with coherent texture or colour [87], for most real images this would results in many different segments.

By learning a model for the foreground and background the segmentation can be made more robust to small colour variations, such as variation in lighting. A Gaussian Mixture Model could, for example, be used to create a model for the foreground and background distribution. For flower segmentation Das *et. al.* [27] learn an image specific foreground and background model. The background colour model is learnt from dominant colours in the periphery of the image. These colours are removed from a pool of possible foreground colours, and the remaining colours used to obtain a foreground region (see figure 2.6). The region is accepted provided it satisfies certain spatial conditions.

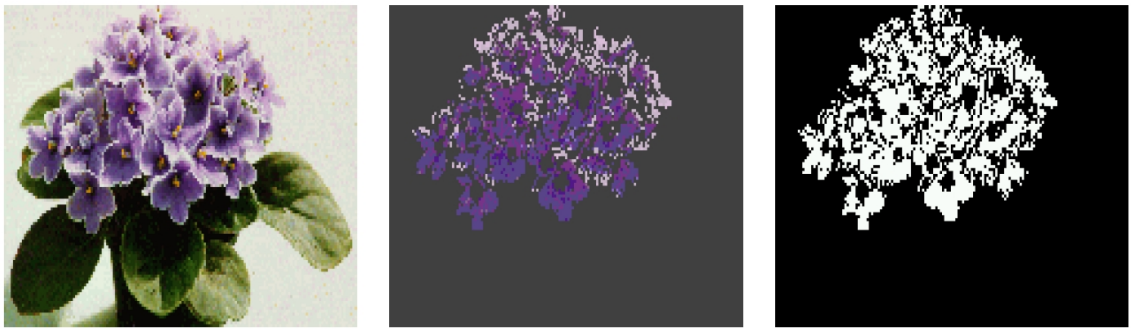


Figure 2.6: Flower segmentation (figure taken from [27]) – (a) original image, (b) image left after deleting non-flower colours and (c) largest valid segment.

2.3.2 Contour based methods

In contour based approaches the idea is to obtain a segmentation from an initialized contour by evolving the contour towards the true object boundary. This is done by locally minimizing an energy function which is subject to some image constraints. Let Ω be a bounded open subset of $\Re^2$, with $\partial\Omega$ its boundary. Let $I : \bar{\Omega} \rightarrow \Re$ be a given image. In the work on Snakes by Kass *et al.* [49] the contour is an explicitly parametrized curve $C(s) : [0, 1] \rightarrow \Re^2$, which is initialized outside the boundary and which is evolved by minimizing the energy:

$$E(C) = \alpha \int_0^1 |C'(s)|^2 ds + \beta \int_0^1 |C''(s)| ds - \lambda |\nabla I(C(s))|^2 \tag{2.13}$$

α, β and λ are non-negative parameters that control the trade-off between smoothness of the curve, determined by the first two terms, and the alignment of the curve to large image gradients, determined by the last term. In order to reduce sensitivity to noise the image can be convolved with a Gaussian before computing the image gradients. The Snakes approach relies on the contour having a

good initialization so that it does not get stuck in local minima and it does not automatically allow for topological changes of the contour. Another way of representing contours is using implicit representations like level sets [84]. The contour is then represented as the zero level curve of a level-set function $\phi(x, y) : \Omega \rightarrow \Re$, i.e. $C = (x, y)|\phi(x, y) = 0$, i.e. inside the curve $\phi(x, y) < 0$ and outside the curve $\phi(x, y) > 0$. The contour is thus evolved by minimizing $\phi(x, y)$. The level set function at time t is given by $\phi(x, y, t)$ and the contour is evolved by gradient descent over time. Caselles *et al.* [20], for example, propose a geometric active contour model where the gradient descent depends on the mean curvature motion. Similar to Snakes, the level set methods depend on image gradients and can get stuck in local minima, but level sets have the advantage of being able to undergo topological changes.

Chan and Vese [21] propose an active contour method which does not use the image gradient to evolve the contours. Instead their method takes into account *region* information. Given an image, I, consisting of two regions of approximately piecewise constant intensities, i_1 and i_2, and let the C_0 denote the boundary between these regions, then a fitting term can be defined as:

$$F_1(C) + F_2(C) = \int_{inside(C)} |I(x, y) - c_1|^2 dxdy + \int_{outside(C)} |I(x, y) - c_2|^2 dxdy, \quad (2.14)$$

where c_1 and c_2 are the averages inside and outside the contour, C, respectively. It is easy to see that in this case C_0 will minimize 2.14 and $c_1 \approx i_1$ and $c_2 \approx i_2$. The energy functional is then defined as:

$$F(c_1, c_2, C) = \mu \cdot Length(C) + \lambda_1 F_1(C) + \lambda_2 F_2(C), \quad (2.15)$$

where the first term is a regularization term. Restricting (2.15) to piece-wise constant functions, allows it to be minimized by finding the zero level curve of $\phi(x, y)$. The benefit of this work is that relies not just on the edge information but on the entire region and is thus less sensitive to poor initialization and weak edges. Mortenson *et. al.* [80] proposed 'intelligent scissors' as a different way of evolving contours towards object boundaries, by creating a local cost function between neighbouring pixels which depends on edge information. The basic idea is that a user selects a seed point on the object boundary and then moves the mouse roughly along the boundary. The shortest path is then computed from the seed point, thus allowing the boundary to snap to the object. This is done by for each pixel, p, computing a local cost, $l(p, q)$, to each of its neighbours q, where

$$l(p, q) = w_Z \cdot f_Z(q) + w_G \cdot f_G(q) + w_D \cdot f_D(p, q), \tag{2.16}$$

f_Z measures the Laplacian Zero Crossing at q, f_G measures the gradient magnitude at q and f_D measures how well the edge between p and q is aligned with the gradient direction. The w's are the relative weights between terms. This is equivalent to generating a graph with edge weights $l(p, q)$, and can then be minimized using Djikstra's shortest path algorithm [23]. For complicated boundaries this method requires many seed points on the boundary to be selected. Saitoh *et. al.* [91] use intelligent scissors for flower segmentation. In their work, an alternative cost function is used which normalizes $l(p, q)$ by the length of the path. This allows the path to follow very jagged boundaries, such as boundaries with many petals, rather than cutting the path short. In addition, they automatically detect potential seed points, so that there is minimal user interaction. Their method relies on the flower

being centred and the sharpest edge being along the boundary of the flower. Neither of these active contour models would deal with textured surfaces very well.

2.3.3 Contour and appearance based methods

So far the methods we have reviewed for segmentation have found a local minimum. Now we review work based on the graph cut segmentation work by Boykov and Jolly [13], which defines a single energy function depending on both appearance and edge properties of an image. Furthermore, on the contrary to level set methods, this method finds a global minima. The problem is formulated as a pixel labelling problem, where the segmentation of an image is represented as a binary vector $\mathbf{x}$. The energy function is then given by:

$$E(\mathbf{x}) = \lambda \cdot R(\mathbf{x}) + B(\mathbf{x}), \tag{2.17}$$

where

$$R(\mathbf{x}) = \sum_{p \in P} D_p(x_p) \tag{2.18}$$

$$B(\mathbf{x}) = \sum_{\{p,q\} \in N} V_{\{p,q\}} \cdot (x_p == x_q) \tag{2.19}$$

p is the pixel index over the set of pixels, P, and N is the set of unordered pairs, $\{p, q\}$. λ determines the trade-off between the appearance properties (e.g. colour likelihood), $R(\mathbf{x})$ and the boundary properties, $B(\mathbf{x})$. $B(\mathbf{x})$ penalizes assigning different labels to neighbouring pixels with a cost depending on the image edges, so that there is a larger cost for assigning different labels to neighbouring pixels if they are very similar and a small cost if they are not. The energy can be minimized

using the minimum cut algorithm by [14]. The equivalent graph-cut problem is constructed by creating a graph where the edge cost to the terminal is determined by the unary term, $R(\mathbf{x})$, and the edge cost between nodes is determined by the pairwise term, $B(\mathbf{x})$.

The approach by Boykov and Jolly will find the minimum cut given an estimate of the colour likelihood. If the model describing for the colour likelihood is incomplete or wrong, the segmentation will most likely fail. The GrabCut extension by Rother *et. al.* [89] proposed a way of improving segmentation by minimum user interaction, by iteratively segmenting and re-estimating the appearance model. Gaussian Mixture Models (GMMs) are used to model the colour likelihood. The user selects some pixels in the background to initialize the GMM. The user can then at each iteration fix the label of additional pixels as foreground or background after which the GMM is updated and the image is resegmented.

Another extension is provided by ObjCut [51], where a shape prior is added to the energy function. Given an object category specific shape model, Ω, and an image , $\mathbf{z}$, the energy function now becomes

$$E(\mathbf{x}; \Omega, \Theta, \mathbf{z}) = \sum_{p \in P} D_p(x_p) + \sum_{\{p,q\} \in N} V_{\{p,q\}} \cdot (x_p == x_q) + \sum_{p \in P} S_p(x_p; \Omega) \qquad (2.20)$$

ObjCut tries to maximize the probability

$$p(\mathbf{x}|\mathbf{z}, \Omega, \Theta) = \frac{1}{Z(\Theta, \Omega)} \exp(-E(\mathbf{x}; \Omega, \Theta, \mathbf{z})) \qquad (2.21)$$

The shape model, Ω, is treated as a latent variable and marginalized out using an

Expectation-Maximization (EM) approach. In the E-step the shape model is fitted to the image, which provides us with a number of samples for different putative poses of the object. Given these samples Ω can be marginalized out in the M-step.

Chapter 3

Datasets

This chapter outlines the details of each dataset used in this thesis and discusses the difficulty of each of them. There are many flower websites, but most only have one or two examples per category, and this is not enough to be used for classification. We have therefore created two datasets by gathering images from various websites, with some supplementary images from our own photographs. The first dataset is a smaller one, consisting of 17 different flower categories, and the second dataset is much larger, consisting of 102 different categories of flowers common to the UK.

3.1 17 Category Dataset

This dataset consists of 17 species of flowers with 80 images of each, totalling 1360 images. Figures 3.1 and 3.2 show the different categories. There are species that have very unique visual appearance, for example Fritillaries and Tiger Lilies, as well as species with very similar appearance, such as Dandelions and Colt's Feet.

The flower categories are deliberately chosen to have some ambiguity on each aspect. For example, some categories can not be distinguished by colour alone (e.g. Dandelion and Buttercup), others can not be distinguished by shape alone (e.g. Daffodils and Windflowers). The figures also show typical variations for each category. There are large viewpoint, scale and illumination variations. The large intra-class variability and the sometimes small inter-class variability makes this database challenging enough for developing our methods. The images are rescaled so that the smallest dimension is 500 pixels. The database is publicly available at the following URL: `http://www.robots.ox.ac.uk/~vgg/data/flowers/`. There are 10 different training, validation and test set splits. For each splits the training set and validation set consist of 20 images per class (totalling 340 images each) and the test set consists of 40 images per class.

Figure 3.1: Images from the 17 category database - Part 1. Each row shows 5 images from each category. Each category shows pose variation, scale changes, illumination variations, large intra-class variations and self-occlusion.

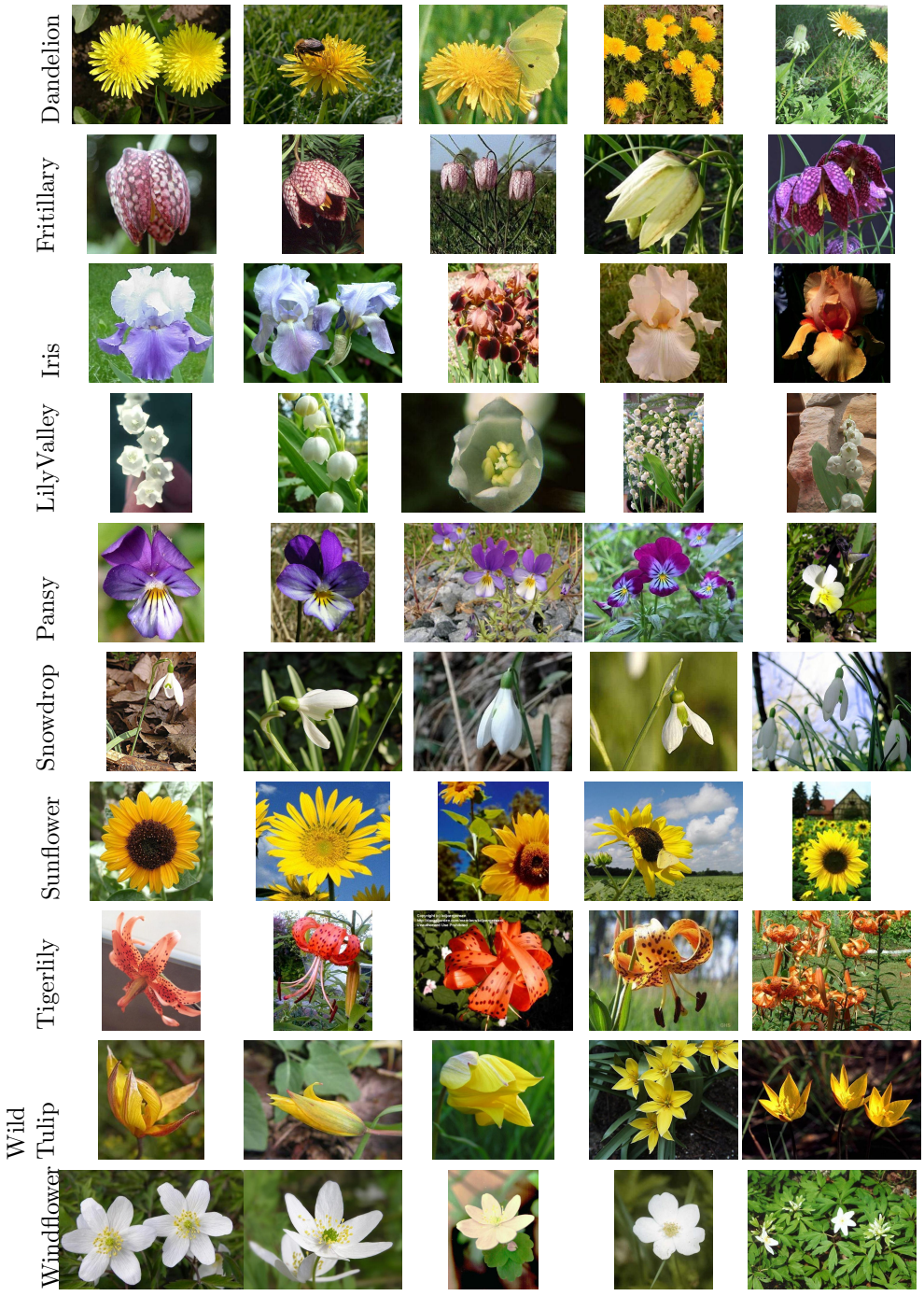


Figure 3.2: Images from the 17 category database - Part 2. Each row shows 5 images from each category. Each category shows pose variation, scale changes, illumination variations, large intra-class variations and self-occlusion.

3.1.1 Ground Truth

753 images have been ground truth labelled into foreground and background regions using a trimap. A trimap is a pre-segmented image consisting of three regions: foreground, background and unknown. In chapter 4, these will be used to evaluate segmentation accuracy. Trimaps are not provided for images in the dataset that contain: fields of flowers; or very small flowers; or that are too subsampled; or where there is no clear foreground region. Following these criteria, four of the classes (snowdrops, lily of the valley's, cowslips and bluebells) have insufficient images and are not ground truth labelled. This leaves 13 flower classes with a total of 753 images.

In the trimap a small proportion of pixels and certain regions are left unlabelled – principally the pixels very close to the flower boundary, and regions that are difficult to classify, e.g. a small out of focus flower in the background. Not labelling pixels close to the flower boundary greatly simplifies the annotation procedure as precisely selecting pixels near a boundary is painfully slow. Figures 3.3 and 3.4 show examples of trimap labellings for some images from different categories. The red area is labelled foreground, the green area background, and the black area is unlabelled. The unlabelled pixels are not used in the evaluation. The trimaps are also available at `http://www.robots.ox.ac.uk/~vgg/data/flowers/`.

Figure 3.3: Images and their corresponding ground truth trimaps – Part 1. The red region is foreground, and the green region background. The black region is unlabelled, and is not used in assessing performance. Regions close to boundaries and regions which are not clearly foreground or background are left unlabelled.

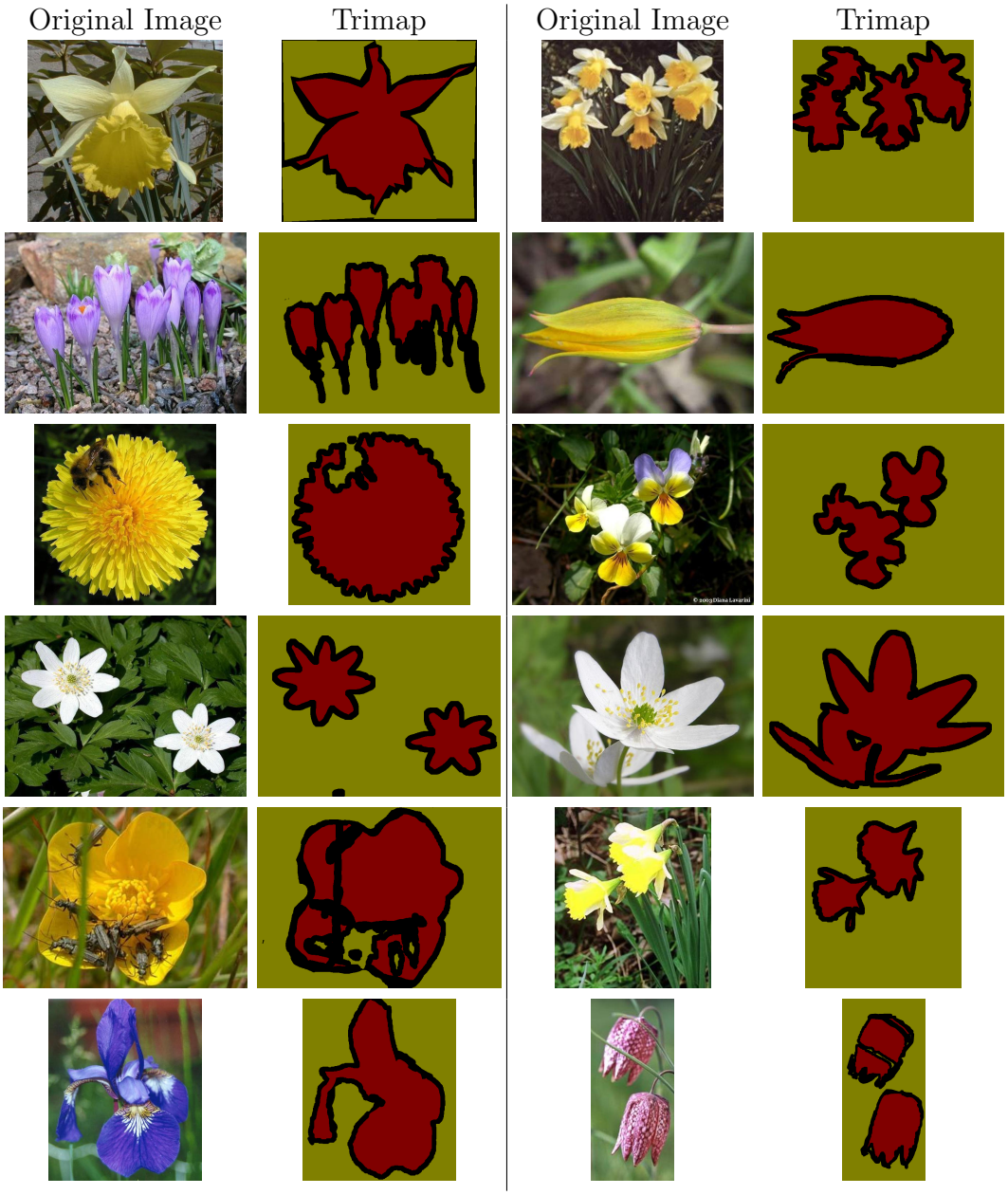


Figure 3.4: Images and their corresponding trimaps of ground truth labelling – Part 2. The red region is foreground, and the green region background. The black region is unlabelled, and is not used in assessing performance. Regions close to boundaries and regions which are not clearly foreground or background are left unlabelled.

3.2 102 category dataset

This larger dataset consists of 8189 images divided into 102 flower classes. These are chosen to be flowers commonly occurring in the United Kingdom according to the Royal Horticultural Society [1]. Each class consists of between 40 and 250 images. Figures 3.5 - 3.7 show one example per class and the number of images for each class. Passion flower has the greatest number of images and Eustoma, Mexican Aster, Celosia, Moon Orchid, Canterbury Bells and Primrose have the least, i.e. 40 per class. It is a much more difficult database than the 17 category database, not only because of the greater number of classes, but also because of the few samples per class and the large inter-class similarity between many of the classes. Figure 3.8 shows some typical variations within a category. The images are rescaled so that the smallest dimension of the image is 500 pixels. The database can be found at the following URL: `http://www.robots.ox.ac.uk/~vgg/data/flowers/`. The dataset is divided into a training set, a validation set and a test set. The training set and validation set each consist of 10 images per class (totalling 1020 images each). The test set consists of the remaining 6149 images (minimum 20 per class).

	Category	Nr of images		Category	Nr of images
	alpine sea holly	43		buttercup	71
	anthurium	105		californian poppy	102
	artichoke	78		camellia	91
	azalea	96		canna lily	82
	ball moss	46		canterbury bells	40
	balloon flower	49		cape flower	108
	barbeton daisy	127		carnation	52
	bearded iris	54		cautleya spicata	50
	bee balm	66		clematis	112
	bird of paradise	85		colt's foot	87
	bishop of llandaff	109		columbine	86
	black-eyed susan	54		common dandelion	92
	blackberry lily	48		corn poppy	41
	blanket flower	49		cyclamen	154
	bolero deep blue	40		daffodil	59
	bougainvillea	128		desert-rose	63
	bromelia	63		english marigold	65

Figure 3.5: 102 category database - Part 1

	Category	Nr of images		Category	Nr of images
	fire lily	40		king protea	49
	foxglove	162		lenten rose	67
	frangipani	166		lotus	137
	fritillary	91		love in the mist	46
	garden phlox	45		magnolia	63
	gaura	67		mallow	66
	gazania	78		marigold	67
	geranium	114		mexican aster	40
	giant white arum lily	56		mexican petunia	82
	globe thistle	45		monkshood	46
	globe-flower	41		moon orchid	40
	grape hyacinth	41		morning glory	107
	great masterwort	56		orange dahlia	67
	hard-leaved pocket orchid	60		osteospermum	61
	hibiscus	131		oxeye daisy	49
	hippeastrum	76		passion flower	251
	japanese anemone	55		pelargonium	71

Figure 3.6: 102 category database -Part 2

	Category	Nr of images		Category	Nr of images
	peruvian lily	82		stemless gentian	66
	petunia	258		sunflower	61
	pincushion flower	59		sweet pea	56
	pink primrose	40		sweet william	85
	pink-yellow dahlia	109		sword lily	130
	poinsettia	93		thorn apple	120
	primula	93		tiger lily	45
	prince of wales of feathers	40		toad lily	41
	purple coneflower	85		tree mallow	58
	red ginger	42		tree poppy	62
	rose	171		trumpet creeper	58
	ruby-lipped cattleya	75		wallflower	196
	siam tulip	41		water lily	194
	silverbush	52		watercress	184
	snapdragon	87		wild pansy	85
	spear thistle	48		windflower	54
	spring crocus	42		yellow iris	49

Figure 3.7: 102 category database - Part 3

Figure 3.8: Example images from different classes. Each row shows typical variations for one class.

3.3 Visualization of the 102 category dataset

We present two visualizations of the dataset; one according to colour and the other according to shape. To compute the visualization, we first extract features for each training image and cluster these using k-means clustering. Given a set of cluster centres, we then obtain a frequency histogram of word assignments, $x_f(i)$, for each image i and feature f. The χ^2 distance is computed between the frequency histograms for each of the training images. We then construct an intra-class distance matrix D_f by taking the minimum distance between the images from each pair of classes $c1$ and $c2$, so that

$$D_f(c1, c2) = \underset{i \in c1, j \in c2}{\operatorname{argmin}} \chi^2_f(x_f(i), x_f(j)) \tag{3.1}$$

Given the intra-class distance matrix, we compute a low-dimensional isomap embedding [104], which uses the local distances to learn the global geometry of the dataset. This is done by first determining which classes are neighbours by finding the K nearest neighbours for each class. A weighted graph is created with edges of weight $D_f(c1, c2)$ for each of the pair of classes $c1$ and $c2$ which are neighbours. The geodesic distances are then estimated by taking the shortest path of the weighted graph. Finally, multi-dimensional scaling is applied to find an embedding that preserves the shortest path distance.

Figure 3.9 shows the isomap using shape features (SIFT) and with $K = 2$. The figure shows clusters of bigger petals and tubular shaped flowers, spiky and ball shaped flower and flowers with many thin petals. Figure 3.10 shows the same isomap for the colour features(HSV) and also with $K = 2$. Note that the images displayed are randomly sampled, hence for flowers that occur in multiple colours it is not necessarily the closest colour which is plotted. Nevertheless, there are clusters of yellow, pink/red, white and blue/purple. The features are described in chapter 5.

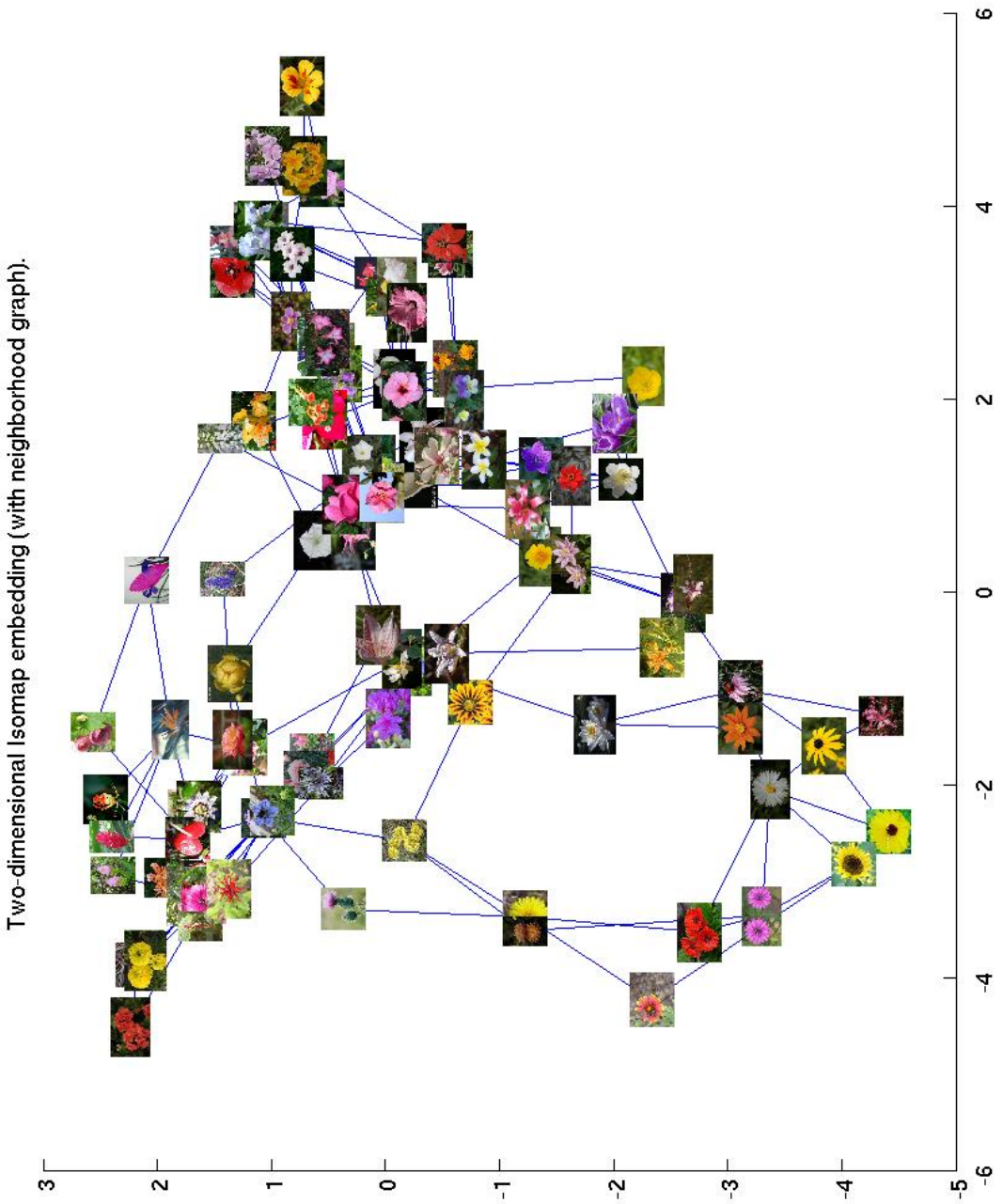


Figure 3.9: Isomap using shape features and K nearest neighbour, where $K = 2$. The images displayed are randomly chosen from the class. The figure shows clusters of bigger petals and tubular shaped flowers, spiky and ball shaped flower and flowers with many thin petals.

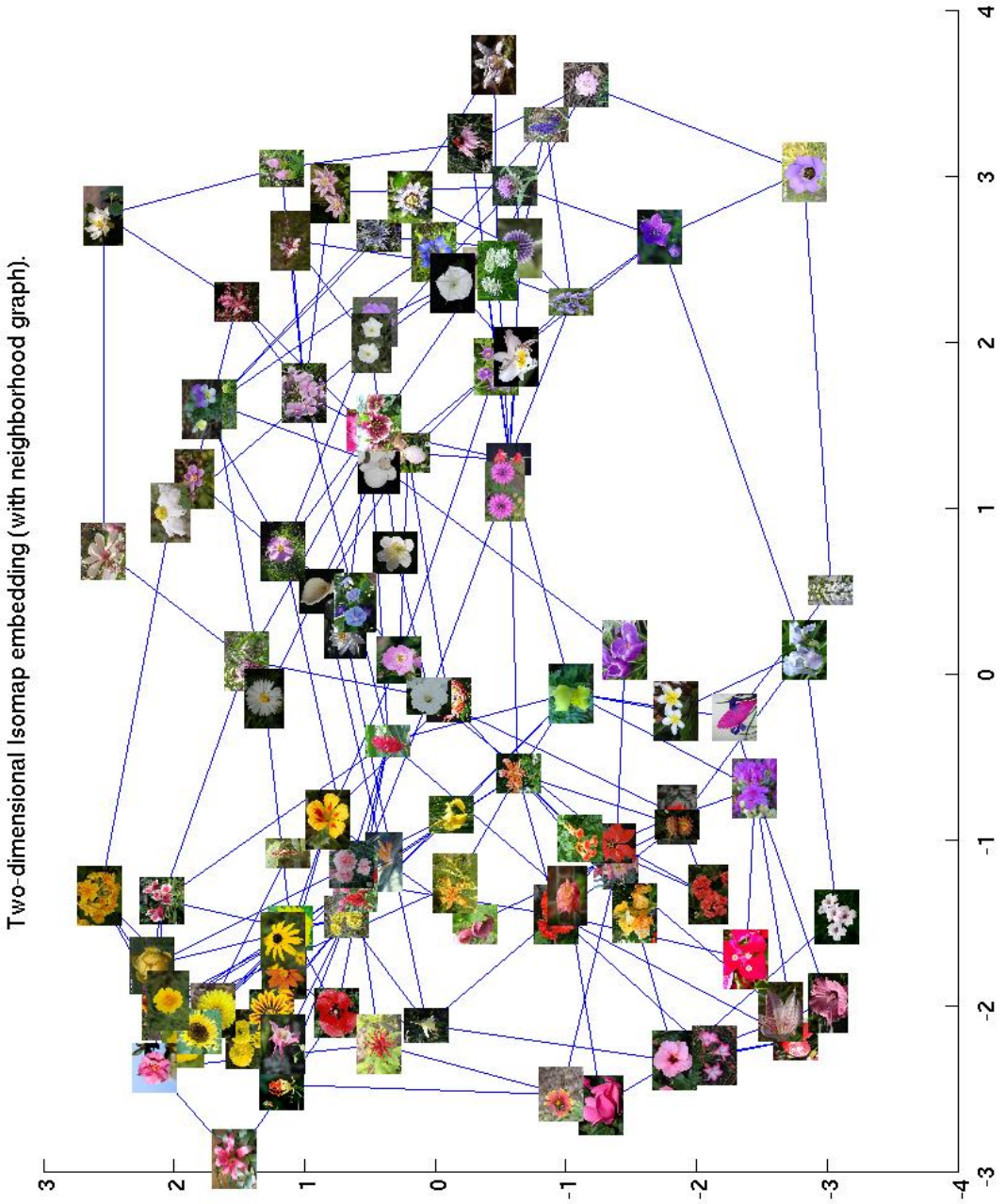


Figure 3.10: Isomap using colour features and K nearest neighbour, where $K = 2$. The images displayed are randomly chosen from the class. There are clusters of yellow, pink/red, white and blue/purple.

Chapter 4

Segmentation

In this chapter, we address the problem of automatically segmenting out the flower in a colour image given only that the image is known to contain a flower. The goal here is to obtain a foreground/background segmentation, where the foreground segmentation contains the flower and the background segmentation contains all the remaining parts of the image. Ultimately the purpose of the segmentation is to improve the flower classification (chapter 5). Flowers often live in similar environments and thus have similar background (see figure 4.1). We want to reduce the confusion caused by similar backgrounds and make sure that we are indeed classifying the flower. In addition, in order to improve segmentation we will fit a geometric model, which in itself will be used later to improve classification (chapter 6). Here we will present an iterative segmentation scheme and thoroughly evaluate how it improves segmentation and in the next chapter we will evaluate how it improves classification.

Kumar *et al.* [51] use pictorial structures with boundary shapes and texture features to localize the object and then use colour from the proposed region to

Figure 4.1: Examples of images with similar background.

obtain the segmentation. In their work they use a different model for most classes and poses. Here we assume that we have no information about the class or pose. Due to the sheer variety of flower classes, with varying shape, pose etc., we can not apply their method to our problem. Instead we present two variations on this theme: first, we reverse the order in which the features are used – we start with colour to propose a foreground/background segmentation and use this to initialize image specific shape measurements; second, we use a generic shape model which is applicable across a number of classes and viewpoints.

We first describe our segmentation algorithm, and introduce the flower shape model. Parameters are then optimized using a training set and performance is thoroughly evaluated on the ground truth trimap segmentations from a subset of the 17 category database, consisting of 13 flowers classes and more than 750 examples (section 3.1.1). The algorithm is compared to segmentation using a CRF cost function with general (non-class specific) foreground and background colour distributions optimized using graph cuts, and to the method of Saitoh *et al.* [91] for flower segmentation. We also present qualitative results on the 102 category database.

4.1 The Segmentation Algorithm

4.1.1 Overview

We first obtain an initial flower segmentation using *general* (non-class specific) foreground and background colour distributions. These distributions are learnt by labelling pixels in a few training images of each class in the dataset as foreground (i.e. part of the flower), or background (i.e. part of the greenery), and then averaging the distributions across all classes. Given these general foreground and background distributions, a binary segmentation is obtained using the contrast dependent CRF cost function of [13], optimized using graph cuts. This segmentation may not be perfect, but is often sufficient to extract at least part of the external boundary of the flower.

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. The image regions for the petals deemed to be consistent are used to obtain a new *image specific* foreground colour model. The foreground colour model is then updated by blending the image specific foreground model with the general foreground model. Similarly the *background* colour model is updated by blending an image specific model with the general background distribution. The CRF segmentation is repeated using the new colour models. In cases where the initial segmentation was not perfect, the use of the image specific foreground and background often harvests more of the flower. The steps of shape model fitting and image specific foreground and background learning can then be iterated until convergence.

The algorithm is illustrated in figure 4.2. We first describe these stages in more

detail (section 4.1.2-4.1.3) and then give implementation details (section 4.1.4). Variations on the schedule for combining the image specific and general distributions are discussed in section 4.3.

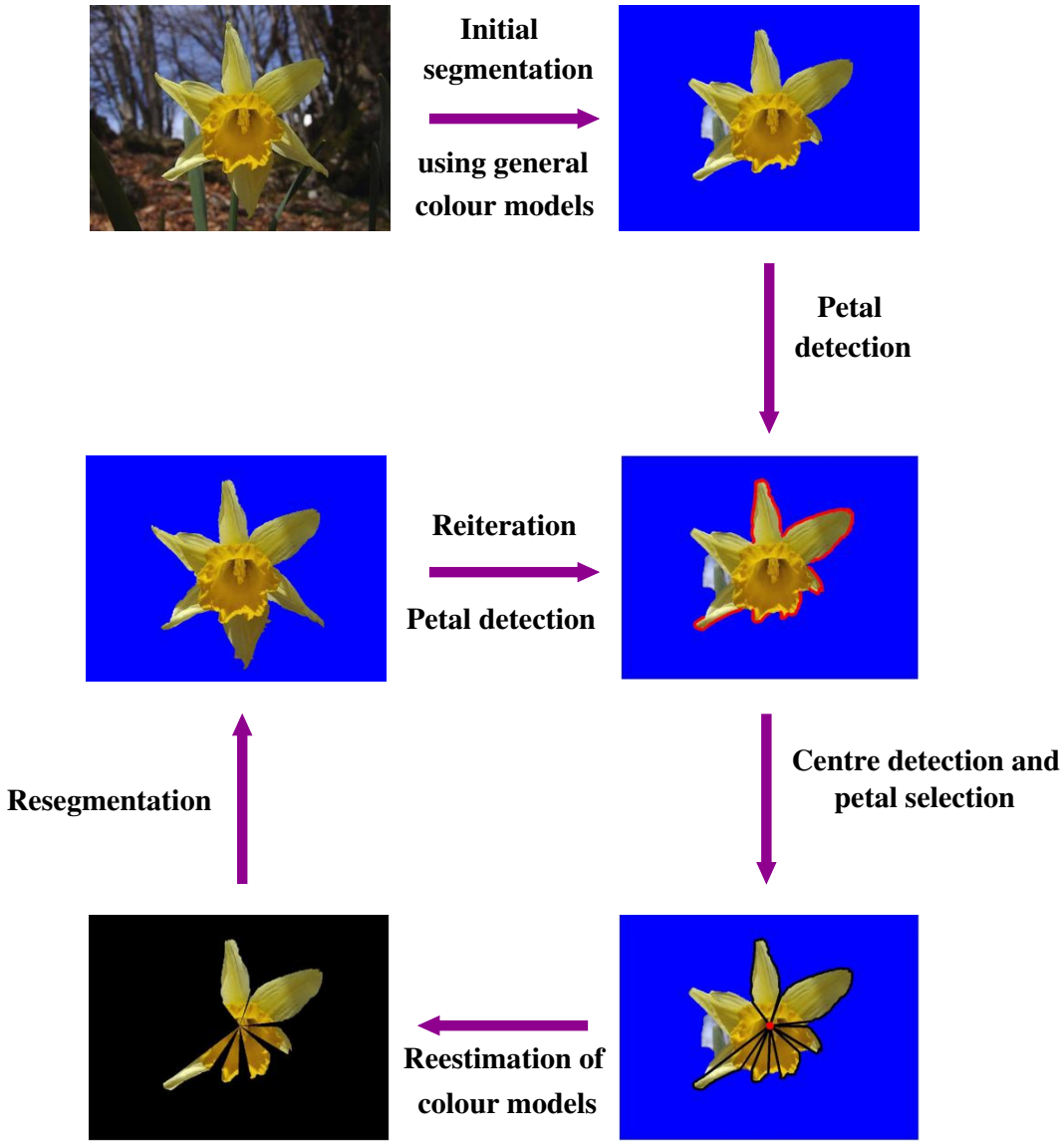


Figure 4.2: An overview of the segmentation algorithm.

4.1.2 Segmentation

The segmentation method is an implementation of the binary CRF with a contrast dependent prior proposed by [13]. Each pixel label, $\mathbf{x}_i$, has two possible states corresponding to foreground or background. The probability of each state depends on the likelihood that the pixel's colour corresponds to the foreground or background colour distribution. The cost function that is minimized (equation 4.1) has a unary term for each pixel (corresponding to the log-likelihood of its state) and a pairwise prior which adds a cost if there is a difference in the state between a pixel and its neighbour, *but* this cost is diminished if there is a strong gradient between the pixels (as measured in the images). The inclusion of data-dependent pairwise terms for pixels in a clique gives a substantial improvement in segmentation quality, and the resulting CRF can still be minimized using graph cuts as described in [13]. The relative trade-off between the unary term and the pairwise term is determined by the weight parameter γ. A large γ will align the segmentation with edges in the image and a small γ will make the segmentation rely more on the colour model. Given an image, $\mathbf{D}$, the cost function has the form:

$$E(x) = \sum_i \left(\phi(\mathbf{D}|\mathbf{x}_i) + \gamma \sum_j (\phi(\mathbf{D}|\mathbf{x}_i, \mathbf{x}_j) + \psi(\mathbf{x}_i, \mathbf{x}_j)) \right) + const \qquad (4.1)$$

where i is summed over all pixels, and j is summed over the neighbours of i. $\phi(\mathbf{D}|\mathbf{x}_i)$ is the colour log-likelihood, $\phi(\mathbf{D}|\mathbf{x}_i, \mathbf{x}_j)$ is the contrast dependant term and $\psi(\mathbf{x}_i, \mathbf{x}_j)$ is the smoothness dependent term.

4.1.3 Generic flower shape model

The flowering parts of a flower can be either petals, tepals or sepals. For simplicity we will refer to all three as petals.

The geometric representation of a petal and flower is illustrated in figure 4.3. In a frontal view of a "perfect" flower (think of a daffodil) the lines joining the extreme point to the mid-point of each petal coincide at the flower centre. However, under reasonable non-frontal viewpoints, and provided the deformation of the flower is not excessive, this construction is still a good approximation, and the agreement of each petal with a common flower centre can be used to verify or remove putative petals. The model represents a loose geometric configuration of petals. The particular shape of the petals and their number is not specified, nor is the symmetry of their arrangement. The only requirement is that the image regions deemed to be petals should agree on the flower "centre".

The construction of the extreme point is based on the canonical frame representation of Lamdan *et al.* [52]. Note, this construction (and indeed the entire flower shape model) is affine invariant. So, for example, it is unaffected if the image is rotated, scaled, or its aspect ratio changed. In particular this degree of invariance makes the construction tolerant to out of plane rotations – as the flower head turns from the camera. Figure 4.4 shows the model fitted to daffodils of different viewpoint, ranging from frontal viewpoint to side view. As the out of plane rotation increases some erroneous petals might be detected, e.g. several petals are detected on the tube of the daffodil. However together with the actual petals we are still able to detect a suitable "centre" of the flower and fit the model.

Since the model does not make strong requirements on the number or arrangement of petals, it is applicable to flowers with a small number of petals with ro-

tational symmetry (e.g. Daffodils, Windflowers), to flowers with very many petals with rotational symmetry (e.g. Sunflowers, Dandelions), and to flowers with a small number of petals without rotational symmetry (e.g. Iris, Pansies). Somewhat surprisingly, the same construction may be used for side views of flowers as illustrated by the Fritillaries and Tiger Lilies in figure 4.13. However, the generic model fitting is not expected to work well for many small flowers, e.g. Snapdragons, or flowers which are very assymetrical, e.g. Birds-of-Paradise.

4.1.4 Implementation details

Colour distribution representation and likelihood. The colour distribution is represented in RGB-space using a histogram with $N = 10^3$ bins, i.e. 10 bins per dimension. The colour log-likelihood, $\phi(\mathbf{D}|\mathbf{x}_i)$, of a pixel under the normalized histogram $hist(r, g, b)$ is then proportional to $-\log(hist(r, g, b))$. Empty bins are dealt with by adding $0.5/N$ to all bins (including non-empty ones) and then renormalizing the histogram. This avoids the problem of computing the log of empty bins, and is equivalent to a uniform prior over the colour space [67]. Following [13], the contrast dependant term is defined by

$$\phi(\mathbf{D}|\mathbf{x}_i, \mathbf{x}_j) \propto \exp\left(-\frac{(I(x_i) - I(x_j))^2)}{2\sigma^2}\right) \cdot \frac{1}{dist(x_i, x_j)}, \tag{4.2}$$

where $I(x_i)$ and $I(x_j)$ denote the weighted linear combination of the RGB values of pixels x_i and x_j respectively. The smoothness term $\psi(\mathbf{x}_i, \mathbf{x}_j))$ is set to a constant, i.e. it adds a constant penalty to neighbouring pixels being assigned different labels.

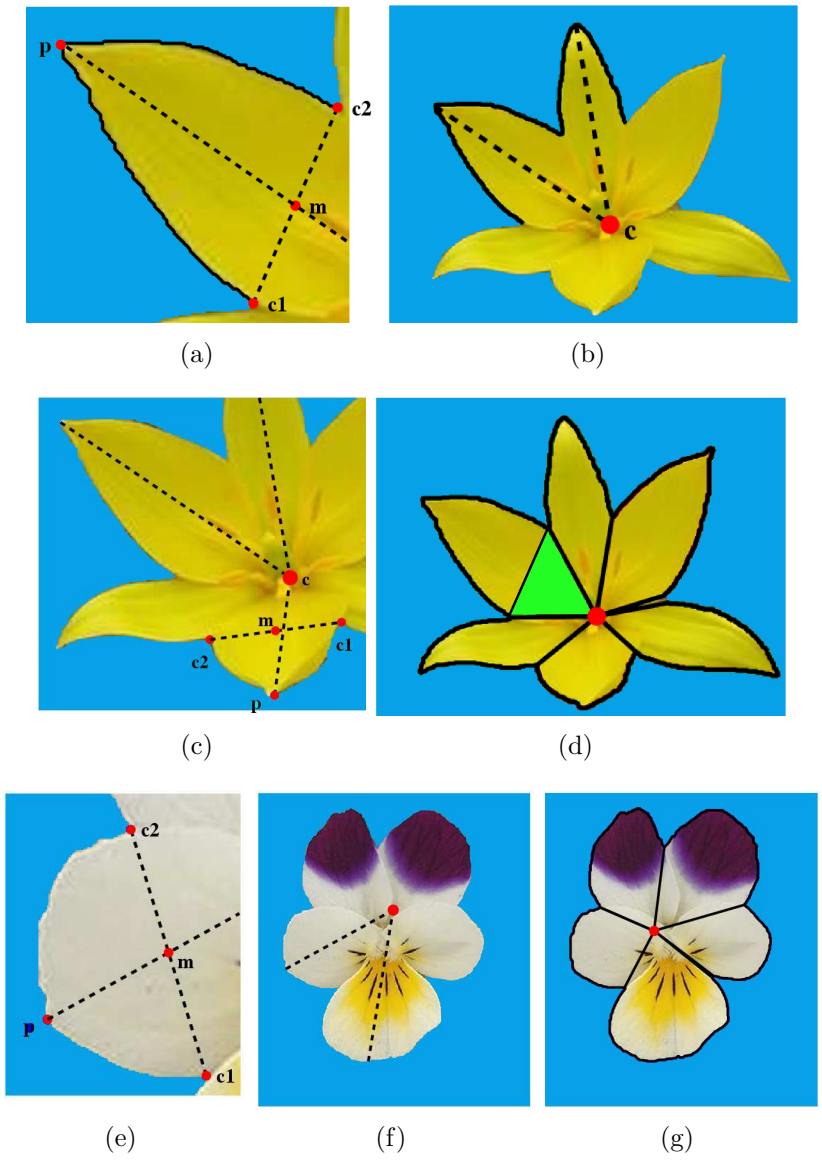


Figure 4.3: **The geometric flower model.** – a) A petal is defined by the boundary curve between the corners $\mathbf{c}_1$ and $\mathbf{c}_2$. The extreme point $\mathbf{p}$ is the point on the curve furthest from the line $\langle \mathbf{c}_1, \mathbf{c}_2 \rangle$, and with tangent parallel to the line. b) A hypothesis for a centre obtained by intersecting two petal mid lines $\langle \mathbf{p}, \mathbf{m} \rangle$. c) The consistency of a putative centre $\mathbf{c}$ is measured by the deviation of the line $\langle \mathbf{c}, \mathbf{p} \rangle$ from the mid-point $\mathbf{m}$, and d) the petal regions consistent with the centre with greatest consistency. Note that the wedge region connecting the petal with the centre (marked in green) also is included in the petal region. Bottom row: A second example of fitting the model, here for a pansy. e)Single petal model, f) putative centre and g)final centre and verified petals consistent with the centre.

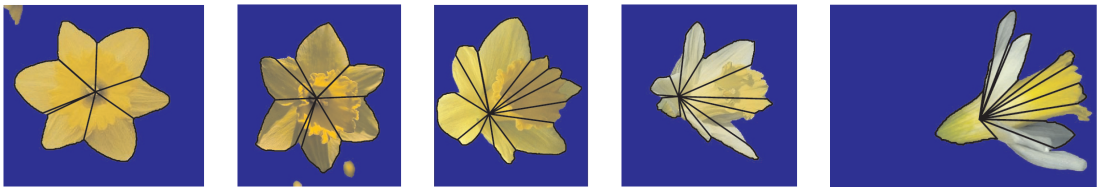


Figure 4.4: Geometric flower model fitted to daffodils with varying viewpoint. Note that the flexibility of the model allows it to be fitted to frontal views as well as side views.

Petal detection. The boundary of the segmented foreground region is traced by examining an 8-neighbourhood around each foreground pixel. Once all the boundary pixels are found, corners are detected using a "worm" that slides around the boundary and declares a corner at a boundary point where the distance to the straight line between its head and tail exceeds a threshold, and is a local maximum. Convex corners (wrt to the background region) are the potential petal corners.

The top row in figure 4.3 shows how petals are detected. Given two neighbouring corner points, $\mathbf{c}_1$ and $\mathbf{c}_2$, the line $\langle \mathbf{c}_1, \mathbf{c}_2 \rangle$ is determined. Next we search for a point, $\mathbf{p}$, on the boundary between $\mathbf{c}_1$ and $\mathbf{c}_2$, whose tangent has the same orientation as the line $\langle \mathbf{c}_1, \mathbf{c}_2 \rangle$. If no points are found or if the ratio between the distance along the boundary and the length of $\langle \mathbf{c}_1, \mathbf{c}_2 \rangle$ is smaller than a threshold, t, the petal between the two corners is removed. The line $\langle \mathbf{p}, \mathbf{m} \rangle$ joining $\mathbf{p}$, and the middle point $\mathbf{m}$ of the segment $\langle \mathbf{c}_1, \mathbf{c}_2 \rangle$ indicates where the centre of the flower might lie. Figure 4.5 shows the fitted generic flower shape model for several flower classes.

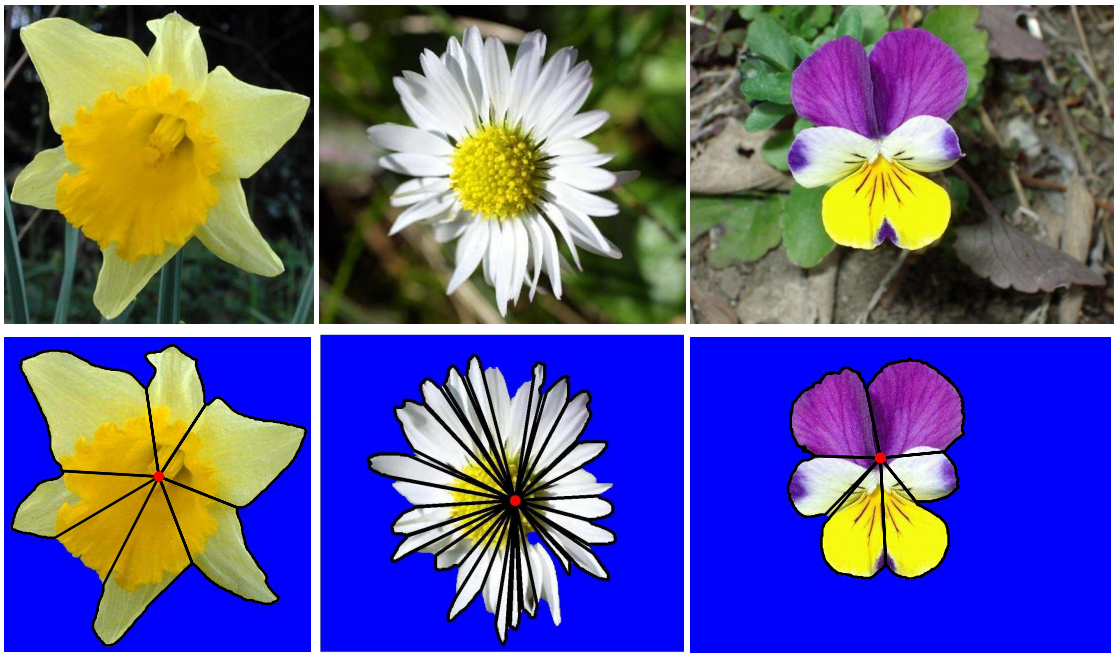


Figure 4.5: Top row: typical flower photographs, note the variety of imaging conditions. The classes are (left to right) daffodil, daisy and pansy. Bottom row: the flower shape model fitted automatically (note the lines to the flower "centre") and the resulting segmentation. The same shape model is used in all cases despite differing numbers of whorls, the large variation in the number of petals (daffodil & pansy vs daisy), and variation in the shape of the petals.

Centre voting. For each pair of petals we find the intersection between the corresponding lines $\langle \mathbf{p}, \mathbf{m} \rangle$. This point of intersection, $\mathbf{c}$, is a putative centre. Additional petals are added with a cost corresponding to deviation from $\mathbf{m}$ of the intersection, $\mathbf{i}$, of the lines $\langle \mathbf{c}_1, \mathbf{c}_2 \rangle$ and $\langle \mathbf{p}, \mathbf{c} \rangle$ (the cost is measured on each petal). The deviation is normalized by the half-length of the segment $\langle \mathbf{c}_1, \mathbf{c}_2 \rangle$. If the deviation is greater than unity, i.e. the intersection $\mathbf{i}$ lies outside the segment, then the petal is not added to the putative centre. The centre with highest consistency, in terms of number of petals and lowest cost, is chosen to be the centre of the flower.

Image specific foreground. Given the centre, **c**, of the flower, the image specific foreground model is obtained by computing the colour histogram for the petals, including the triangular wedge of the centre region consistent with this given centre (figure 4.3).

Image specific background. The image specific background model is obtained by computing the colour histogram for the image region segmented as background. For blending, this histogram is first thresholded so that only dominant colours in the background remain.

Segmentation termination. The detection of petals and refinement of the foreground and background models is repeated iteratively. Once there is no change in the segmented region between two consecutive iterations the algorithm has converged. If the algorithm does not converge, we revert to the initial segmentation. In this way we can deal with categories where the geometric model fitting does not work, by reverting to the graphcut segmentation.

4.2 Dataset and experimental evaluation

4.2.1 Dataset

For the experiments we use the ground truth labelled subset of the 17 category dataset (section 3.1.1). This consists of 13 flower classes with a total of 753 images. The dataset is split into a training set and a test set. The training set consists of 20 images per category (260 images) and is used to learn the general foreground and background colour models, and other parameters. The test set consists of the remaining 493 images, with a minimum of 20 images per category.

4.2.2 Performance measures

The segmentation accuracy is measured by computing an overlap score, P, between the ground truth segmentation and the segmentation obtained by the algorithm:

$$P = \frac{\text{true foreground} \cap \text{segmented foreground}}{\text{true foreground} \cup \text{segmented foreground}} \tag{4.3}$$

In evaluating this score only the labelled pixels are considered (i.e. the unlabelled pixels have no effect). The score is 1 for a perfect segmentation, and less than 1 otherwise.

To give an idea of the type of error that occurs for various overlap errors, figure 4.6 shows typical segmentation deficiencies at overlaps in the range 0.92 – 1.0 for two different images.

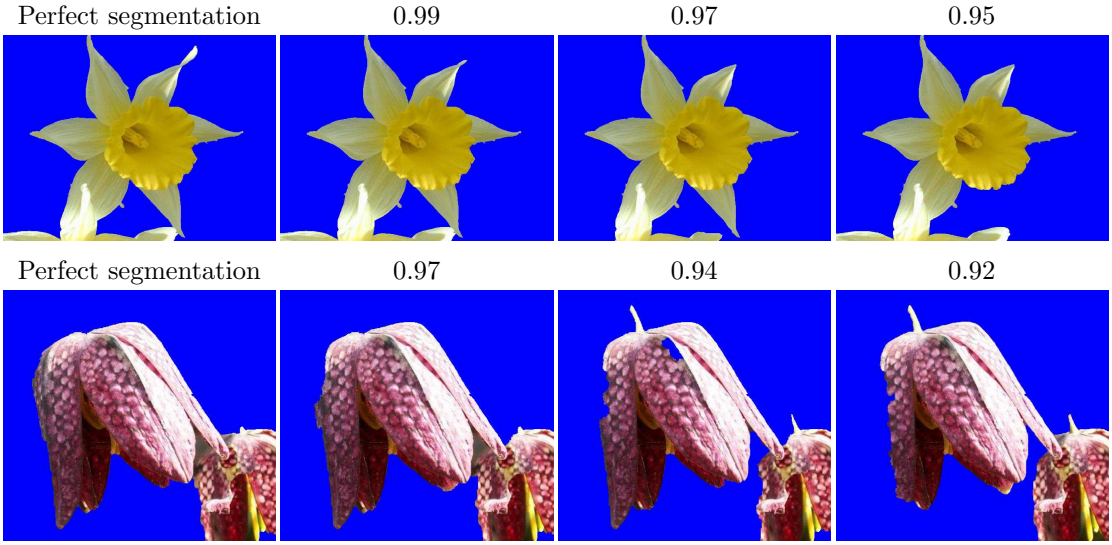


Figure 4.6: Segmentation deficiencies for different levels of overlap. The overlap score is given above the images. Top row: A bit of the top right petal disappears at 0.99, and at 0.95 an entire petal is lost. Bottom row: At 0.97 a bit of the left edge is lost, at 0.94 a bit of the stalk is added and a hole appears, and at 0.92 part of the left petal is missing.

4.3 Optimization on the training set

The algorithm has several parameters that must be set (such as thresholds for the petal detection) and are then fixed throughout the iterations. Conversely there are also model parameters that can be varied throughout the iterations, such as how to combine the image specific and general colour distributions. We use the training set with provided ground truth segmentations to optimize parameters and investigate a number of *schedules*. The cost that is maximized is the overlap score summed over all the training images. We investigate three schedules (foreground, background, CRF γ), and show that in each case the freedom of choosing the schedule can be used to overcome specific failure modes of the algorithm (without introducing additional but different errors). As will be seen we succeed in obtaining a method that is applicable to many flower classes, colours, shapes, and backgrounds (figure 4.13).

Petal detection parameters. The parameters here are (i) the worm length, and (ii) the minimum distance required from a potential corner and the straight line between the worm's head and tail. If the worm length is too short, many corners will be detected in small indentations of the petal; if the worm length is too long many corners will be missed. The second parameter determines how acute a corner must be to be detected. A large distance means that the corners have to be very acute. If the threshold on the distance is too small a more or less straight line segment could be considered a corner, and if too large corners will be missed and in particular the petal detection will generalize poorly to flowers with less acute corners. In the case of (i), for example, the performance is inferior and sensitive to worm length in the range of 10–20 pixels, but for a length of 20–50 pixels the performance is much the same, and optimal. The worm length is constant across all images.

Background schedule. Here we investigate the schedule of combining the image specific background with the general background distribution as the iterations proceed. This schedule can be used to overcome the problem of *background leaking*: sometimes the segmented foreground region will include a little bit of the background around the petals. Consequently the image specific foreground colour model will be polluted by some image specific background colours – colour X, say. If only the general background model is used in the next graph cuts iteration, and X has a relatively low likelihood of being background under this model, then background regions of colour X will leak into the foreground in the next iteration (figure 4.7). By updating the background colour model for the image, we can increase the likelihood of background regions with colour X in the specific image and thus overcome this problem. This requires some care: in some cases large foreground regions, like a petal, are marked as background. These large foreground regions are likely to be included in the thresholded image specific background distribution. This inhibits the flower region from growing and recovering this petal. To prevent this (i) regions in the background are only included in the background distribution if they are outside a certain distance from the petals in the foreground region, and (ii) only large peaks (above the threshold th) in the image specific background distribution are blended with the general background distribution. (ii) works under the assumption that the background consists of few frequently occurring colours. If there are no large peaks the background distrubition will not be updated.

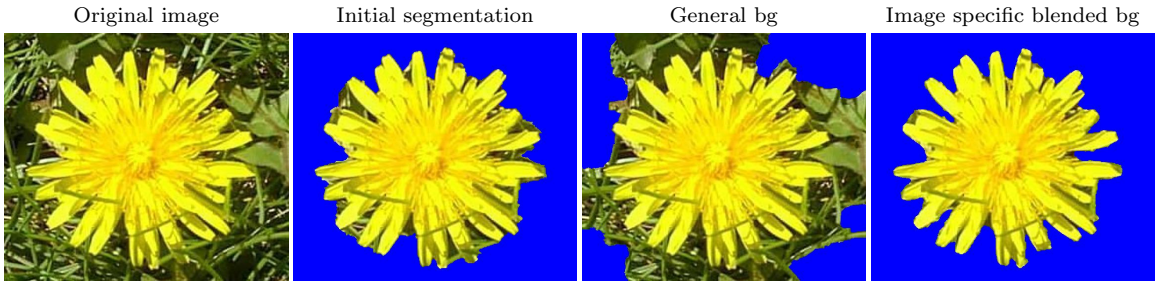


Figure 4.7: **Background leaking**. Some of the green background has been segmented as foreground. If no image specific background information is included this region will grow as foreground in the next iteration. Including image specific background information prevents this from happening.

Foreground schedule. If an incorrect detection of petals occur, a scheme which simply learns an image specific foreground model from the detected petals is unlikely to recover. In figure 4.8, for example, the petal detection in a previous iteration has resulted in a perfect foreground/background segmentation. However, in this iteration the spatial model has not detected one of the petals, thus the colour of this petal will not be included in the foreground distribution. This results in the petal being removed from the segmentation (column 3) in the next iteration. The scheme should be able to recover from an incorrect detection of petals in one single iteration. By including some hysteresis we can limit the magnitude of the changes that occur between consecutive iterations (column 4).

The foreground distribution, h_t, starts as the general foreground distribution, $h_{general}$. In consecutive iterations it is updated by computing the histogram for the colours from the petals detected in the previous iteration and the petals detected in the current iteration:

$$h_t = \left\{ \begin{array}{rl} h_{general} & \text{for} \quad t = 0 \\ (h(petals_t) + h_{t-1})/2 & \text{for} \quad t > 1 \end{array} \right\} \tag{4.4}$$

Figure 4.8: **Foreground schedule**. After model fitting, one petal (the lower one) is missed by the spatial model, and consequently its colours will not be included in the image specific foreground model if this is computed only from the detected petals in this iteration. Instead, this can be prevented by including colour information from petals computed in the previous iteration.

CRF γ schedule. If the initial segmentation based on the general foreground and background colour models is particularly poor, then the scheme is unlikely to recover (figure 4.9). We can guard against this to some extent by initially increasing the edge weight in the CRF cost function (4.1) to encourage segmentation at flower boundary edges (as well as other internal and background edges). Once some petals have been detected, more weight can be put on the likelihood terms as the colour models will have been updated with the image specific distributions. This is achieved by initially using a large weighting factor γ between the pairwise and unary terms in the CRF (4.1) and subsequently reducing γ. The actual values of γ used are optimized on the training set.

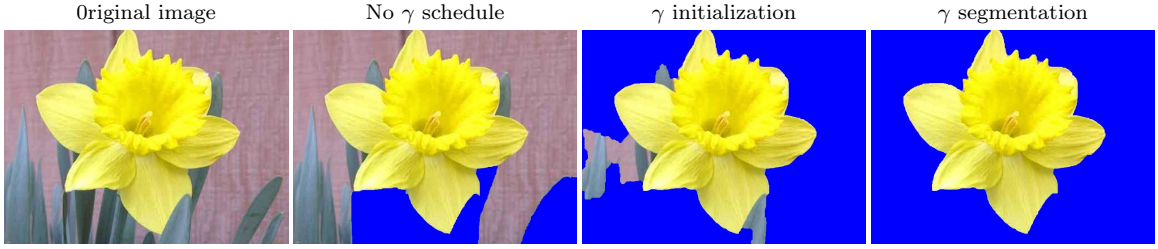


Figure 4.9: γ **schedule**. From left to right: The original image, segmentation using the same γ for all stages of the iterative scheme, initial segmentation using a high γ (encouraging segmentation at the edges in the image) and the final segmentation using a lower γ (relying more on the colour likelihood).

Centre clamping. The centre of some flowers, e.g. Sunflowers, will never be segmented as foreground because the dark brown colour is much more likely to be background (figure 4.10). Since the scheme detects the centre of the flower, we have an idea of where the centre is and can choose to clamp a region around this centre as foreground. Thus we are forcing the flower centre region to be foreground. Again this requires some care: if we are looking at a side ways view of a flower the centre can fall outside the flower region. In order not to include centres like these, we only clamp centres inside the convex hull of the flower region. The size of the region included around the centre is optimized on the training set.

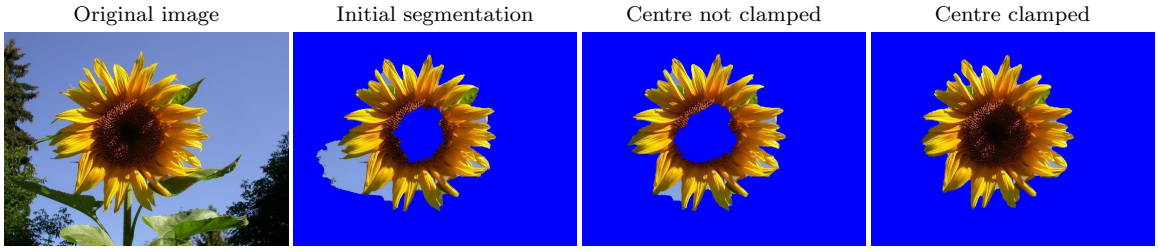


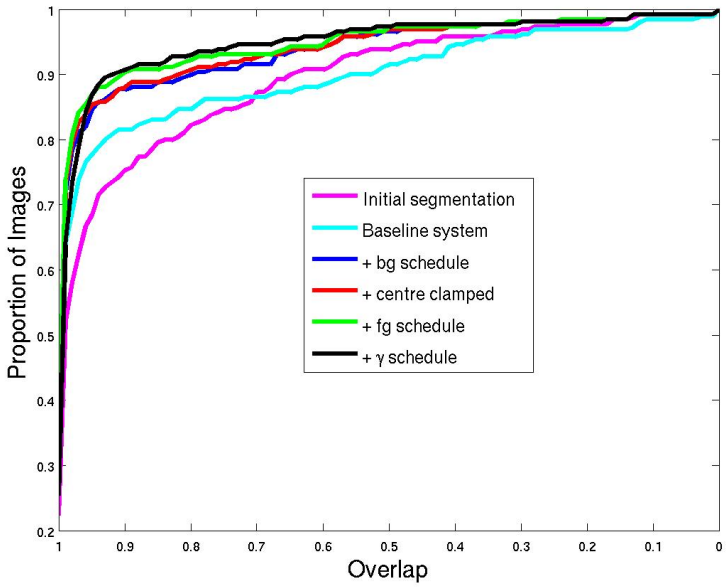
Figure 4.10: **Centre clamping**. If the colour of the centre of the flower has a low likelihood of being foreground it will not be segmented as foreground. This problem can be avoided by clamping the centre of the flower as foreground. Note, the centre of the flower is clamped (not the centre of the image).

4.3.1 Performance on the training set

Figure 4.11 shows how these schedules improve the results on the training set. We plot the proportion of images that have an overlap score greater than P, against P. This graph is chosen to resemble an ROC curve, in that perfect performance corresponds to a curve approaching the top left corner.

We start off with a baseline system which uses the general background distribution (no update), only the image specific foreground (i.e. no blending with the

general foreground) after the initial segmentation, no centre clamping, and a fixed value for γ. To the baseline system we first incorporate the background schedule, secondly clamp the centre region, thirdly the foreground schedule is added, and finally the γ schedule. By making all these changes to the baseline system the average overlap score increases from 89.7% to 95.0%.



Average Overlap Score	
initial segmentation	89.7%
baseline system	90.2%
+ bg schedule	94.0%
+ centre clamped	94.3%
+ fg schedule	94.9%
+ γ schedule	95.0%

Figure 4.11: Comparison of segmentation performance on the training image set. Left: The x axis shows the overlap score P, and the y axis the proportion of images that have an overlap score greater than or equal to P. The magenta line shows the results of the initial segmentation using the general foreground and background colour models, the cyan line the results of the baseline system, the blue line the results when the background schedule is added, the red line when the centre is clamped, the green line when the foreground is blended with previous iterations and the black line when γ is changed between iterations. Right: Average overlap score for the different schemes.

A scatter plot of the overlap score per class is shown in figure 4.12. The figure shows the overlap score for the complete iterative scheme plotted against the overlap score for the initial segmentation. Some improvement is obtained for most classes. The largest improvement from 72.8% to 94.6% is for the Sunflower class, where

the initial segmentation often misses the centre and includes part of the sky as foreground. Two classes, Tiger Lilies and Wild Tulips, show a slight deterioration of the overlap score, 1.1% and 2.6% respectively. In the case of the Tiger Lilies the decline is due to minor changes in the segmentation for several images. In the case of the Wild Tulips the segmentation fails for an image with a frame around the picture. Introducing the γ schedule makes the initial segmentation align with strong edges, i.e. the frame.

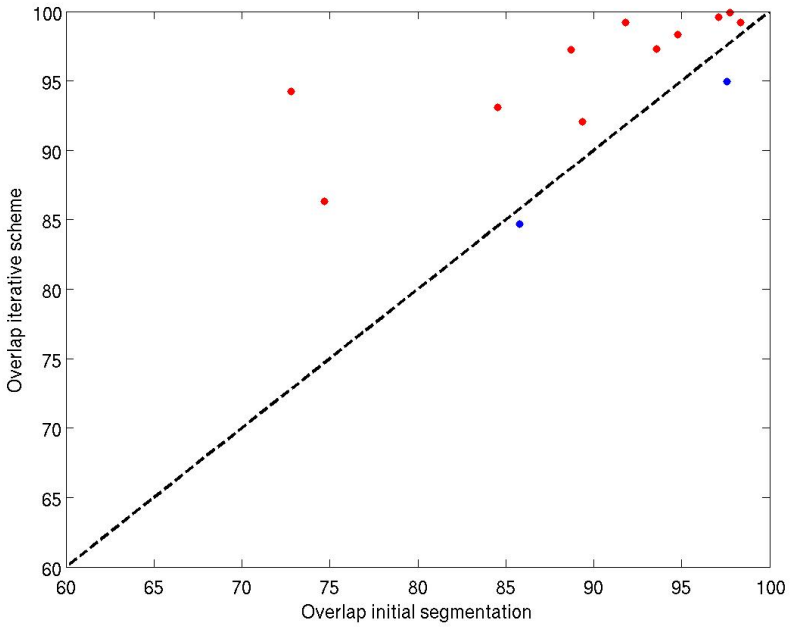


Figure 4.12: Scatter plot of overlap scores on the training set for each class. Red dots indicate classes where the iterative segmenation scheme improves the segmentation and blue dots where it deteriorates the segmentation.

4.4 Flower segmentation on the test set

The results in this section are presented on the test set consisting of 493 images, with a minimum of 20 images per category. Figure 4.13 shows examples of the coupled algorithm improving the segmentation for different classes. It can be seen that it both manages to retrieve missing flower regions, and remove background regions that initially were misclassified. Typically only a few iterations are required – in 68% of the cases there is very little change after the first iteration. After five iterations 97% of the segmentations have converged.

In figure 4.15 we compare performance using only the general colour distributions without using the shape model, to the performance using the coupled segmentation algorithm (iterating colour and shape fitting). It can be seen that learning an image-specific foreground distribution significantly improves the segmentation: for example the initial segmentation has an overlap of 95% for 62% of the images and this is increased to 83% of the images.

A scatter plot of the overlap score per class is shown in figure 4.16. The figure shows the overlap score for the iterative scheme plotted against the overlap score for the initial segmentation. Some improvement is obtained for most classes. As for the training set the largest improvement (16.5%) is for the Sunflower class. However, in contrast to the training set all of the classes achieve a higher overlap score for the iterative scheme.

Figure 4.17 shows example segmentations obtained using our segmentation scheme on the 102 category dataset. It can be seen that it also works well for flowers very different in shape to those the method was developed on. Figures 4.18-4.19 show one example segmentation for each class in 102 category database. The captions highlight some mistakes made. The segmentation fails when the background or part

of the background have similar colour to the flower (see for example the Bishop of Llandalf, the Gaura and the Grape Hyacinth). If there are multiple flowers with different colour only one of the flower will be segmented (see for example the Carnation).

Figure 4.13: Example results for a number of classes on the test set. In each case iteratively updating the foreground colour model leads to a full segmentation, despite only a partial segmentation at the start. There are examples where missed foreground is retrieved (crocus, iris) and where erroneous background is removed (tiger lily, wild tulip).

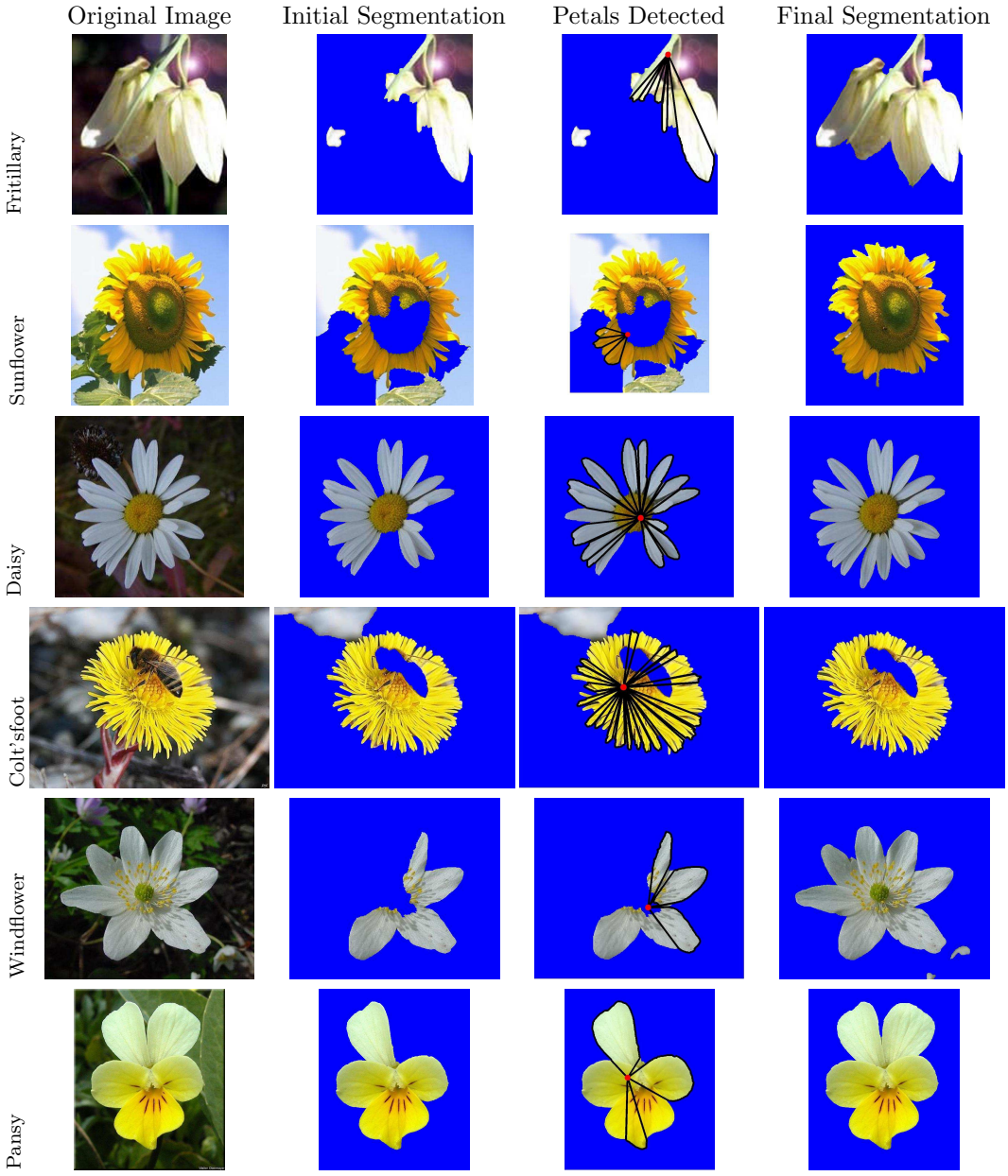
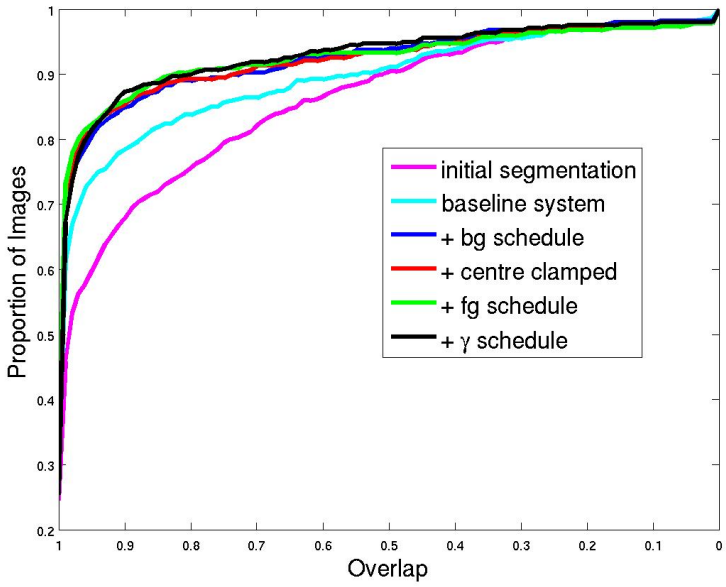


Figure 4.14: Example results for a number of classes on the test set. In each case iteratively updating the foreground colour model leads to a full segmentation, despite only a partial segmentation at the start. There are examples where missed foreground is retrieved (crocus, iris) and where erroneous background is removed (tiger lily, wild tulip).



Average Overlap Score	
initial segmentation	86.2%
baseline system	89.5%
+ bg schedule	92.4%
+ centre clamped	92.5%
+ fg schedule	92.7%
+ γ schedule	93.0%

Figure 4.15: Comparison of segmentation performance on the test set.

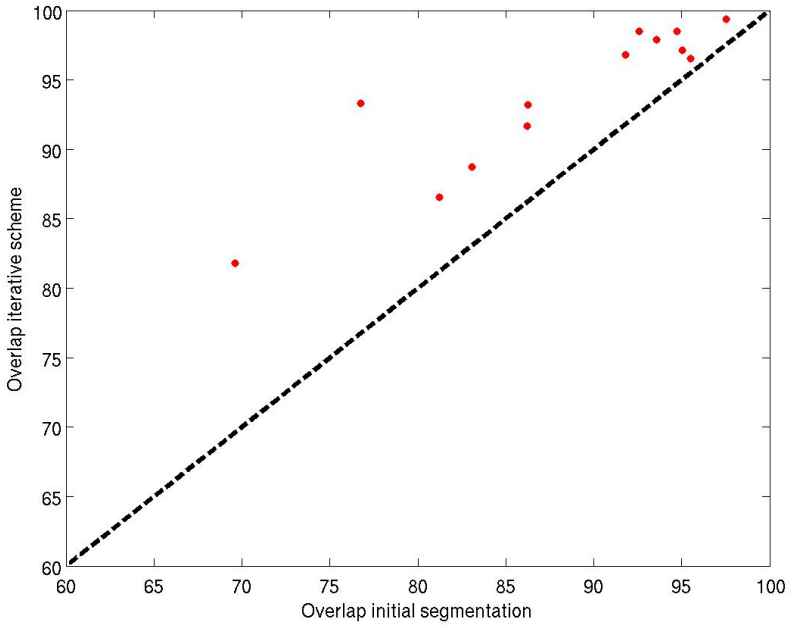


Figure 4.16: Scatter plot of overlap scores on the test set for each class. Red dots indicate classes where the iterative segmenation scheme improves the segmentation and blue dots where it deteriorates the segmentation. Note that all dots are red.

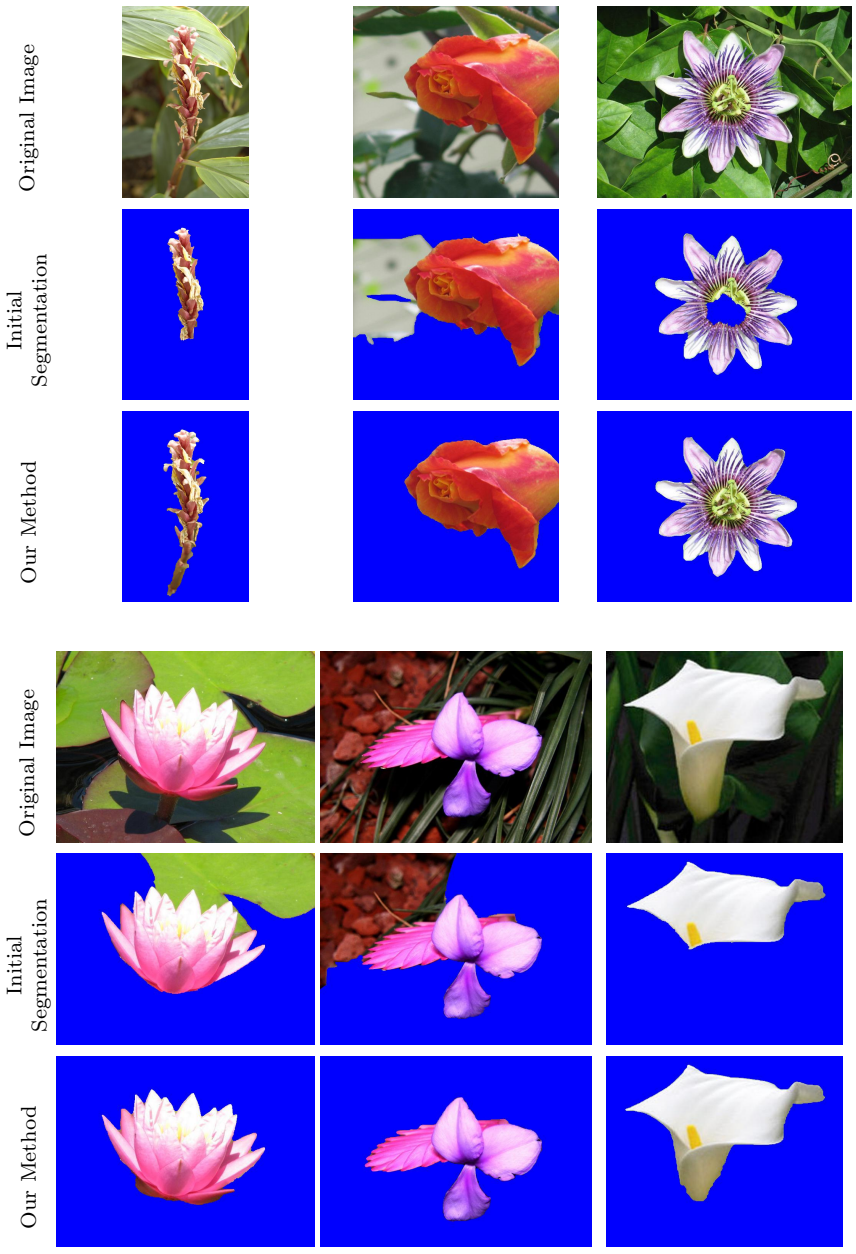


Figure 4.17: Segmentation of the 102 category dataset. Note that we are able to both find missing parts of the flower and remove bits that have been incorrectly segmented as foregound.

Figure 4.18: Example segmentations 102 dataset– Part 1. The figure shows one example segmentation for each class. Note that our scheme can deal with many different classes. Mistakes: The stalk of the Blackberry lily is included in segmentation; the orange Barbeton Daisy's have not been segmented from the orange background; middle is missing for the Bolero Deep Blue; only one of the Carnations has been segmented.

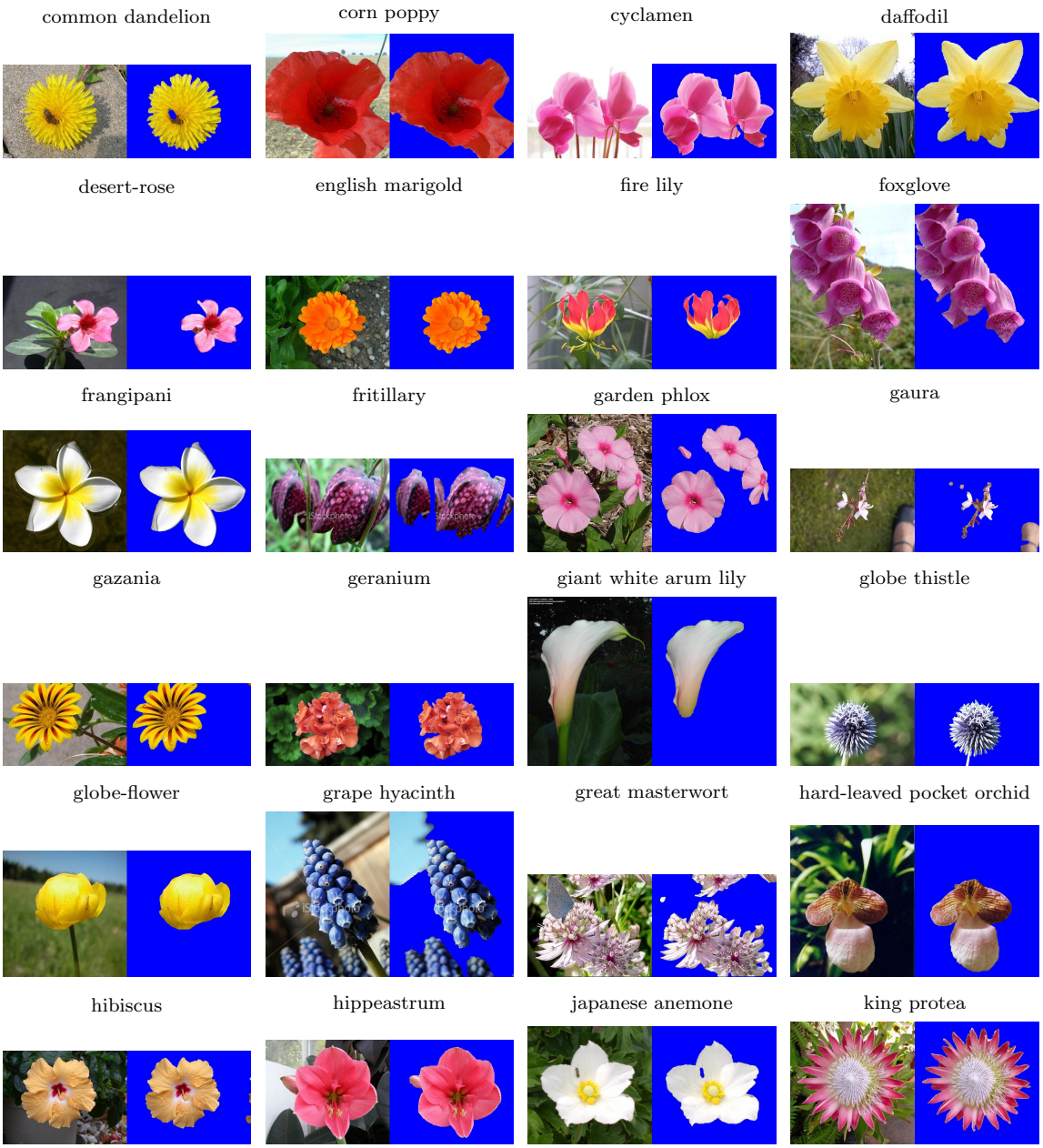


Figure 4.19: Example segmentations 102 dataset– Part 2. The figure shows one example segmentation for each class. Note that our scheme can deal with many different classes. Mistakes: the Gaura's colour is too similar to the foot so both have been segmented as foreground; the sky in the background of the Grape Hyacinth has not been separated from the blue of the flower.

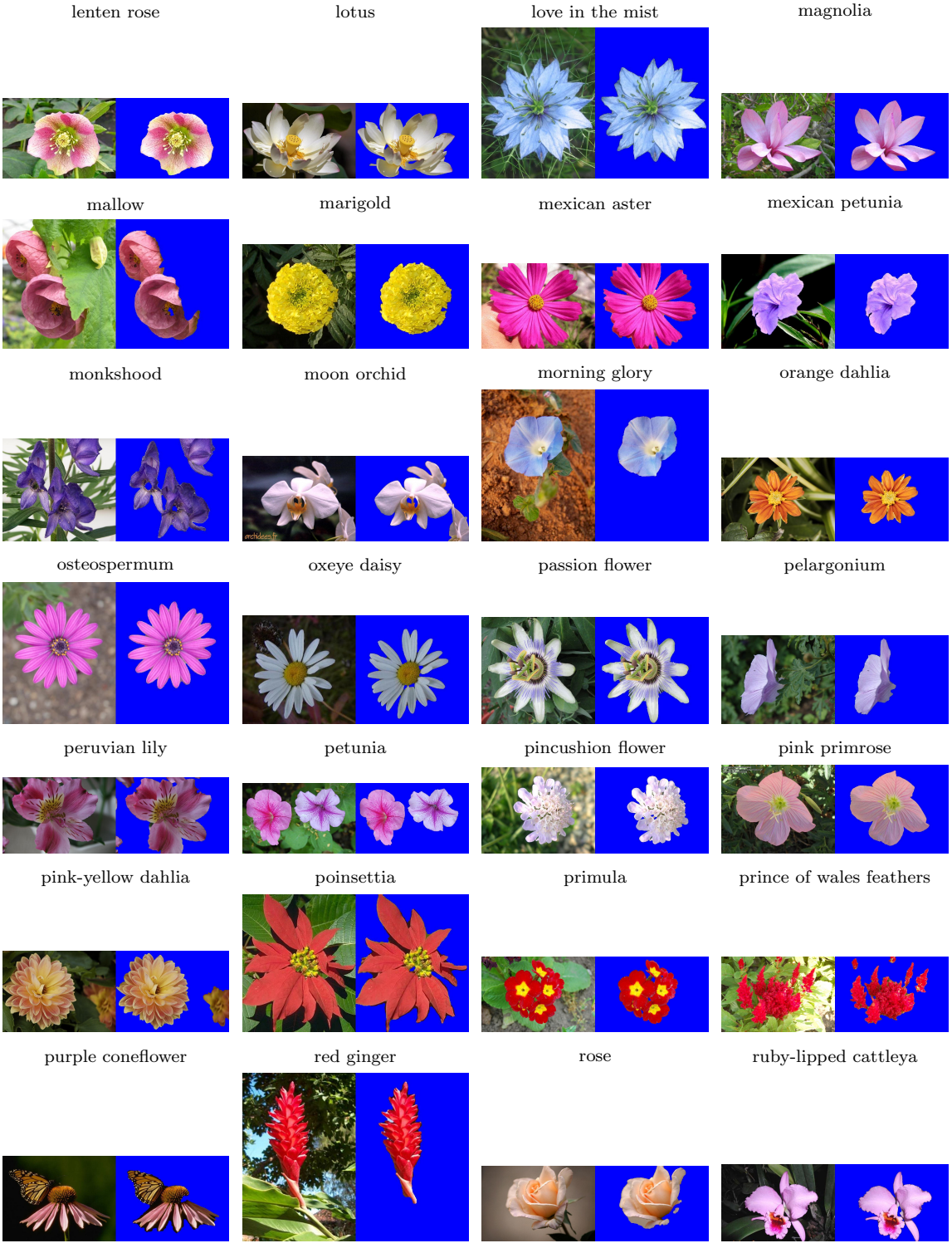


Figure 4.20: Example segmentations 102 dataset– Part 3. The figure shows one example segmentation for each class. Note that our scheme can deal with many different classes.

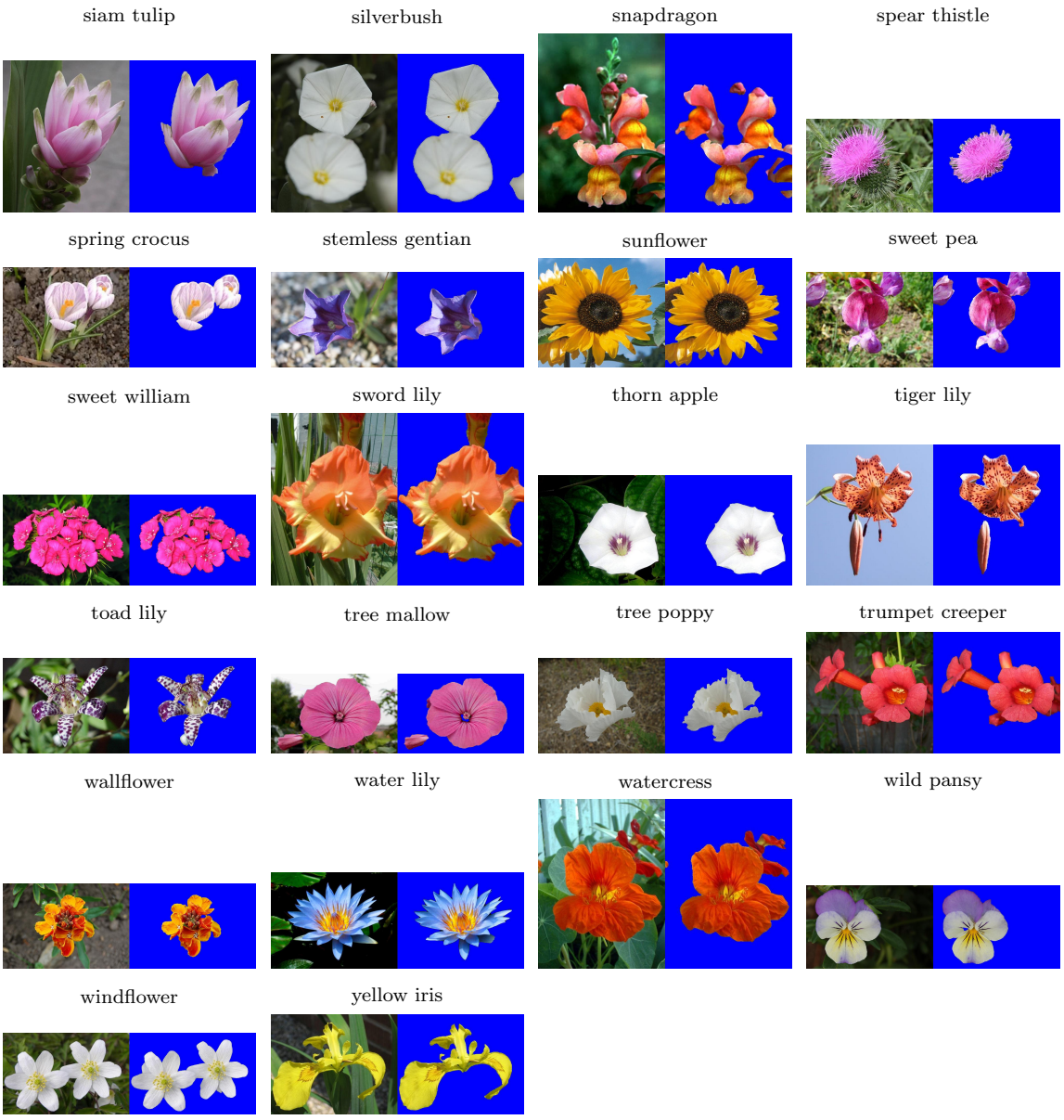


Figure 4.21: Example segmentations 102 dataset– Part 4. The figure shows one example segmentation for each class. Note that our scheme can deal with many different classes.

4.4.1 Comparison with other methods

We compare our method to the method of Saitoh *et al.* [91]. They start with no prior colour information. The centre region is clamped and this centre region is expanded using "Intelligent Scissors" [80] to cut out the flower region. We compare our method to theirs with the modification that graphcuts are used to cut out the flower region, i.e the centre region of the image is taken as foreground distribution and the periphery of the image is taken as background distribution. The average overlap score for their method is 82% compared to our score of 95%. Their method is a real alternative baseline method and many flower images are in fact centred in the image, so as expected it works well for many images. The main reason for their failures is that some flowers are not centred in the image. It will also fail for cases where the flower centre has a different colour to the periphery of the flower. Figure 4.22 shows some images that fail using Saitoh's method.

4.5 Conclusion

We have demonstrated that flower segmentations can be significantly boosted by using an image specific colour distribution, and that this distribution can be learnt automatically by fitting a general flower shape model. We have shown that our scheme is able to deal with many categories. We can deal with categories to which the model does not fit, by automatically detecting when a segmentation is bad and reverting to the original graphcut segmentation for those images. We have concentrated here on segmenting out the flower. A similar method (general colour model, generic shape model) could now be used for segmenting out the leaves and stalk – making strong use of green in the colour model. The current model is not

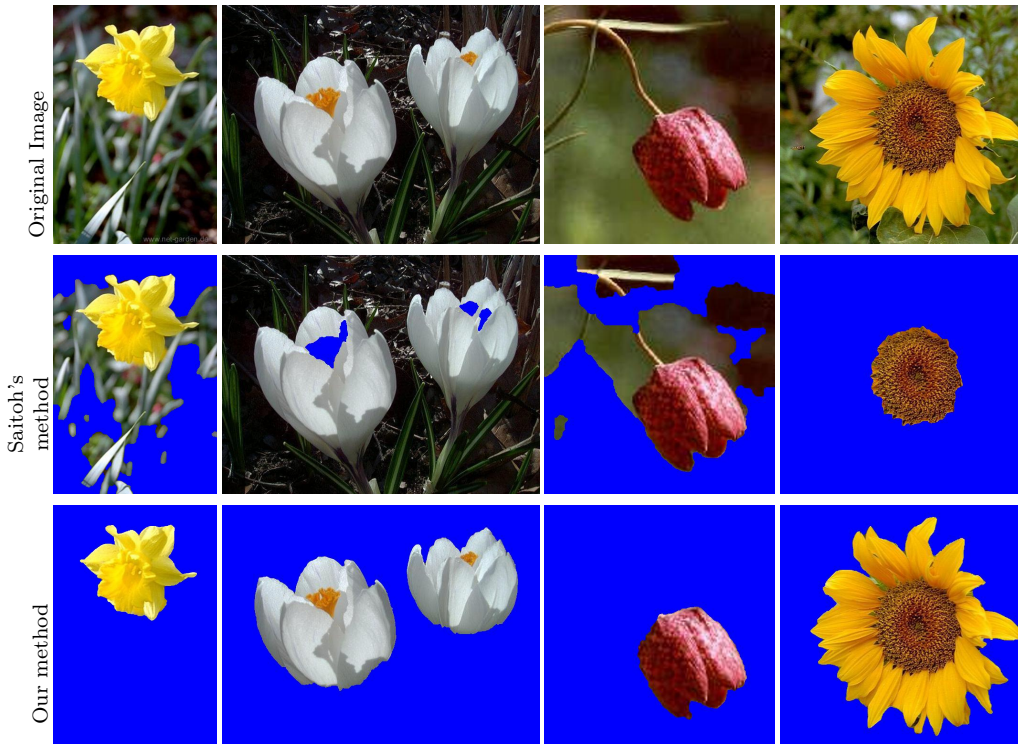


Figure 4.22: Comparison with Saitoh *et al.* [91]. Top row: Original images, Middle row: Segmentations using our implementation of Saitoh's method, Bottom row: Segmentations using our method.

suitable for fields of flowers. For such cases a model that groups repeated shapes would be more appropriate, such as [109].

The method of Das *et al.* [27] is complementary to ours, in that it does not use a spatial model at all. In their implementation the authors always remove green, brown, gray and black from the pool of foreground colours, which means it would fail to segment the centre of the Sunflower, the spots on the Tiger Lily, the stripes on the Pansy, for example. It could, however, be used as a starting point for determining image specific background distributions for the classes we excluded from the 17 category data set (e.g. fields of flowers or very small flowers: Snowdrops, Lily of the Valley's, Cowslips and Bluebells). The segmentation would then be obtained using

the CRF optimization, as here.

In the next chapter, we will evaluate how our segmentation scheme improves classification. The geometric model we have introduced here is a step towards extracting features that are invariant to affine transformations, which we will explore further in chapter 6.

Chapter 5

Classification

In this chapter, we address the problem of flower classification. What distinguishes one flower from another can sometimes be the colour, e.g. Bluebell vs Sunflower, sometimes the shape, e.g. Daffodil vs Dandelion, and sometimes patterns on the petals, e.g. Pansies vs Tiger Lilies etc. The difficulty lies in finding suitable features to represent colour, shape, patterns etc. and and building a classifier capable of learning which feature or features to use.

Here, we demonstrate that by developing a visual vocabulary which explicitly represents various aspects that distinguish one flower from another and combining these, we can overcome ambiguities that exist between flower categories. In chapter 4, we presented an iterative segmentation scheme and we will now use it to segment the flowers before extracting the features. In section 5.1, we introduce the features and the motivation for them and evaluate the effectiveness of each feature using a Support Vector Machine (SVM) classifier on the 17 category dataset (section 3.1) and the 13 categories which the segmentation scheme was developed on. We also

evaluate how the iterative segmentation scheme improves the performance compared to a segmentation obtained by minimizing a CRF using graphcuts. We also compare the SVM classifier to classification using a k-nearest neighbour (kNN) classifier for each feature. In section 5.2 we investigate which features to use and how to best combine them. Finally, in section 5.3, we select the best features and investigate their performance on the 102 category dataset and compare this to using the method of Lazebnik *et al.* [54].

5.1 Features

Some flowers have very distinctive colour, some very distinctive shape and some very distinctive patterns, but for most flowers it is the combination of these aspects that make them distinctive. We want to create a visual vocabulary for each of these aspects. In this section we investigate how to capture these aspects and evaluate the performance of different low level features. The features used are HSV values, MR8 filters [113], SIFT [65] features sampled both on the foreground region and its boundary, and histogram of gradient orientations (HOG) [26]. We evaluate the performance on the 17 category database (section 3.1) and on the 13 category subset which was used for evaluating the segmentation scheme. For each feature we present the performance using three different segmentations: the baseline system of Saitoh *et al.* [91], a CRF minimized using graph cuts, and our iterative segmentation scheme (chapter 4). We evaluate the performance on a validation set using three different data splits. Once the parameters have been determined on the validation set, the classifiers are retrained using all the available training data (both the training and validation sets) and classification is then done on the test set.

Creating visual words. The features presented in this chapter are all represented as visual words. A visual vocabulary is created by extracting a feature f from the N training images and clustering these using k-means clustering. Given a set of cluster centres (visual words) w_i^f, $i = 1, 2, ..., V_f$, each image I is then represented by a V_f dimensional normalized frequency histogram $n(w^f|I)$. For each feature the number of words in the vocabulary V_f is optimized on the validation set. A novel image j is classified using a one-vs-rest SVM and the kernel $K(i, j) = \exp(-\mu_f \chi^2(w_i^f, w_j^f))$. The parameter μ_f is set to 1/mean value of the χ^2 distances between the vectors w_i^f over all the training images [121].

5.1.1 Colour

We want to create a vocabulary to represent the colour of a flower. Some flowers exist in a wide variety of colours, but many have a distinctive colour. The colour of a flower can help narrow down the possible species, but it will not enable us to determine the exact species of the flower. For example, if a flower is yellow then it could be a Daffodil or a Dandelion, but it could not be a Bluebell.

Images of flowers are often taken in natural outdoor scenes where the lighting varies with the weather and time of day. In addition, flowers are often more or less transparent, and specular highlights can make the flower appear lighter or even white. These environmental factors cause large variations in the measured colour, which in turn leads to confusion between classes. Figure 5.1 shows an example of these variations.

Figure 5.1: Buttercups have a characteristic colour. In the images this is not apparent because of the variations in illumination conditions.

HSV: One way to reduce the effect of illumination variations is to use a colour space which is less sensitive to it. Hence, we describe the colour using the HSV colour space.

We optimize the number of words on the validation set. Figure 5.2 shows how the average recognition rate varies with the number of words using a SVM. We can see that the classification performance does not vary much with the number of words. The results show that using some image specific knowledge to segment the image (the iterative scheme and Saitoh *et al.* [91]) leads to better classification performance than using a general colour model (the initial graph cut segmentation). For the 13 categories, the iterative segmentation scheme has the better recognition rate over the vocabulary range, and reaches $70.0 \pm 3.7\%$ using 1100 words. For the 17 categories, the recognition rate of Saitoh *et al.* is on par with the iterative segmentation scheme over the entire vocabulary range. In fact, the highest recognition rate of $63.6 \pm 2.6\%$ at 1200 words is obtained using Saitoh *et al.*, although the best performance of the iterative segmentation scheme, $63.3 \pm 0.9\%$ at 600 words, is within the standard deviation of Saitoh's method. The reason for the loss in the performance advantage of the iterative scheme compared to Saitoh is that the four additional categories (Bluebells, Cowslips, Lily-of-the-Valey's and Snowdrops) contain many images of

fields of flowers; sufficient statistics for the colour of these flowers can be obtained using Saitoh's method, but the iterative scheme is not designed for such images.

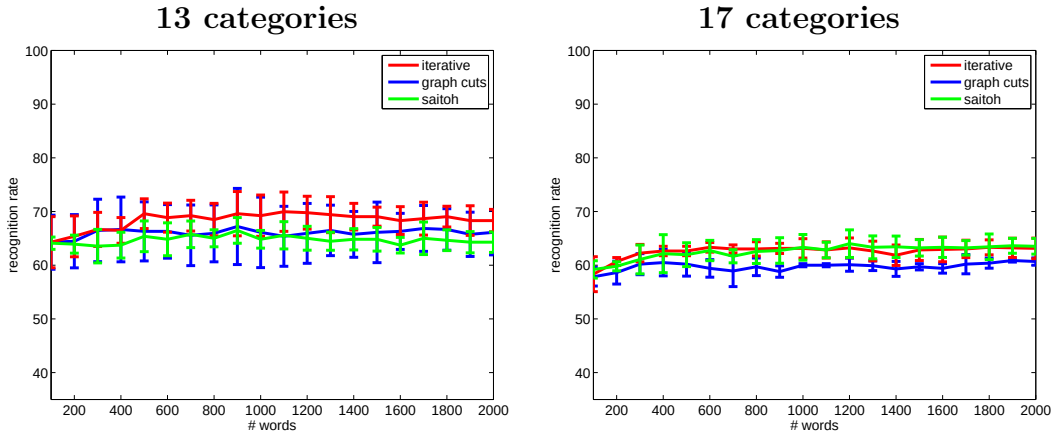


Figure 5.2: Optimization of **HSV** vocabulary size – Average recognition rate is shown for a Support Vector Machine classifier. The results are shown for 13 categories and for the full 17 categories. The results are presented for the iterative segmentation scheme, the initial graph cut segmentation and the baseline segmentation method of Saitoh *et al.* [91]. The results are shown on the validation sets and are averaged over three random permutation of training and validation sets. The best results are obtained using the iterative segmentation scheme.

Once the vocabulary size has been optimized, the SVM can be retrained using both the training and validation set. The average recognition rate obtained using our segmentation scheme is then $66.5 \pm 3.2\%$ for the 13 categories and $65.9 \pm 1.3\%$ for the 17 categories. Table 5.1 shows the confusion matrix averaged over the three splits of the datasets. We can see that some classes, e.g. Tiger Lilies, are very well distinguished by colour, and that there are some classes with very high intra-class confusion, e.g. Snowdrops and Lilies-of-the-Valley. Also note that we obtain good performance for the Bluebells and the Cowslips, for which our segmentation scheme was not developed. Using Saitoh's method we obtain a recognition rate of $66.8 \pm 0.9\%$ for the 13 categories and $65.8 \pm 1.1\%$ for the 17 categories, which is equivalent to that of our segmentation scheme. However, with Saitoh's method we

often obtain the colour information from partially segmented flower regions. Whilst these regions give sufficient information about the colour of the flower, we show in the following sections that the missing regions are important for classification using the other features.

HSV

	Bluebell	Buttercup	Colt's Foot	Cowslip	Crocus	Daffodil	Daisy	Dandelion	Fritillary	Iris	Lily Valley	Pansy	Snowdrop	Sunflower	Tiger Lily	Wild Tulip	Windflower
Bluebell	81.7				5.0				1.7		3.3	5.0	3.3				
Buttercup		60.0	5.0	1.7		6.7		6.7						3.3		16.7	
Colt's Foot		1.7	75.0	5.0				6.7						1.7	3.3	6.7	
Cowslip				78.3		3.3		1.7				1.7	3.3	1.7		8.3	1.7
Crocus	5.0				65.0		5.0		3.3	3.3		6.7	5.0	3.3			3.3
Daffodil		8.3		10.0		56.7		5.0			5.0	5.0		3.3		5.0	1.7
Daisy					3.3		75.0	1.7		3.3	1.7		8.3	1.7			5.0
Dandelion		1.7	13.3	1.7				73.3			1.7			1.7	1.7	5.0	
Fritillary	1.7			3.3	1.7	1.7	1.7		75.0	1.7	10.0		3.3				
Iris	8.3	1.7	1.7		10.0				3.3	55.0		1.7	6.7		5.0		6.7
Lily Valley	3.3			5.0	3.3	1.7			3.3	5.0	51.7		20.0	1.7		1.7	3.3
Pansy	5.0		3.3	3.3	6.7		5.0			6.7		60.0	1.7	3.3		1.7	3.3
Snowdrop	5.0			1.7			10.0		5.0	5.0	30.0		38.3			1.7	3.3
Sunflower		5.0	5.0	3.3		1.7		8.3	1.7					73.3		1.7	
Tiger Lily				1.7					1.7					1.7	95.0		
Wild Tulip		13.3	8.3	5.0	3.3	15.0	1.7	10.0			6.7		5.0	5.0		25.0	1.7
Windflower					1.7		1.7			1.7	1.7	3.3	8.3				81.7

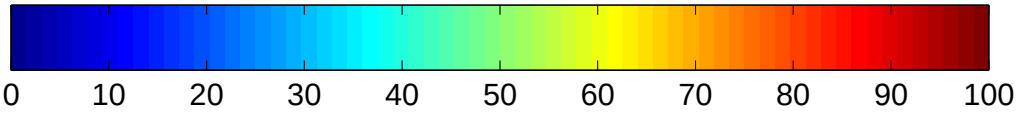


Table 5.1: Confusion matrix using the iterative segmentation scheme for the **HSV** features for 17 categories. The confusion matrix is averaged over three different test set splits. Note that classes of similar colour such as the white Snowdrops and Lilies-of-the-Valley, and the yellow Dandelions and Colt's Feet have a stronger confusions. The average recognition rate is $65.9 \pm 1.3\%$.

5.1.2 Texture

Some flowers have characteristic patterns on their petals. These patterns can be more distinctive, such as the Pansy's stripes, the Fritillary's checks or the Tiger-Lily's dots (figure 5.3), or more subtle in the form of characteristic veins in the petals. The subtle patterns are sometimes difficult to distinguish due to illumination conditions – a problem that also affects the appearance of more distinctive patterns. In addition, some flowers have fine petal structure (e.g. Dandelions) and some have spikes (e.g. Globe Thistle), which can be described using texture features.

Figure 5.3: Flowers with distinctive patterns. From left to right: Pansy with distinctive stripes, Fritillary with distinctive checks and Tiger Lily with distinctive dots.

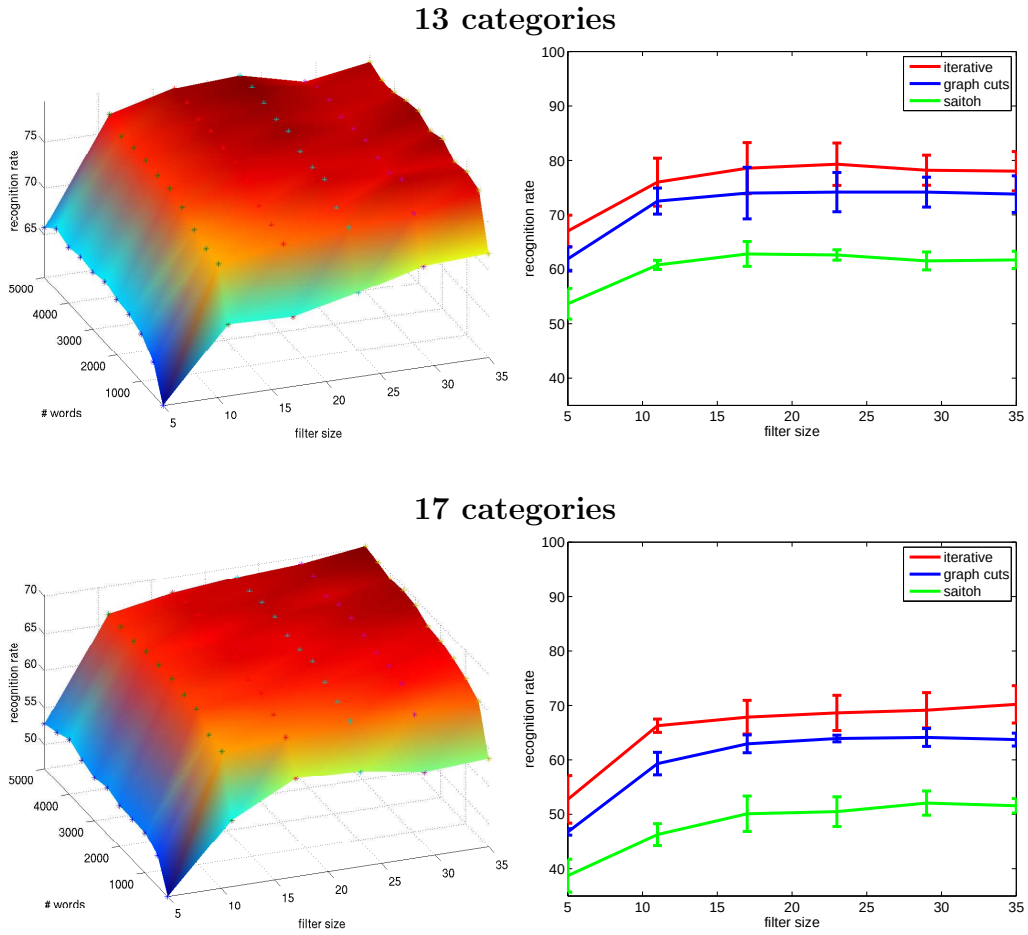
MR8 filter banks: One way of describing the texture is by convolving the images with MR8 filter bank introduced by Varma and Zisserman [113]. The filter bank consists of filters at multiple orientation. Rotation invariance is obtained by choosing the maximum response over orientations. A vocabulary is created by clustering the descriptors and the frequency histograms $n(w^t|I)$ are obtained. The classification performance is optimized on the validation set for varying filter and vocabulary sizes. We vary the size of the square support regions of the filters from 5 to 35 pixels. The vocabulary size is varied between 100 and 5000 words. Figure 5.4 shows

how the performance varies with filter and vocabulary size. We see that small filters typically lead to deterioration in performance. In contrast to the HSV features, Saitoh's segmentation method misses regions that are important for classifying the flower and the best performance is clearly obtained using the iterative segmentation scheme. The best average performance on the validation sets is $79.3 \pm 3.9\%$ for the 13 categories, which is obtained using 4500 words and a filter size of 23, and $70.2 \pm 3.2\%$ for the 17 categories, which is obtained using 3500 words and a filter size of 35. Note that performance does not vary much with the filter size for filters of size greater than 15. Larger filters increase the computational complexity. In addition, when using larger filter we are essentially describing shape instead of texture.

The average recognition rate on the test set after retraining the SVM using both the training and validation data is $82.9 \pm 3.3\%$ for 13 categories and $67.5 \pm 1.1\%$ for 17 categories. Table 5.2 shows the confusion matrix. Note that the MR8 filter perform well for small flowers, e.g. Snowdrops and Bluebells, and also for classes with many thin petals, e.g. Sunflowers and Dandelions.

5.1.3 Shape

The shape of individual petals, their configuration, and the overall shape of the flower can all be used to distinguish between flowers. In figure 5.5 it can be seen that although the overall shape of the Windflower (left) and the Buttercup (middle) are similar, the Windflower's petals are more pointed. The Daffodil (right) has petals more similar to that of the Windflower, but the overall shape is very different due to the tubular shape corolla in the middle of the Daffodil. Changes in viewpoint and occlusions of course change the perceived shape of the flower. The difficulty of describing the shape is increased by the natural deformations of a flower. The



a) SVM b) SVM (fixed $\#words$)

Figure 5.4: Optimization of the vocabulary and filter size for the **MR8** filter responses – Average recognition rate is shown for a) a SVM and b) shows how the performance varies using the SVM whilst fixing the number of words (4500 and 3500 for the 13 and 17 categories respectively) and varying the filter size for three different segmentation schemes: the iterative segmentation scheme, the initial graph cut segmentation and the segmentation method of Saitoh *et al.* [91]. The top row shows the results for 13 categories and the bottom row for the full 17 categories. The results are shown on the validation sets and are averaged over three random permutation of training and validation sets. The best results are obtained using the iterative segmentation scheme.

petals are often very soft and flexible and can bend, curl, twist etc., which makes the shape of a flower appear very different. The shape of a flower also changes with the age of the flower and petals might even fall off.

MR8

	Bluebell	Buttercup	Colt's Foot	Cowslip	Crocus	Daffodil	Daisy	Dandelion	Fritillary	Iris	Lily Valley	Pansy	Snowdrop	Sunflower	Tiger Lily	Wild Tulip	Windflower
Bluebell	70.0	1.7	1.7	5.0	1.7				5.0	1.7			10.0		3.3		
Buttercup		80.0		3.3		3.3					1.7	1.7				5.0	5.0
Colts Foot		3.3	75.0	1.7	6.7		1.7	5.0	3.3						3.3		
Cowslip	16.7			55.0		1.7			8.3	1.7			8.3		5.0	3.3	
Crocus	5.0	1.7	1.7	1.7	46.7	5.0	1.7	3.3	1.7	1.7	1.7	8.3	3.3	5.0	6.7	3.3	1.7
Daffodil		5.0		6.7	10.0	51.7		1.7		13.3	1.7		1.7		3.3	5.0	
Daisy				1.7		5.0	76.7				1.7	1.7	6.7	3.3	3.3		
Dandelion		1.7	8.3		3.3			85.0				1.7					
Fritillary		1.7	1.7	3.3		6.7	1.7	1.7	71.7				6.7		3.3	1.7	
Iris					3.3	6.7		1.7		86.7					1.7		
Lily Valley	5.0	1.7		15.0	1.7	5.0			3.3	1.7	48.3	1.7	8.3	6.7		1.7	
Pansy	1.7	1.7	3.3	1.7	5.0				1.7			65.0	1.7	6.7	1.7		10.0
Snowdrop	1.7	6.7		3.3	1.7		1.7			5.0			70.0	1.7	5.0	3.3	
Sunflower		3.3			5.0		1.7					1.7		86.7	1.7		
Tiger Lily	6.7			3.3			3.3		1.7				5.0	6.7	73.3		
Wild Tulip	11.7	1.7		8.3	5.0	10.0		3.3	3.3	1.7	6.7		8.3	1.7	3.3	35.0	
Windflower		8.3		3.3		1.7	8.3			5.0						1.7	71.7

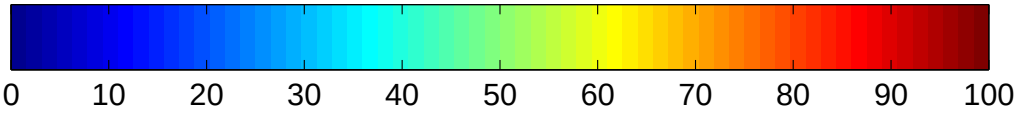


Table 5.2: Confusion matrix using the iterative segmentation scheme for the MR8 features for 17 categories. The confusion matrix is averaged over three different test set splits. The MR8 filter perform well for small classes, e.g. Snowdrops and Bluebells, and also for classes with many thin petals, e.g. Sunflowers and Dandelions. The average recognition rate is $67.5 \pm 1.1\%$.

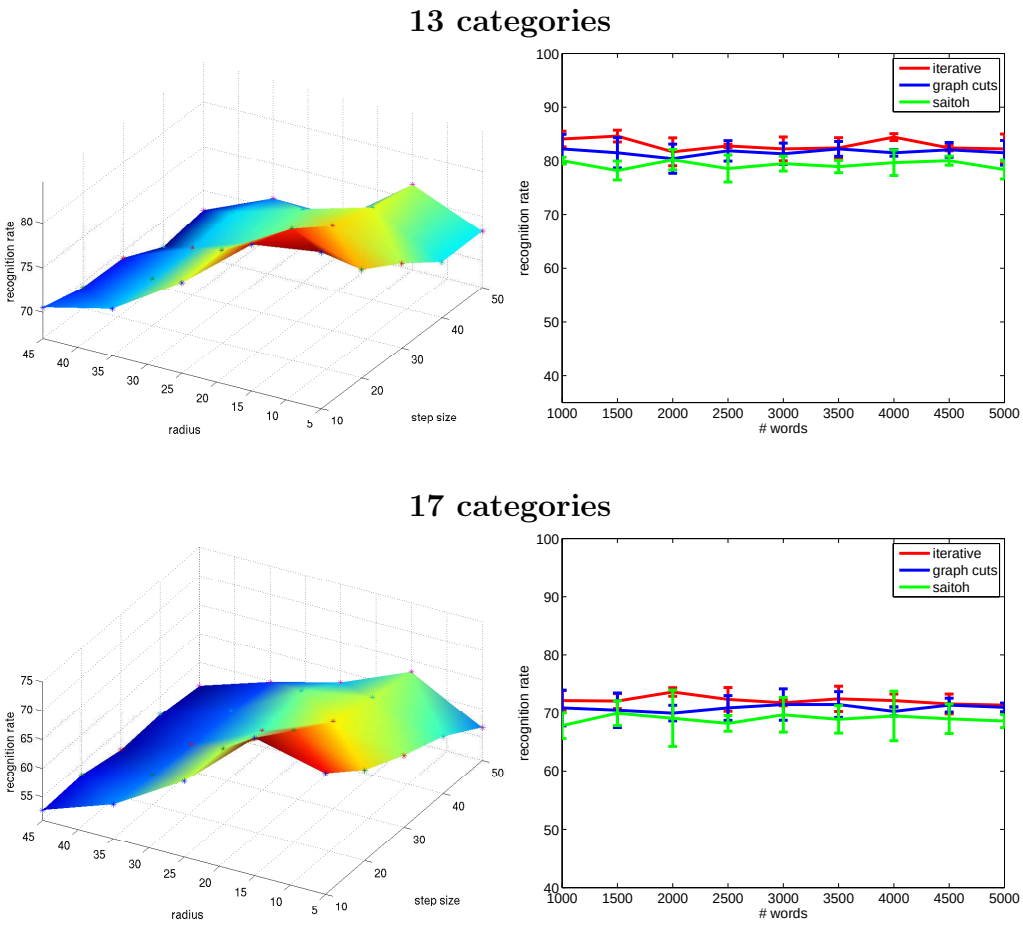
SIFT on the foreground region: By computing SIFT features from the internal region of the foreground, we can describe both the texture and the local shape of the flower. The internal SIFT [65] descriptors are computed at points on a regular grid with spacing S pixels over the foreground flower region. At each grid point the descriptors are computed over circular support patches with radii R pixels. Only the grey value is used (not colour), and the resulting SIFT descriptor is a

Figure 5.5: Images of similar shapes. Note that the windflower's (middle) petals are more pointy than the buttercup's (left). The daffodil (right) and the windflower have similar shaped petals, but are overall quite different due to the daffodil's tubular corolla.

128 dimensional vector. To cope with empty patches, all SIFT descriptors with $L2$ norm below a threshold (200) are zeroed. Note, we use rotationally invariant features. We obtain $n(w^f|I)$ through vector quantization in the same way as for the colour features. The step size S is optimized on the validation set over a range of 10 to 50 pixels and the radius of the circular support patches R is searched for over a range of 5 to 45 pixels. Figure 5.6 shows how the performance varies for the different parameters. Column a) shows the results as the radius and step size is changed and column b) shows the results as the vocabulary size varies for the three different segmentation schemes. Again, changing the vocabulary size has very little affect on the performance. A smaller radius and step size is however beneficial. The best performance on the validation set is $84.6 \pm 1.1\%$ and $75.0 \pm 2.4\%$ for the 13 and 17 categories respectively. The best performance is obtained using 1500 words, a radius of 15 pixels and a step size of 10 pixels. Since the SIFT features are extracted from within the segmented region obtaining a perfect segmentation is less important. As the results show the choice of segmentation scheme does not have a significant effect on the results.

The performance on the test set is $85.2 \pm 1.3\%$ and $72.1 \pm 5.7\%$ for 13 and 17 categories respectively. Table 5.3 shows the confusion matrix. We can see that the

internal SIFT descriptors perform very well on categories with many thin petals, e.g. Sunflowers and Dandelions, and also on patterned flowers like Irises and Pansies.



a) SVM ($\#words = 1500$) b) SVM ($R = 15, S = 10$)

Figure 5.6: Optimization of the vocabulary, radius and step size for the **internal SIFT** features – Column a) shows the average recognition rate while varying the radius and step size of the descriptor and column b) shows the recognition rate as the number of words varies for the three different segmentation schemes. The iterative segmentation scheme and Saitoh's method have comparable performance. See figure 5.4 for a more detailed caption.

Internal SIFT

	Bluebell	Buttercup	Colt's Foot	Cowslip	Crocus	Daffodil	Daisy	Dandelion	Fritillary	Iris	Lily Valley	Pansy	Snowdrop	Sunflower	Tiger Lily	Wild Tulip	Windflower
Bluebell	75.0			6.7	1.7						8.3		5.0		3.3		
Buttercup		86.7	1.7		1.7		1.7			3.3		5.0					
Colts Foot		8.3	78.3					8.3	3.3		1.7						
Cowslip	18.3			41.7	13.3	1.7					3.3		6.7		11.7	3.3	
Crocus	5.0	3.3	1.7	11.7	45.0	1.7	1.7			5.0	1.7	3.3	3.3		5.0	10.0	1.7
Daffodil	6.7			3.3	1.7	66.7		1.7		3.3			5.0	1.7		6.7	3.3
Daisy				6.7			78.3	5.0	1.7				3.3	5.0			
Dandelion			5.0					93.3		1.7							
Fritillary		1.7			8.3				73.3		1.7	1.7	3.3		1.7	8.3	
Iris	1.7	3.3								93.3		1.7					
Lily Valley	13.3	3.3		5.0	3.3	5.0			1.7		56.7	5.0	1.7		1.7	3.3	
Pansy		1.7		3.3	5.0		1.7		1.7	1.7	3.3	76.7					5.0
Snowdrop	10.0	3.3		3.3	5.0						5.0		66.7		1.7	3.3	1.7
Sunflower							1.7	1.7			1.7			95.0			
Tiger Lily	6.7	3.3		5.0	6.7				1.7	1.7	1.7		5.0	1.7	66.7		
Wild Tulip	6.7		1.7	3.3	5.0	3.3			3.3		8.3		10.0		5.0	51.7	1.7
Windflower		6.7		3.3		5.0							1.7			3.3	80.0

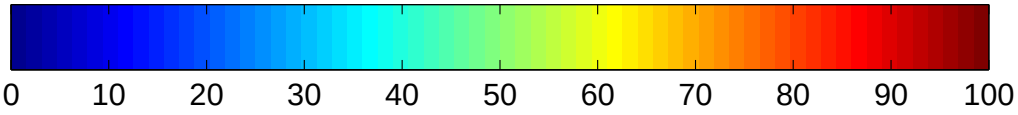


Table 5.3: Confusion matrix using the iterative segmentation scheme for the **internal SIFT** features for 17 categories. The confusion matrix is averaged over three different test set splits. The internal SIFT features perform well for classes with many thin petals, e.g. Sunflowers and Dandelions, but also for flowers with patterns, such as Irises. The average recognition rate is $72.0 \pm 5.7\%$.

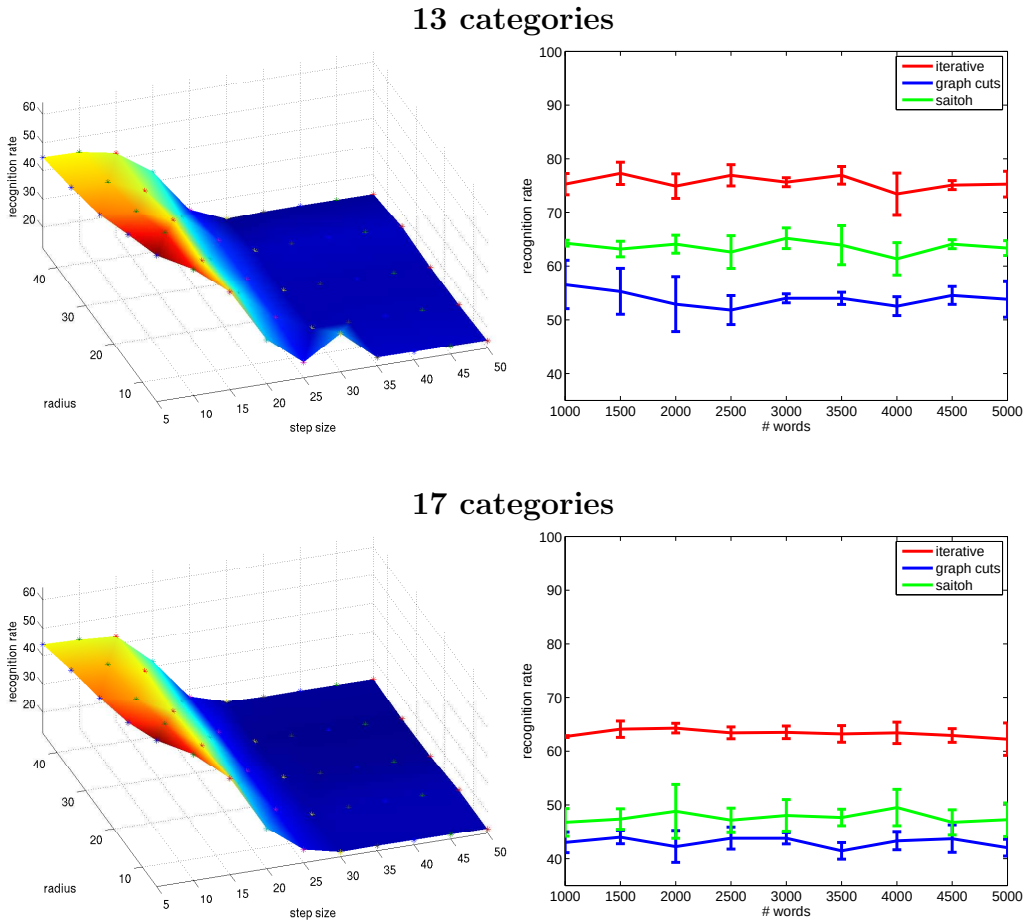
SIFT on the foreground boundary: The boundary of the segmentation gives us the boundary of the flower. By sampling SIFT features on the boundary of the flower we can can give greater emphasis (over the internal features) to the local shape of the boundary. A similar boundary feature was used in [66]. The 128 dimensional SIFT descriptors with radii R pixels are computed at each step S along

the boundary. In a similar manner to the SIFT features for the internal region, only the grey value is used. $n(w^b|I)$ is obtained by clustering only the boundary SIFTs, i.e. separate vocabularies are used for the boundary and internal SIFT features.

The step size S and the radius of the circular support patches R are optimized on the validation set over a range of 5 to 50 pixels. Figure 5.7 shows how the performance varies for the different parameters. Columns a) shows the results as the radius and step size is changed and column b) shows the results as the vocabulary size varies for the three different segmentations. Note that the iterative segmentation scheme shows a significant improvement for both the 13 and 17 categories compared to the method of Saitoh *et al.* [91] which only achieves a recognition rate on par with minimizing a CRF using graphcuts. This indicates that the iterative segmentation scheme provides a much more accurate segmentation of the boundary. As with the internal SIFT a smaller radius and dense sampling improves the performance. The best performance on the validation set is $77.3 \pm 2.1\%$ and $64.3 \pm 0.9\%$ for the 13 and 17 categories respectively. The best performance for both datasets is obtained using 1500 words, a radius of 5 pixels and a step size of 5 pixels.

Again, we retrain the SVM using both the training and validation sets as training data. The average recognition rate obtained is $79.1 \pm 0.5\%$ and $65.3 \pm 2.5\%$ for 13 and 17 categories respectively. Table 5.4 shows the confusion matrix. We can see that it works well for categories with big petals, e.g. Buttercups and Windflowers.

Histogram of Gradients: HOG features [26], are similar to SIFT features, except that they use an overlapping local contrast normalization between cells in a grid. However, instead of being applied to local regions (of radius R in the case of SIFT), the HOG is applied here over the entire flower region (and it is not made rotation



a) SVM ($\#words = 1500$) b) SVM ($R = 5, S = 5$)

Figure 5.7: Optimization of the vocabulary, radius and step size for the **boundary SIFT** features – Column a) shows the average recognition rate while varying the radius and step size of the descriptor and column b) shows the recognition rate as the number of words varies for the three different segmentation schemes. The best results are obtained using the iterative segmentation scheme. See figure 5.4 for a more detailed caption.

invariant). The segmentation is used to guide the computation of the HOG features. Unlike [26] here we cluster the individual HOG cells using K-means clustering and thus obtain $n(w^h|I)$ through vector quantization in the same way as for the previous features.

Figure 5.8 shows the performance for different vocabulary sizes. Again, the best performance is obtained using the iterative segmentation scheme. A small

Boundary SIFT

	Bluebell	Buttercup	Colt's Foot	Cowslip	Crocus	Daffodil	Daisy	Dandelion	Fritillary	Iris	Lily Valley	Pansy	Snowdrop	Sunflower	Tiger Lily	Wild Tulip	Windflower
Bluebell	68.3	1.7		10.0					6.7		6.7	1.7		1.7	1.7	1.7	
Buttercup		85.0		1.7	1.7						3.3	1.7	1.7		1.7		3.3
Colts Foot			83.3	3.3				8.3	3.3						1.7		
Cowslip	8.3	1.7		43.3		8.3		1.7	1.7	11.7	8.3	1.7	1.7	1.7	8.3	1.7	
Crocus	6.7	1.7	1.7	6.7	36.7	1.7	3.3		5.0	5.0	1.7	5.0	10.0	1.7	1.7	3.3	8.3
Daffodil	1.7	1.7		1.7	5.0	58.3			3.3	1.7	3.3	3.3	8.3	1.7	1.7	3.3	5.0
Daisy			3.3	1.7			73.3	5.0	1.7	1.7	1.7		1.7	5.0	1.7		3.3
Dandelion			16.7		3.3			78.3							1.7		
Fritillary	3.3			3.3	1.7	8.3		1.7	63.3		1.7	3.3	6.7		5.0		1.7
Iris	5.0								1.7	91.7				1.7			
Lily Valley	11.7			5.0	3.3	1.7			3.3		51.7		13.3		10.0		
Pansy	5.0	6.7		3.3	3.3				6.7	5.0		53.3		1.7	3.3		11.7
Snowdrop	6.7			3.3	1.7	5.0			1.7	1.7	5.0		53.3	8.3	1.7	8.3	3.3
Sunflower	10.0	1.7			1.7	1.7	3.3	1.7	3.3		1.7	1.7	1.7	71.7			
Tiger Lily	10.0	1.7		1.7	5.0	3.3			3.3	10.0			5.0	1.7	55.0	3.3	
Wild Tulip	3.3	1.7		1.7	1.7	5.0			6.7		3.3	1.7	5.0	6.7	6.7	56.7	
Windflower		5.0				1.7			1.7	3.3				1.7			86.7

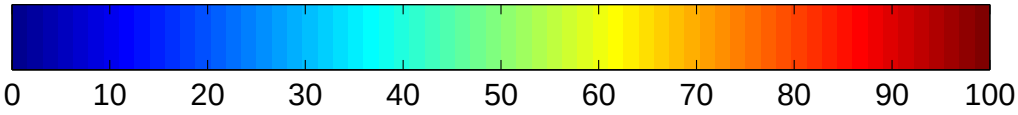


Table 5.4: Confusion matrix using the iterative segmentation scheme for the **boundary SIFT** features for 17 categories. The confusion matrix is averaged over three different test set splits. Results are computed using the iterative segmentation scheme and a SVM. The boundary SIFT features perform well for classes with large petals, e.g. Buttercups and Windflowers. The average recognition rate is $65.3 \pm 2.5\%$.

vocabulary leads to poor performance, best performance is obtained using 1500 words for the 13 categories and 2500 for the 17 categories. The average recognition rate for these are $71.8 \pm 1.4\%$ and $58.2 \pm 2.1\%$ respectively.

Table 5.5 shows the confusion matrix for the 17 category test set using the iterative segmentation scheme and a SVM trained on both the training and validation

sets. The HOG features perform well for classes with a disk structure, e.g. Sunflowers, Dandelions, Buttercups and Windflowers, which mainly have gradient in the radial direction. The average recognition rate is $58.6 \pm 0.2\%$ for 17 categories. The average recognition rate for 13 categories is $74.5 \pm 2.0\%$.

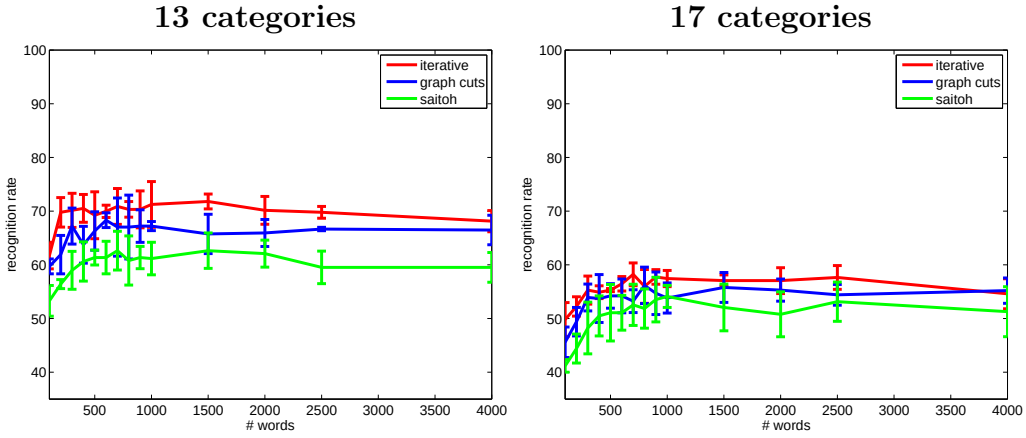


Figure 5.8: Optimization of **HOG** vocabulary size - Average recognition rate using a Support Vector Machine. Results are shown for 13 categories and for the full 17 categories. The results are presented for the iterative segmentation scheme, the initial graph cut segmentation and the segmentation method of Saitoh *et al.* [91]. The results are shown on the validation sets and are averaged over three random permutation of training and validation sets. The best results are obtained using the iterative segmentation scheme.

Comparison with k-Nearest Neighbour classifier We have so far evaluated results for five different features using three different segmentation schemes and a Support Vector Machine (SVM). We now compare the SVM classifier to a k-Nearest Neigbour (kNN) classifier, where $k = 1, 7, 15$. The images are segmented using our iterative segmentation scheme. Figure 5.9 shows the results for both the 13 and 17 categories for all features. The error rate of a kNN classifier approaches the Bayes optimal as the sample size goes to infinity. However, given our limited amount of training samples it is clear that the SVM classifier outperforms the kNN classifier.

HOG

	Bluebell	Buttercup	Colt's Foot	Cowslip	Crocus	Daffodil	Daisy	Dandelion	Fritillary	Iris	Lily Valley	Pansy	Snowdrop	Sunflower	Tiger Lily	Wild Tulip	Windflower
Bluebell	38.3	1.7	1.7	11.7		1.7	1.7			3.3	11.7	1.7	8.3	1.7	15.0	1.7	
Buttercup		80.0		1.7		1.7		1.7	3.3	1.7		3.3	1.7	3.3		1.7	
Colts Foot			66.7	1.7	3.3	1.7		16.7		8.3					1.7		
Cowslip	13.3	5.0	1.7	31.7	6.7	1.7			6.7	3.3	3.3	1.7	1.7	1.7	11.7	8.3	1.7
Crocus	3.3		1.7	10.0	38.3		1.7	1.7	1.7	6.7	3.3		13.3		5.0	8.3	5.0
Daffodil	5.0	10.0		10.0	3.3	43.3		1.7	1.7	5.0		1.7	3.3	5.0	1.7	6.7	1.7
Daisy	5.0			3.3	3.3		58.3	3.3	1.7	3.3	1.7	5.0	6.7	3.3	5.0		
Dandelion	3.3	3.3	5.0		1.7	3.3		76.7		3.3					3.3		
Fritillary	5.0	1.7			10.0	3.3	1.7	1.7	61.7	1.7			6.7		6.7		
Iris	1.7	13.3		6.7	1.7					71.7	1.7	1.7			1.7		
Lily Valley	3.3			6.7	1.7	3.3	5.0		1.7	1.7	58.3	1.7	10.0			6.7	
Pansy	1.7	3.3		5.0	3.3	8.3			1.7	1.7		60.0				10.0	5.0
Snowdrop	6.7			1.7	6.7	1.7			5.0	3.3	1.7		66.7	1.7	1.7	3.3	
Sunflower		3.3		6.7				1.7						88.3			
Tiger Lily	13.3	1.7		5.0	10.0	1.7	1.7	5.0	3.3	1.7				3.3	45.0	6.7	1.7
Wild Tulip		6.7		13.3	3.3	8.3			3.3	1.7	6.7	8.3	1.7	5.0	5.0	36.7	
Windflower		1.7		5.0		3.3	1.7		1.7		1.7	5.0	1.7		1.7	1.7	75.0

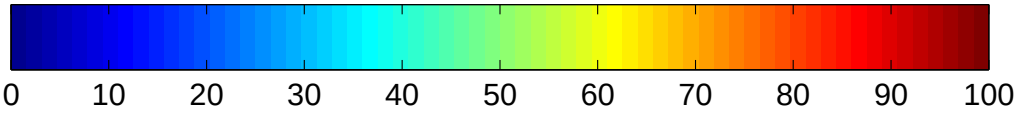
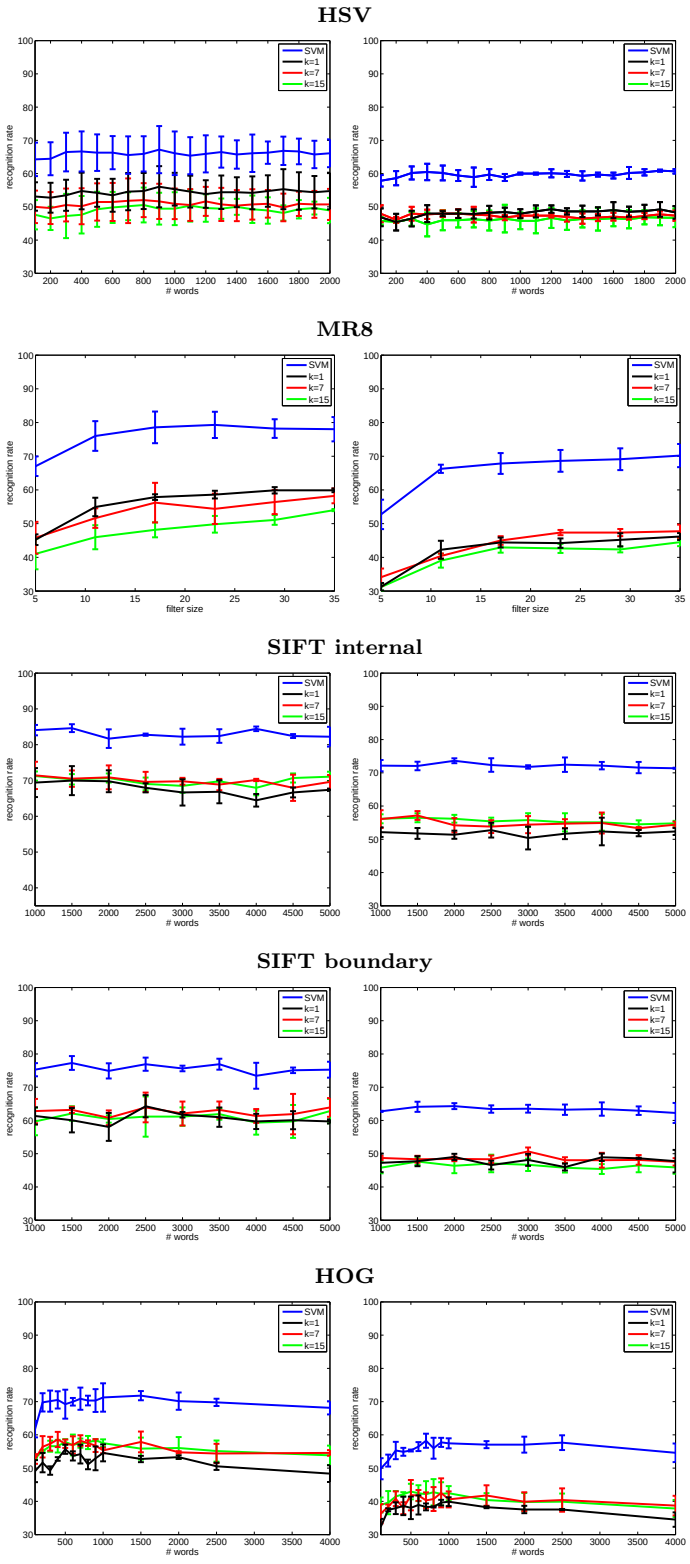


Table 5.5: Confusion matrix using the iterative segmentation scheme for the **HOG** features for 17 categories. The confusion matrix is averaged over three different test set splits. The boundary HOG features perform well for classes with a disk structure, e.g. Sunflowers, Dandelions, Buttercups and Windflowers. The average recognition rate is $58.6 \pm 0.2\%$.

Furthermore, each one-vs-all SVM classifier will rescale the margin. This effectively stretches or shrinks the distances to the margin for each task depending on how close the support vectors are to the hyperplane, i.e. if the support vectors are close to the hyperplane the space will be stretched and if they are far away from the hyperplane the space will be shrunken.



a) 13 categories b) 17 categories

Figure 5.9: SVM vs kNN classifier - The graphs shows the comparison of recognition rate using a SVM and kNN classifiers. Column a) shows the results for 13 categories and column b) for 17 categories. The blue line shows the recognition rate using a SVM, the black line using a kNN classifier with $k = 1$, the red line with $k = 7$ and the green line with $k = 15$.

Summary: We have evaluated five different features describing three different aspects. Comparing the results using three different segmentation schemes: the baseline system of Saitoh *et al.* [91], minimizing a CRF using graphcuts, and our iterative segmentation scheme, has shown that overall the biggest improvement is obtained using our segmentation scheme for both the 13 categories the scheme was developed on, and on the full 17 category dataset (section 3.1). It is worth noting that Saitoh's method performs on par with our method for the HSV and SIFT internal features. This is because capturing the entire flower region is not crucial for these features. Furthermore using a SVM is significantly better than using a Nearest Neighbour classifier. The summary of the average recognition rates on the 17 category test set (averaged over three different splits) is listed in table 5.6. We can see that most features except the HOG features give quite an equivalent performance. From evaluating the individual features we can deduce that they are good at classifying different categories, but it is also worth noting that the MR8 filters and the internal SIFT features show quite a strong overlap between their confusion matrices with the SIFT features constantly outperforming the MR8 filters. There are also some categories which are not well distinguished by any of the features, for example the Wild Tulips. In the next section, we look at combining useful features out of this feature pool in such a way that superior performance is obtained for all categories.

	Bluebell	Buttercup	Colt's Foot	Cowslip	Crocus	Daffodil	Daisy	Dandelion	Fritillary	Iris	Lily Valley	Pansy	Snowdrop	Sunflower	Tiger Lily	Wild Tulip	Windflower	Average
HSV	**81.7**	60.0	75.0	**78.3**	**65.0**	56.7	75.0	73.3	**75.0**	55.0	51.7	60.0	38.3	73.3	**95.0**	25.0	81.7	**65.9**
MR8	70.0	80.0	75.0	55.0	46.7	51.7	76.7	85.0	71.7	86.7	48.3	65.0	**70.0**	86.7	73.3	35.0	71.7	**67.5**
SIFT int	75.0	**86.7**	78.3	41.7	45.0	**66.7**	**78.3**	**93.3**	73.3	**93.3**	56.7	**76.7**	66.7	**95.0**	66.7	51.7	80.0	**72.1**
SIFT bd	68.3	85.0	**83.3**	43.3	36.7	58.3	73.3	78.3	63.3	91.7	51.7	53.3	53.3	71.7	55.0	**56.7**	**86.7**	**65.3**
HOG	38.3	80.0	66.7	31.7	38.3	43.3	58.3	76.7	61.7	71.7	**58.3**	60.0	66.7	88.3	45.0	36.7	75.0	**58.6**

Table 5.6: Recognition performance on the test set using the iterative segmentation scheme.

5.2 Feature combination

In this section, we use the 17 category database to find useful features for flower classification. We want to combine the features in a flexible manner in a way such that the discriminative power of each feature is preserved but also such that combining them leads to improvement in performance. A weighted linear combination of kernels is used, with one kernel corresponding to each feature. The final kernel has the following form for two data points i and j:

$$K(i,j) = \sum_{f \epsilon F} \beta_f \exp\left(-\mu_f \chi_f^2(x_f(i), x_f(j))\right) \tag{5.1}$$

where x_f is the feature vector for descriptor f (e.g. the normalized bag-of-visual-words histogram in the case of local shape), and β_f is the weight for feature f. $\chi^2(x,y)$ is the symmetric Chi-squared distance between histograms x and y. Note, $K(x,y) = \exp(-\mu_f \chi^2(x,y))$ is a Mercer kernel, and consequently (5.1) is a Mercer kernel by the sum rule. The parameter μ_f is set to 1/mean value of the χ^2 distances between the vectors x_f over all the training images [121]. We use a one-vs-rest classifier. We evaluate three different ways of setting the kernel weights β_f:

1. Equal weights – All the feature kernels are simply added together, i.e. $\beta_f = 1$.

2. Global weights – A global weight is determined for each feature, i.e. all β_f remain constant for all one-vs-rest classifications. We exhaustively search for the best weights on the validation set.

3. Local weights – Following [12], the weights β_f are learnt for each class in the one-vs-rest classifier. The weights are determined on the validation set using the optimization method of Varma and Ray [112].

In order to determine which features benefit the performance, we evaluate each combination of 2, 3, 4 and 5 features. Table 5.7 shows the average recognition rate on the test set for all combinations of 2 features. The number in brackets are the global weights learnt for each feature. The best performance of $85.8 \pm 2.4\%$ is obtained by combining the internal SIFTs with the HSV and is the same for all weighting schemes, and as we can see for the global weights the features have been given equal weights. Generally, combining any of the shape/texture features with colour leads to the best performance, which follows our intuition.

1st feature	2nd feature	Equal β_f	Global β_f	Local β_f
Sift Int(1.00)	MR8(2.00)	77.7 ± 4.0	75.8 ± 3.8	77.9 ± 3.8
Sift Int(1.00)	Sift Bdy(0.75)	77.2 ± 2.5	77.8 ± 2.4	77.7 ± 2.7
Sift Int(1.00)	HSV(1.00)	$\mathbf{85.8 \pm 2.4}$	$\mathbf{85.8 \pm 2.4}$	$\mathbf{85.8 \pm 2.4}$
Sift Int(1.00)	HOG(1.00)	77.1 ± 3.1	77.1 ± 3.1	77.1 ± 3.1
MR8(1.00)	Sift Bdy(0.50)	76.2 ± 2.5	75.1 ± 3.2	76.4 ± 2.0
MR8(1.00)	HSV(0.25)	80.4 ± 1.2	81.0 ± 3.0	81.7 ± 2.9
MR8(1.00)	HOG(0.50)	72.3 ± 3.4	72.1 ± 2.4	72.9 ± 2.2
Sift Bdy(1.00)	HSV(0.25)	80.9 ± 1.8	81.3 ± 1.8	81.4 ± 1.3
Sift Bdy(1.00)	HOG(1.50)	73.2 ± 2.8	72.5 ± 2.5	73.8 ± 2.9
HSV(1.00)	HOG(1.50)	77.6 ± 3.7	77.3 ± 2.4	77.9 ± 3.5

Sift Int	$\mathbf{72.1 \pm 5.7}$
MR8	67.5 ± 1.1
Sift Bdy	65.3 ± 2.5
HSV	65.9 ± 1.3
HOG	58.6 ± 0.2

Table 5.7: Average recognition rates on the 17 category database for combinations of two features. The number in brackets next to each feature is the global weight. The right table shows the results for the individual features for comparison. The best performance is obtained by combining the internal SIFTs with the HSV features.

Table 5.8 shows the average recognition rate for all combinations of 3 features. Combining the two different SIFT features with the HSV features leads to the best performance, but an almost equally good performance is obtained by combining the internal SIFTs, the MR8 features and the HSV features. Note that here, the best performance ($87.2 \pm 2.1\%$) is slightly better for the local weights, but the difference between the weighting schemes is very small.

1st feature	2nd feature	3rd feature	Equal β_f	Global β_f	Local β_f
Sift Int(1.00)	MR8(1.00)	Sift Bdy(1.00)	80.8 ± 1.9	80.8 ± 1.9	80.8 ± 1.9
Sift Int(1.00)	MR8(0.75)	HSV(0.75)	87.0 ± 3.2	86.4 ± 2.7	$\mathbf{87.2 \pm 3.0}$
Sift Int(1.00)	MR8(1.00)	HOG(0.75)	80.4 ± 3.1	79.1 ± 2.9	80.7 ± 2.8
Sift Int(1.00)	Sift Bdy(1.00)	HSV(0.75)	86.7 ± 1.6	87.0 ± 1.9	$\mathbf{87.2 \pm 2.1}$
Sift Int(1.00)	Sift Bdy(1.00)	HOG(1.75)	78.7 ± 2.5	79.1 ± 2.2	79.2 ± 2.2
Sift Int(1.00)	HSV(2.00)	HOG(0.50)	85.8 ± 2.9	86.1 ± 3.2	86.9 ± 2.5
MR8(1.00)	Sift Bdy(0.25)	HSV(0.25)	83.5 ± 1.1	84.3 ± 2.8	84.4 ± 2.3
MR8(1.00)	Sift Bdy(1.25)	HOG(1.50)	77.6 ± 1.2	76.2 ± 2.5	78.2 ± 1.3
MR8(1.00)	HSV(0.25)	HOG(0.25)	82.5 ± 1.8	82.7 ± 2.5	82.9 ± 2.3
Sift Bdy(1.00)	HSV(0.50)	HOG(0.50)	83.1 ± 1.8	83.0 ± 1.9	83.4 ± 1.9

Table 5.8: Average recognition rates on the 17 category database for combinations of three features.The number in brackets next to each feature is the global weight. The best performance is achieved using the internal SIFT, boundary SIFT and HSV features.

Table 5.9 shows the average recognition for all combinations of 4 features. Note that only one of the four features combinations give a better performance than combining only three features. Namely, combining the two SIFT, the HSV and the HOG features, which gives a performance of $88.2 \pm 2.0\%$. There is still not much different between the weighting schemes, but the local weighting scheme is constantly slightly better. Table 5.10 shows the performance combining all features. Note that the performance using the five features is the same as using the best four features, thus the MR8 feature appears to be redundant. The global weight for the boundary SIFT and the HOG increase when using four instead of five features, which indicates that they compensate for the missing MR8 features. Furthermore,

if we look at the local weights learnt using the five features and the four features (figure 5.10), we can see that the MR8 filters are given very small weights. We can also see that the removal of the MR8 filter is largely compensated for by increasing the weights of the two SIFT features.

1st feature	2nd feature	3rd feature	4th feature	Equal β_f	Global β_f	Local β_f
Sift Int(1.00)	MR8(0.25)	Sift Bdy(1.00)	HSV(0.25)	87.2 ± 2.0	87.3 ± 1.7	87.5 ± 2.0
Sift Int(1.00)	MR8(1.25)	Sift Bdy(0.50)	HOG(1.25)	81.4 ± 2.2	80.8 ± 1.6	81.9 ± 3.1
Sift Int(1.00)	MR8(0.75)	HSV(1.50)	HOG(0.50)	87.0 ± 2.2	87.2 ± 2.7	87.8 ± 2.6
Sift Int(1.00)	Sift Bdy(1.50)	HSV(0.50)	HOG(1.00)	87.3 ± 2.1	87.5 ± 2.2	$\mathbf{88.2 \pm 2.0}$
MR8(1.00)	Sift Bdy(0.50)	HSV(0.25)	HOG(0.75)	84.4 ± 1.6	85.1 ± 2.2	85.4 ± 2.4

Table 5.9: Average recognition rates on the 17 category database for combinations of four features.

Sift Int	MR8	Sift Bdy	HSV	HOG	Equal β_f	Global β_f	Local β_f
(1.00)	(0.50)	(0.50)	(0.75)	(0.75)	87.4 ± 1.8	87.6 ± 1.6	$\mathbf{88.1 \pm 1.9}$

Table 5.10: Average recognition rates on the 17 category database for combinations of all five features.

If we use the best four features and the segmentation obtained by minimizing a CRF using graphcuts we obtain a performance of $83.33 \pm 1.39\%$. This shows that the segmentation scheme chosen affects the performance when combining the features as well as when using single features.

Overall the weighting scheme used does not seem to make much of a different. However, as we will show in the next section, our intuition is that there will be a larger difference for the 102 category database because of the number of categories and the overlap between categories for the different features as indicated by figures 3.9 and 3.10.

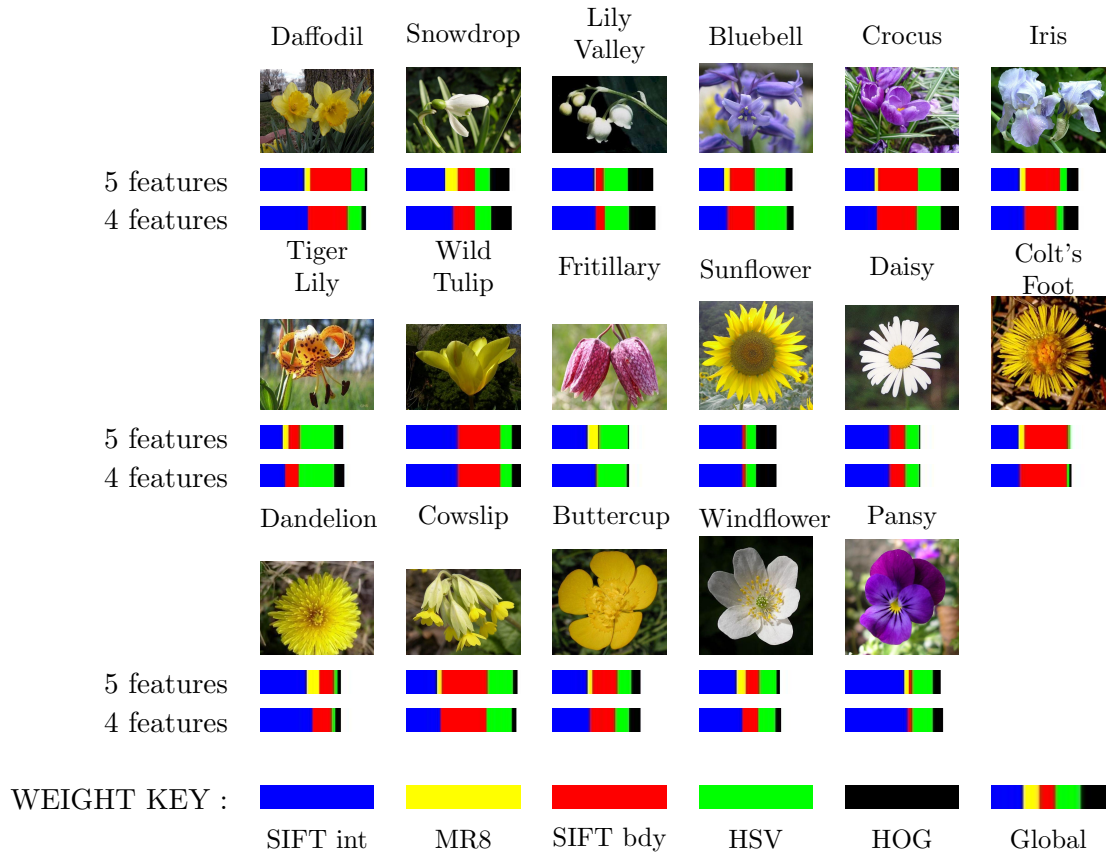


Figure 5.10: Weights for the 17 category dataset – Each image is an instance of a different class. The colour bar below each image shows the magnitude of the local feature weights learnt for each class. The top bar shows the weight learnt using 5 features and the bottom bar the weights learnt using 4 features. Note that very small weights are given to the MR8 features. The global weights are also displayed for comparison (bottom right).

5.3 Evaluation on the 102 category dataset

So far we have evaluated the performance on a small 17 category dataset. We will now evaluate how our classification pipeline performs on the 102 category dataset (section 3.2). We use our iterative segmentation scheme and the multiple kernel framework of Varma and Ray [112] together with the features used for the 17 category dataset. Because of the redundancy of the MR8 filter banks and the time it takes to compute them, we do not use them here.

5.3.1 Optimization on the validation set

We use the same feature parameters, e.g. grid spacing and patch size, as optimized on the 17 category database, but create a new vocabulary for each feature by clustering a subset of the features from the training images using k-means clustering. We then optimize the number of words for the k-means clustering for each feature on the validation set.

After optimization, the optimum number of words is 900 for the colour features, 8000 for the SIFT over the entire foreground region, 1000 for the SIFT on the boundary region only, and 1500 for the HOG features.

Once the parameters have been determined on the validation set, the classifiers are retrained using all the available training data (both the training and validation sets). At this stage the weights β_f are determined using the optimization method of Varma and Ray [112].

5.3.2 Results on the test set

Table 5.11 shows the classification performance for different feature sets. It can be seen that combining all the features results in a far better performance than using the single best features (SIFT internal). Both the internal SIFT and the boundary SIFT contribute to the performance. The internal SIFT individual performance is however much better, which is to be expected as non-rigid deformations of the petals affect the boundary more than the inside. As a comparison, results are also presented using equal weights for the kernels as explained in section 5.2. We can see that using the per-class weights for each one-vs-rest classification gives better recognition rates.

Features	Local β_f	Equal β_f
HSV	44.6%	
SIFT internal	57.5%	
SIFT boundary	34.6%	
HOG	50.9%	
HSV + SIFT int	68.2%	61.7%
HSV + SIFT bdy	59.1%	56.5%
HSV + HOG	64.9%	58.0%
SIFT int + SIFT bdy	60.2%	53.9%
SIFT int + HOG	66.5%	67.6%
SIFT bdy + HOG	56.3%	54.2%
HSV + SIFT int + HOG	72.6%	68.5%
HSV + SIFT int + SIFT bd + HOG	73.7%	69.4%

Table 5.11: Recognition rates for classification on the 102 category test set. Results are presented using the multiple kernel framework of Varma and Ray [112] and using equal weights for the kernels as explained in section 5.2. It can be seen that combining the features within the kernel framework improves the performance.

Figures 5.14 and 5.15 show the weights learnt for each one-vs-rest classifier. Each class is represented by one image and the bar below each image show the distribution of weights for the different features. The bar results show that overall the internal SIFT features are the most discriminative, and have the largest weight for most classifications. However, there are some classes, for example Cautleya Spicata (figure 5.11), that have zero weights on the internal SIFT. These have much more weight put on both the HOG features and the SIFT sampled along the boundary, i.e. they are much better distinguished by the boundary of the petals than the internal texture. It also shows that some classes like Snapdragon (figure 5.12), are not well

Figure 5.11: Cautleya Spicata – examples of images in the dataset.

distinguished by colour, but the orange Dahlia (figure 5.13) has a relatively large colour weight. This is because the Snapdragon occurs in many different colours, whereas the orange Dahlia only occurs in one colour.

Figure 5.12: Snapdragon – examples of images in the dataset.

Figure 5.13: Orange Dahlia – examples of images in the dataset.

Figure 5.16 shows examples of images that are classified correctly by the combination of features, but would be misclassified by some of the features independently. Each row shows the classification for an image (green indicates correct). The last row, for example, shows a case where no single features are classified correctly but the combination of all three results in a correct classification. Note also that sometimes the single features classify the image correctly but the combination does not. Figure 5.17 shows examples of typical misclassifications. The closest image shown is a random sample from that class. Note that the misclassified categories are very similar. We will return to this issue in the discussion of the next chapter.

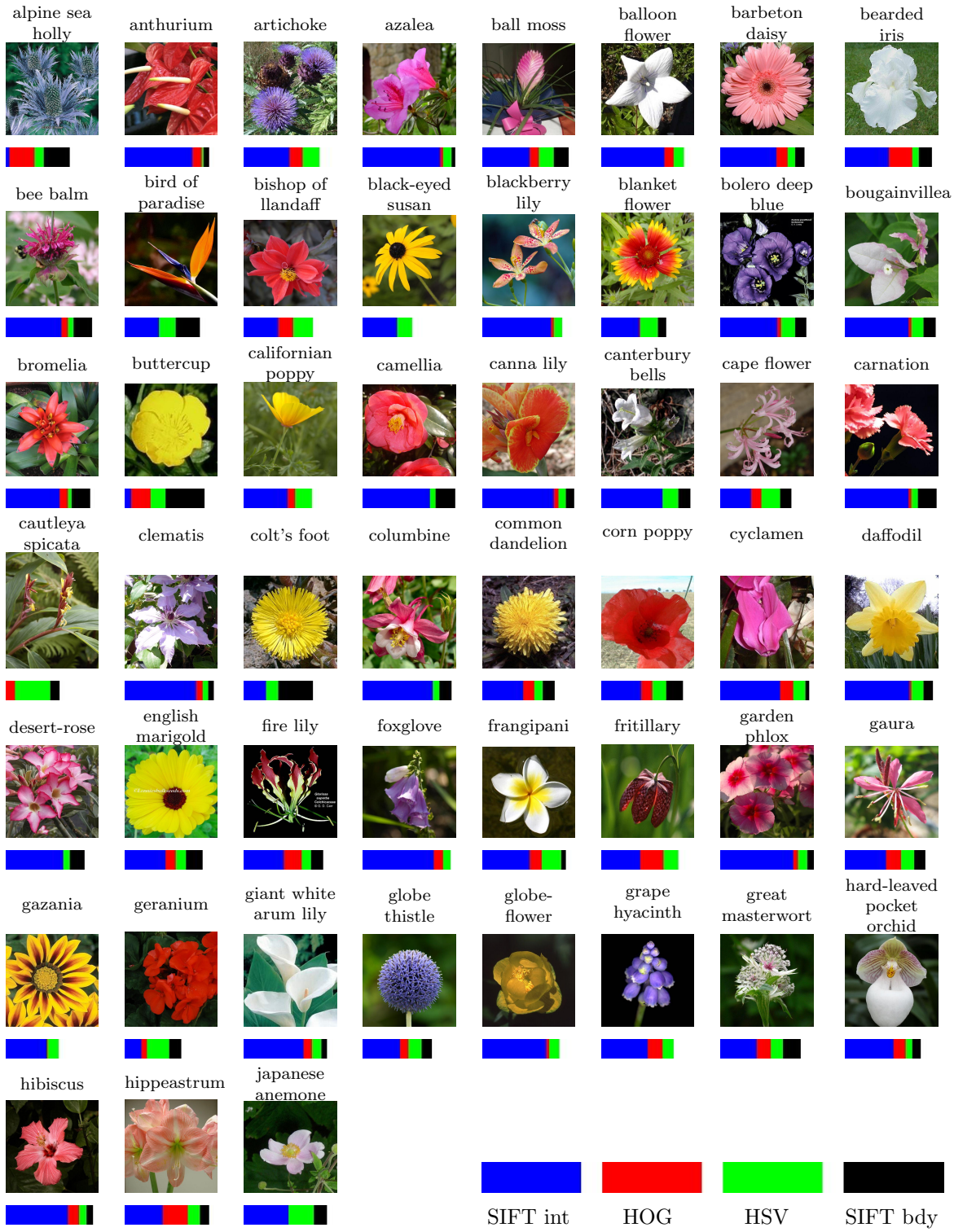


Figure 5.14: Weights for the 102 class flower dataset – Part 1. Each image is an instance of a different class. They are sorted alphabetically. The colour bar below each image shows the magnitude of the feature weights learnt for each class. Blue represents the weight for the internal SIFT, red for the HOG features, green for the colour features and black for the boundary SIFT.

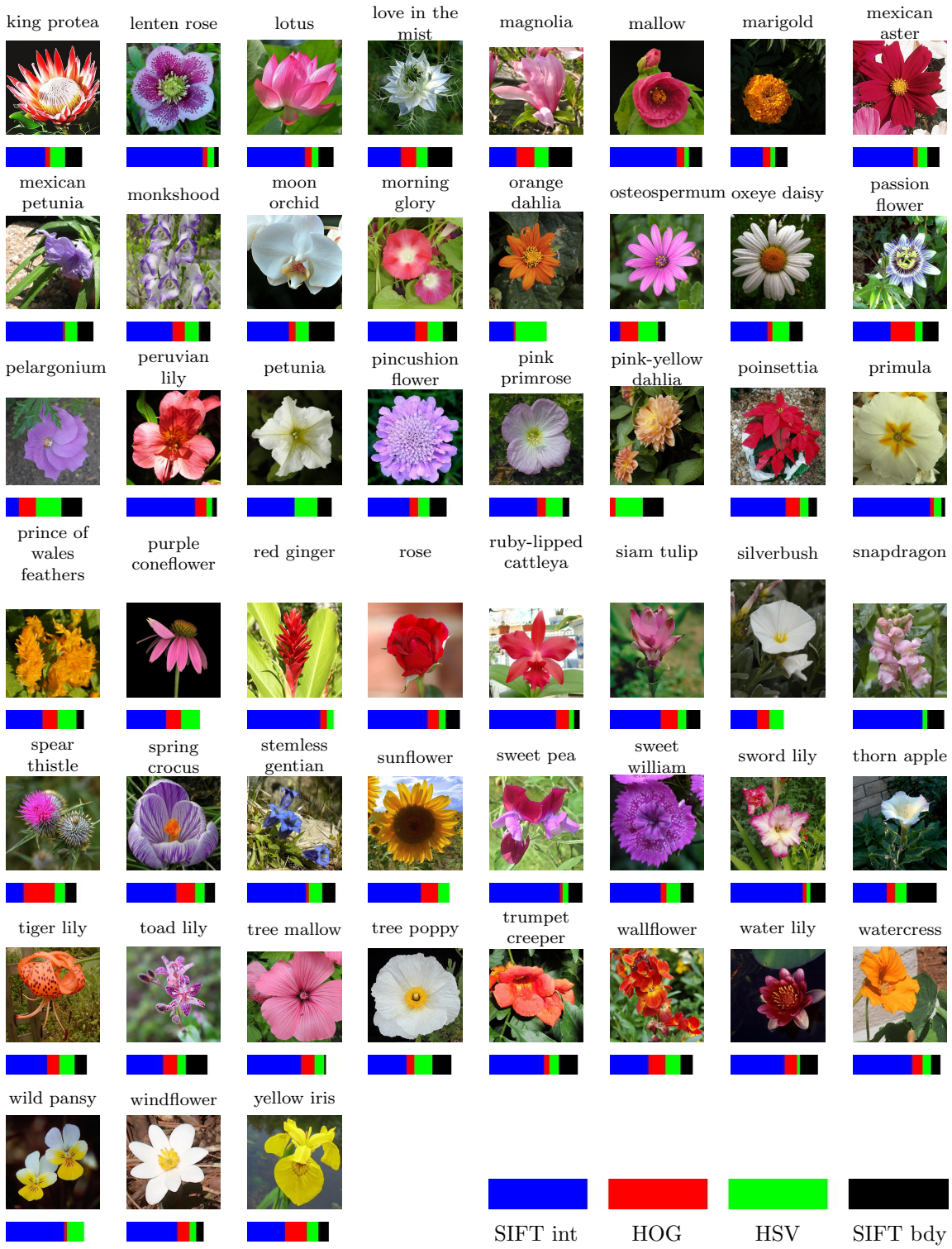


Figure 5.15: Weights for the 103 class flower dataset - Part 2. Each image is an instance of a different class. They are sorted alphabetically. The colour bar below each image shows the magnitude of the feature weights learnt for each class. Blue represents the weight for the internal SIFT, red for the HOG features, green for the colour features and black for the boundary SIFT.

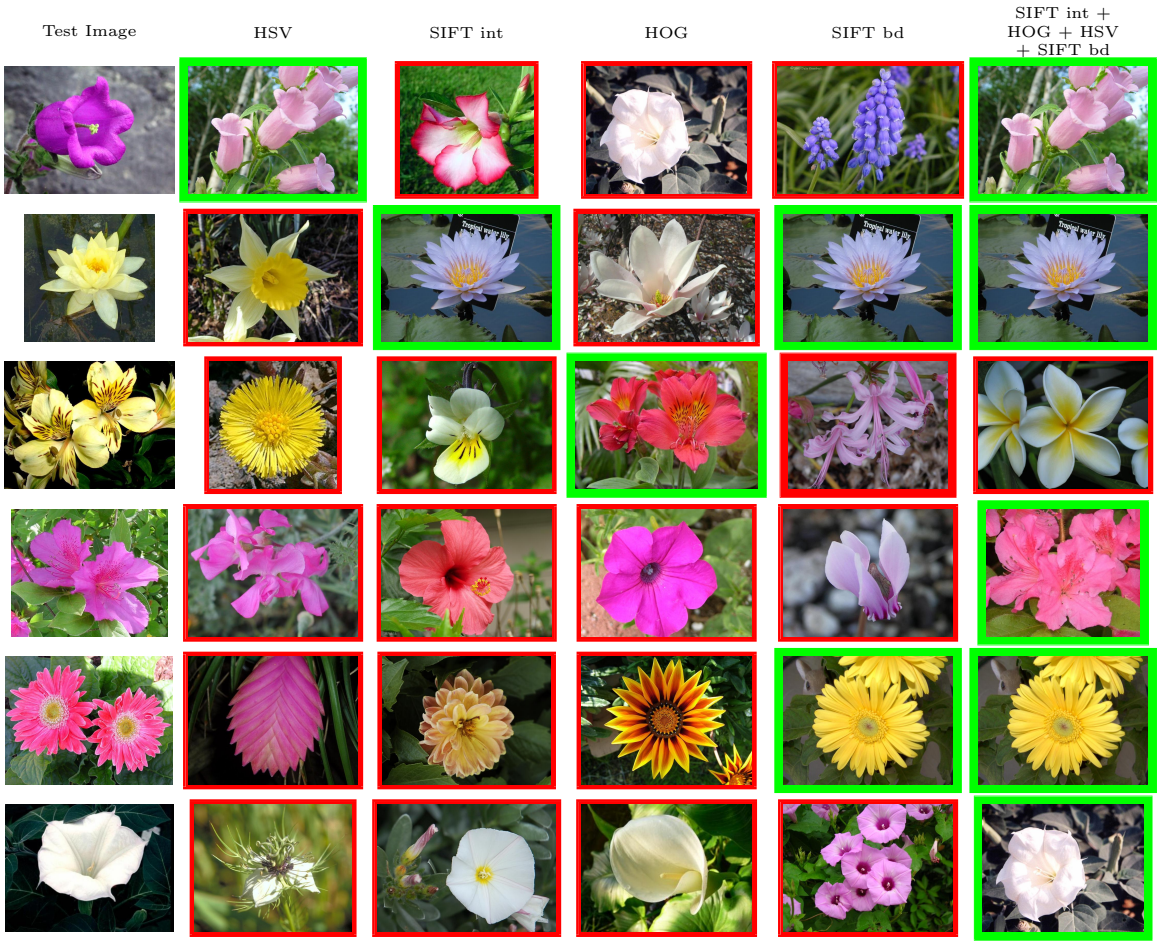


Figure 5.16: Example classifications. Each row shows the classifications for each feature given a test image(left column). The classification is indicated by a random image from that class (i.e. not the "closest" image to the test image). A green/thick border indicates correct classification and a red/thin border incorrect classification. The first row, for example, shows an image where colour is the only single feature that classifies it correctly, but the combination of feature still recognizes it. The third row shows an example of when one of features classifies the image correctly, but the combination of feature does not. The last row shows an example where none of the features gets it right, but the combination of all features still gets it right. Note how each feature captures the aspect of the flower it was designed for, such as local shape with SIFT and colour with HSV.

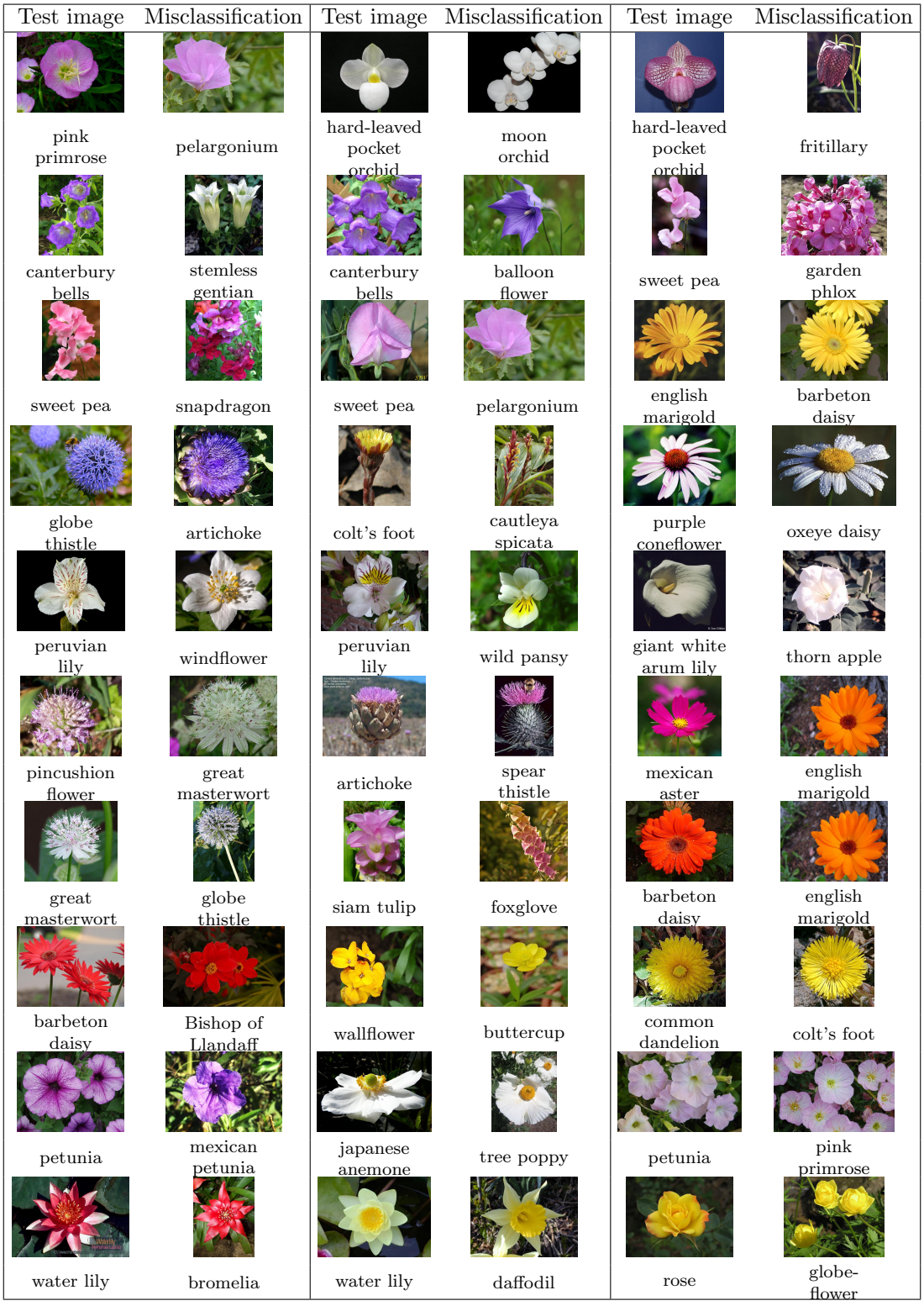


Figure 5.17: Examples of misclassifications. Note, the image shown for the misclassification is only a representative image for that class and not the "closest" image.

The histogram in figure 5.20 shows an overview of the per class recognition rates. We can see that for more than half of the categories we achieve a recognition rate above 80%, but there are also approximately 20 categories for which the recognition rate is below 50%. Figures 5.21 and 5.22 show a more detailed view of the per category recognition rate in alphabetical order. Very good performance is obtained for many of the Asters, for example, the orange Dahlias (figure 5.13), Bishop of Llandalf, Gazania, Osteospermum, Oxeye Daisy and Sunflower. These are all symmetric and flat flowers with many petals originating from the centre (figure 5.18).

Figure 5.18: Some Asters – examples of categories for which good classification performance is achieved.

We also perform well for Cautleya Spicata (figure 5.11), Alpine Sea Hollies, Birds of Paradise, Geranium, Pincushion flowers and Toad Lilies, which all have an atypical flower shape (see figure 5.19), but are either very distinctive or do not exhibit much variation across the dataset.

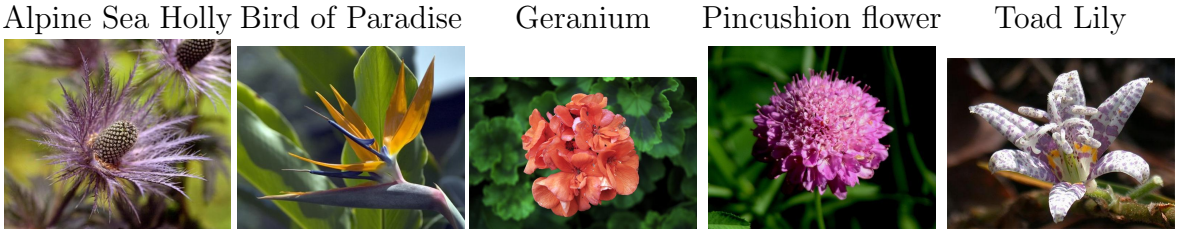


Figure 5.19: Example of categories for which good classification performance is achieved.

The mistakes the system makes are highlighted in figures 5.23 and 5.24. They are ordered from worst to better in term of classification performance. The figures show

a random example of a misclassified images from the categories with a recognition rate of less than 40% together with all the training images for those categories. In general, the system performs badly for categories which exhibit large variations in the training set, but still do not cover the distribution of the dataset. For example, the Mallow does not have a single image in the training set with the same pose or colour, but still the test image has a different colour or poses. An exception is the Camelia, for which the training set does not have much variation in pose, and yet the test image has a completely different pose. In addition, more mistakes are made for categories with many small flowers, such as, Snapdragons and Sweet Peas. Segmentation failures account for 9.5% of the misclassifications.

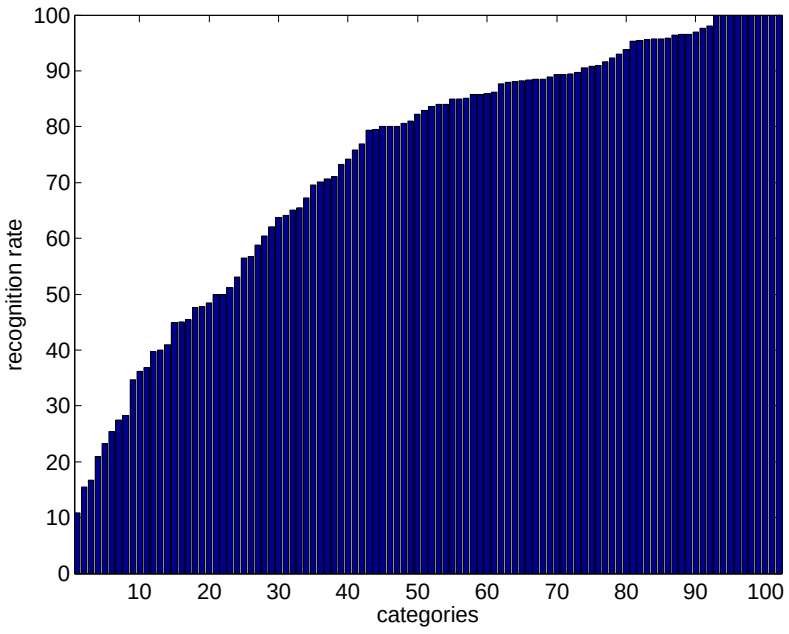


Figure 5.20: Histogram of per-class recognition rates ordered by recognition rate.

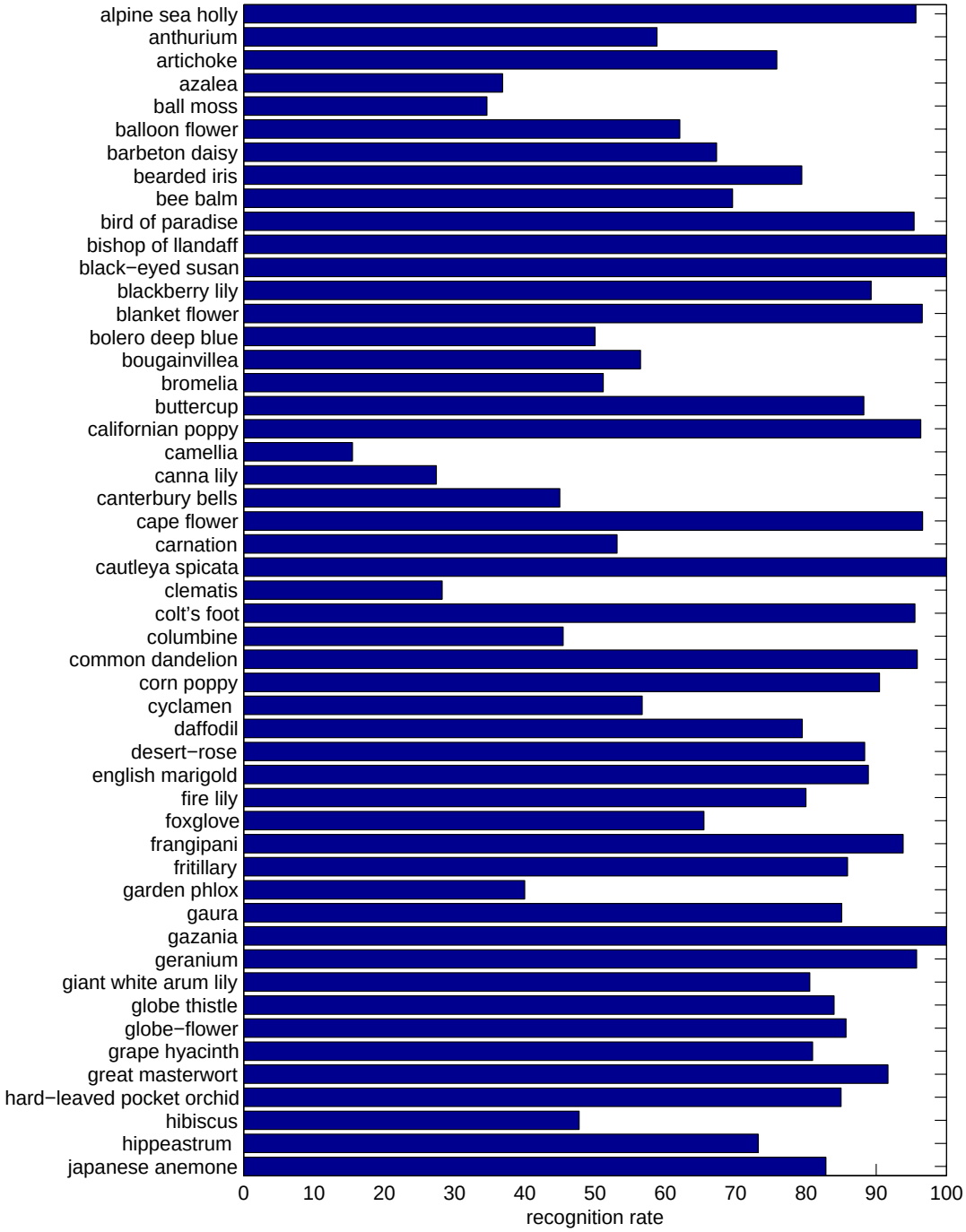


Figure 5.21: Histogram of per-class recognition rates with class labels – Part 1.

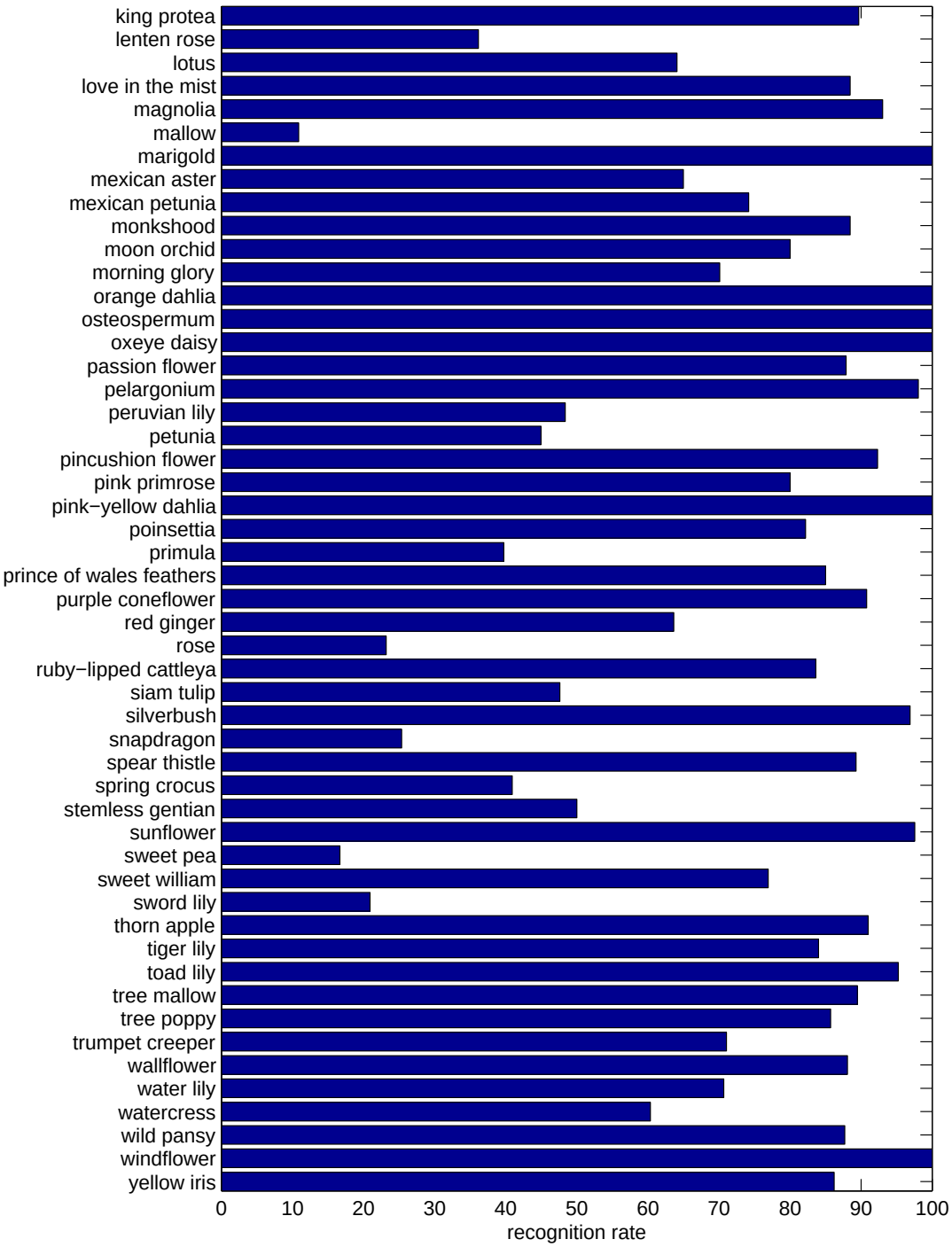


Figure 5.22: Histogram of per-class recognition rates with class labels – Part 2.

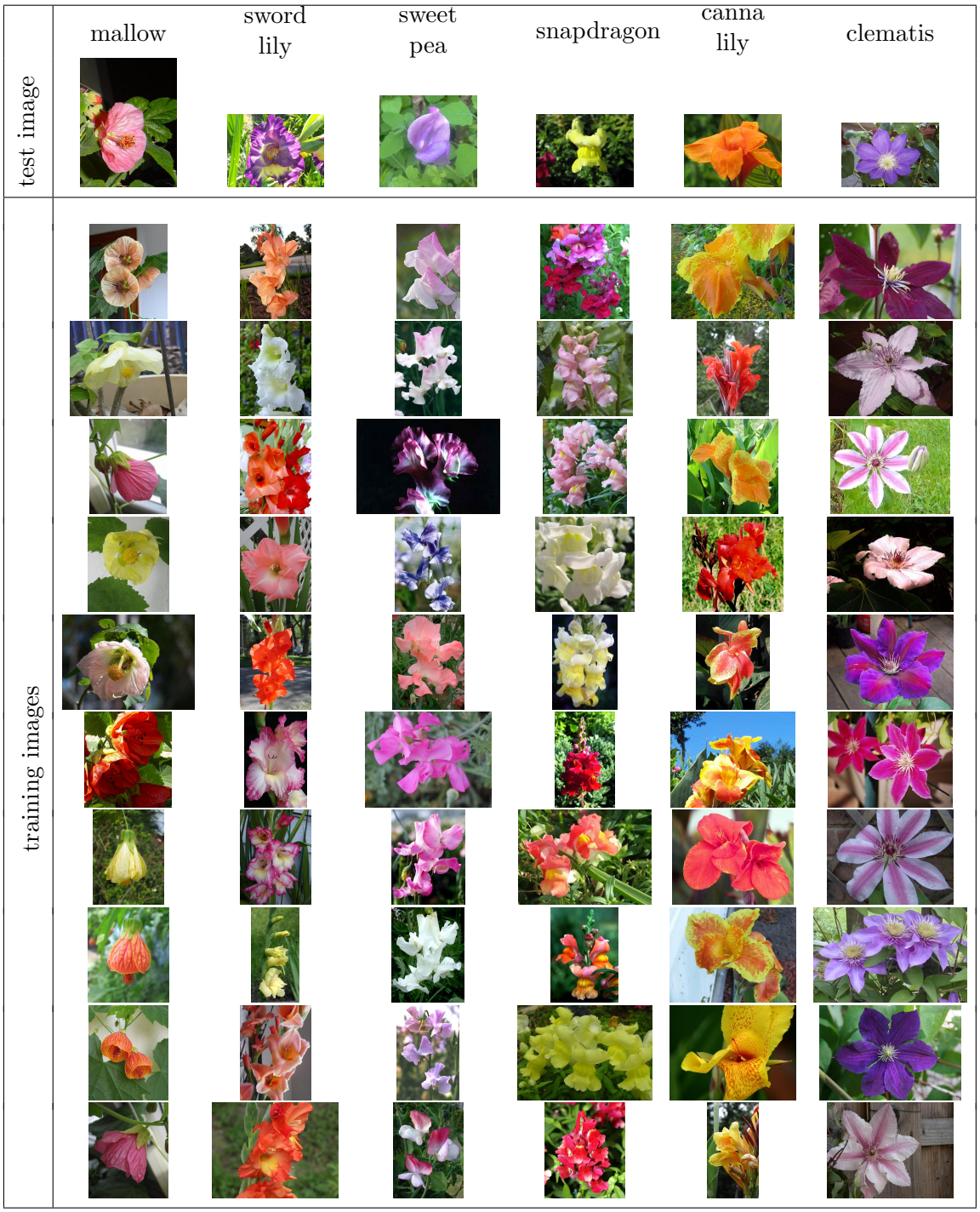


Figure 5.23: Categories with less than 40% recognition rate – Part 1. Each column shows a misclassified test image and the training images for that class.

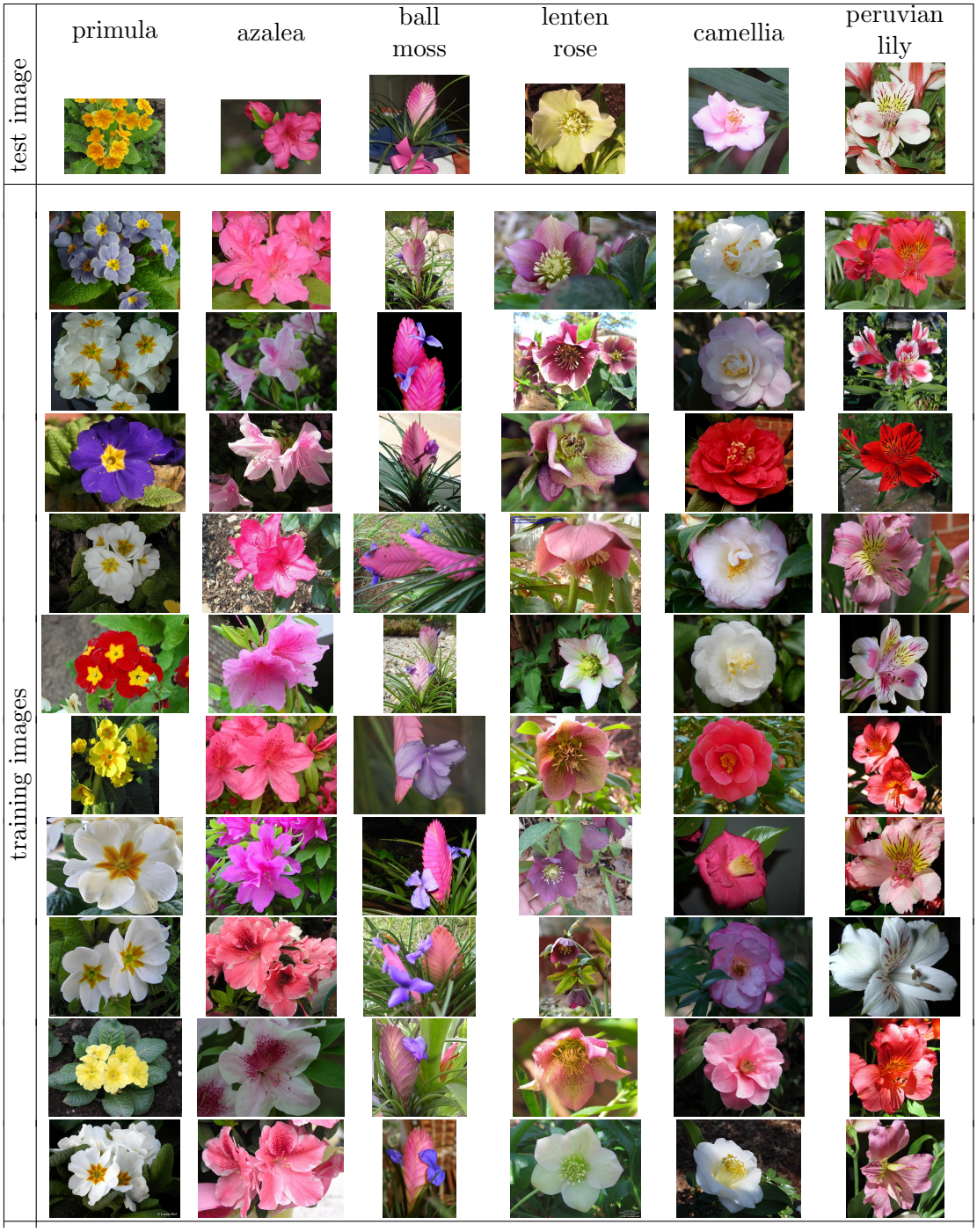


Figure 5.24: Categories with less than 40% recognition rate – Part 2. Each column shows a misclassified test image and the training images for that class.

5.3.3 Comparison with other methods

We compare the bag-of-words approach described in section 5.1 to one using the spatial pyramid match kernel of Lazebnik *et al.* [54]. We use a three-level pyramid and compute this kernel for all features. The spatial grid is applied to the segmented region. Once the pyramid match kernel has been computed for each feature, we use the same MKL learning as in our classification scheme to determine the optimum feature weights. Table 5.12 shows the recognition for both approaches. There is not much difference in the performance between the two methods. The HOG features, which are the only features that are not rotation invariant, show a decreased recognition rate for the method of Lazebnik *et al.*[54]. Overall, the performance using the method of [54] is slightly worse. In addition, it is worth noting that the spatial bag-of-words vector used for a three-level pyramid kernel is 21 times larger than the standard bag-of-words vector and therefore computing the kernel takes much longer.

Features	Bag-of-words	Lazebnik *et al.* [54]
HSV	44.6%	42.9%
SIFT internal	57.5%	56.3%
SIFT boundary	34.6%	35.9%
HOG	50.9%	43.9 %
HSV + SIFT int + SIFT bd + HOG	73.7%	70.1%

Table 5.12: Recognition rates for classification on the 102 category test set. Results are presented using the bag-of-words representation 5.1 and the spatial pyramid match kernel of Lazebnik *et al.* [54]. The weights for the feature combination are learnt using the multiple kernel framework of Varma and Ray [112] in both cases. It can be seen that the bag-of-words approach achieves a higher recognition rate for most features.

5.4 Discussion

We have shown that tuning the features and combining features for several different aspects results in a significant performance boost, with the final classifier having superior performance to each of the individual ones. Using an optimized kernel framework we can improve the classification performance on a large dataset of very similar classes. The learning of different weights for different classes enables us to use an optimum feature combination for each classification. This allows us to incorporate, for example, that some classes are very similar in shape but different in colour and that some classes are better distinguished by the overall shape than the internal shape and vice versa.

Some categories have a significant overlap, which make them difficult to separate. We will expand this issue further in the discussion of the next chapter. However the biggest difficulty arises from categories that have too much variability to be able to cover the whole distribution of the category in the limited number of training samples. We need to address this issue in order to improve classification. In the next chapter, we will explore normalizing the flower region, in order to remove rotation and scale changes. Thus, removing some of the variance in the dataset.

Chapter 6

Geometric layout features

In chapter 4 we introduced a geometric model in order to improve segmentation. In the previous chapter we explored combining features that do not explicitly take into account this geometry. We will now look at using the geometry detected in the segmentation stage to develop geometric layout features for classification. The purpose of these features is two-fold: *i*) to remove intra-class variation that arises from viewpoint variations, and *ii*) extract features that are more discriminative. For *i*) we will present a three step model, where the first step is normalizing under an affine transformation, the second is rotationally aligning the petals, and the third is extracting the descriptors from the transformed images. For *ii*) we propose two layout descriptors: A circular 'colour layout' descriptor and a circular 'histogram of gradient' descriptor. We evaluate the descriptors on the 102 category dataset (section 3.2). We also evaluate how the system generalizes to new unseen images.

6.1 Normalizing the flower

The aim is to obtain a description of a flower which is invariant to flower rotation and scale. The flowers in figure 6.1, for example, would appear to have very different distribution of colours, edges, boundaries etc. due to their relative rotation and scale. We want to normalize the flowers to a canonical frame where they can be described in the same way. This is done in two steps: *i*) obtaining invariance to out-of-plane rotation *ii*) obtaining invariance to in-plane rotation.

Figure 6.1: Example of flowers from the same species with different appearance due to viewpoint variations.

Figure 6.2 shows the scheme. We begin by segmenting the flower using the iterative segmentation scheme (chapter 4). Recall that the foreground region is detected by finding potential petals and having these vote for a centre. We take the segmented region corresponding to the majority vote and fit an ellipse to this region (figure 6.2(b)).

The ellipse is fitted using the method proposed by Fitzgibbon and Fischer [37]. Given a set of points $P = \{\boldsymbol{x_i}\}_{i=1}^{N}$, where $\boldsymbol{x}_i = (x_i, y_i)^T$, we want to find the ellipse $F(\boldsymbol{a}, \boldsymbol{x}) = \boldsymbol{a} \cdot \boldsymbol{x} = ax^2 + bxy + cy^2 + dx + ey + f = 0$, where $\boldsymbol{a} = [\ a \quad b \quad c \quad d \quad e \quad f\ ]$

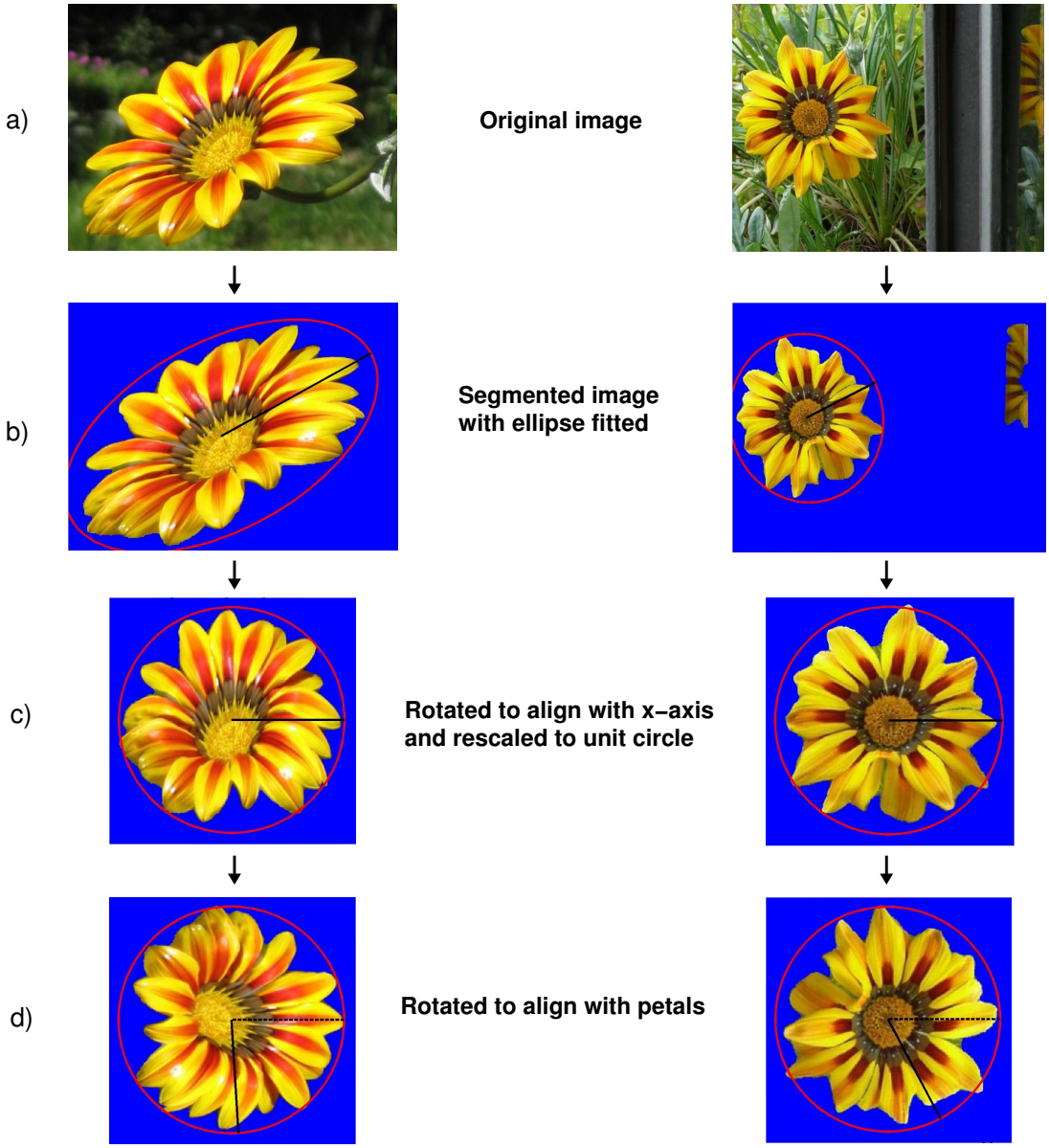


Figure 6.2: Flower normalization scheme – a) Original image. b) An ellipse is fitted to the convex hull of the segmented region. c) The ellipse is aligned with the image axis and transformed to a unit circle. c) The image is rotated to align with the petals

and $\boldsymbol{x} = [\ x^2 \quad xy \quad y^2 \quad x \quad y \quad 1\ ]$. This is done by minimizing the sum of squared algebraic distances given by the objective function

$$E = \sum_{i=1}^{N} F(\boldsymbol{a}; \boldsymbol{x_i})^2 = ||\boldsymbol{D}\boldsymbol{a}||^2. \tag{6.1}$$

In order to avoid the trivial solution and enforce that the conic is an ellipse we minimize this function subject to the constraint $4ac - b^2 = 1$, which Bookstein [9] showed can be solved as a rank-deficient generalized eigensystem. We use the numerically more stable implementation of [43]. Five points are needed to fit the ellipse. One could consider using the petal tips which we have already detected, although this would often give us less than five points and in addition the ellipse fitting could be misaligned if all petals are not detected. Instead we fit the ellipse to the convex hull of the segmented region. Once the ellipse has been fitted, it is transformed to a circle to remove stretch and skew (figure 6.2(c)).

Now to remove in-plane rotation the circle is rotated to align with a detected petal tip (figure 6.2(d)). This scheme generalizes the flower to an elliptical shape, but as can be seen in the examples of figure 6.3-6.4 it also normalizes flowers which are not naturally elliptical but have some symmetry. However, we do not expect it to work for many small flowers or flowers which are very non-planar and asymmetrical. Note that when there are multiple non-overlapping flowers, the ellipse will be fitted to one of the flowers.

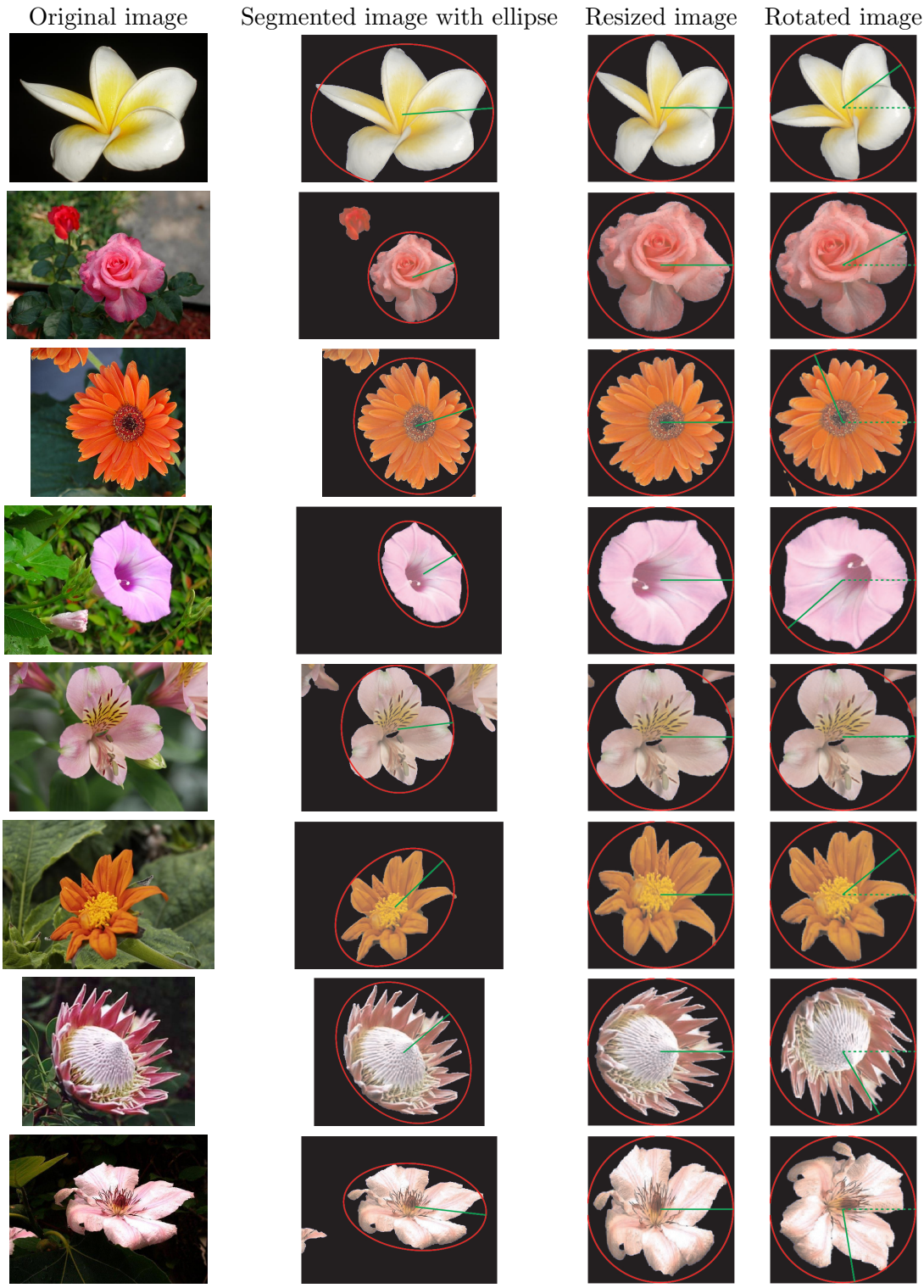


Figure 6.3: Normalized images – Part 1. Each row shows the normalization of an image : the original image, the segmented images with the ellipse fitted, the resized image and the rotationally aligned image. The green line indicates the rotation of the ellipse. Images are from various categories.

Figure 6.4: Normalized images – Part 2. Each row shows the normalization of an image : the original image, the segmented images with the ellipse fitted, the resized image and the rotationally aligned image. The green line indicates the rotation of the ellipse. Images are from various categories. These images were misclassified in chapter 5

6.2 Configuration description and alignment

Now that the flower has been normalized to a canonical frame the next step is to describe the configuration. Belongie and Malik's Shape-Context descriptor [6], models boundary shape using a circular log-polar histogram. Dalal and Triggs [26] also propose using log-polar histograms for the HOG features. The GLOH descriptor [77] is similar to the Shape-Context descriptor but is uses gradients instead and is an extension of the SIFT descriptor to a log-polar grid. Recently, the DAISY descriptor [106] extends this work further and uses summation of Gaussian smoothed gradient orientations from a series of concentric circles.

Here we look at two different geometric configurations: one with a single central cell and multiple angular bins in the outer cells, and one with a divided centre cell (see figure 6.5). These are similar to the circular geometries proposed in [6, 26]. We use two radial bins: the outer radius is determined by the unit circle and the inner radius is determined by the petal intersections, which were detected in the segmentation stage (figure 4.3). We set the inner radius to the mean of the detected petal intersections.

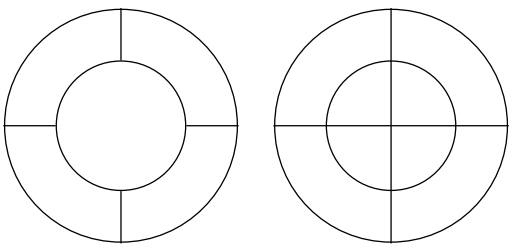


Figure 6.5: Two different geometric layouts

Given two flowers, we need to align their respective configurations. As mentioned in the previous section we rotate the flower to align with a petal. If the flower is symmetric, any arbitrary choice of petal should give a good alignment. However, since some flowers are not symmetric, we will generate as many rotations as detected petals and when matching the configuration from two images we will find the rotation which gives the most similar descriptor.

6.3 Colour Layout

It is not just the colour of the flower that makes it distinguishable but also the *arrangement* of the colour. Figure 6.6 shows an example of some flowers with similar colours but that have quite different colour distributions. The objective is to introduce some notion of colour layout to our bag-of words vector. We take inspiration from methods that compute bag-of-words for different rectangular grid cells [12, 54] and extend it to circular cells. The HSV values from each cells are clustered using the same vocabulary as for the global bag-of-words vector.

Figure 6.6: Example of images with similar colour but different layout.

The two geometries proposed in section 6.2 are evaluated on the validation set of the 102 category dataset using 1, 4, 8 and 16 angular cells. Figure 6.7 shows how the recognition rate varies with the layout. It can be seen that it is important to have a single centre cell, but the number of angular cells is less important. The best performance of 44.2% is obtained using 8 angular cells and one central cell. In [54] the similarity of histograms at different grid resolutions are explicitly weighted so that matches in coarser grid cells will be given less weight. Here, if we combine the global bag-of-words level with the colour layout descriptor, it can be seen as a two level histogram. We do not explicitly weight the relative importance of these levels, but in the multiple kernel learning framework this will be learnt. Results on the test set are presented in section 6.5.

6.4 Gradient layout

The Histogram of Gradients descriptor (HOG, [26]) uses square histogram cells that are concatenated, thus preserving the layout of the object. This is suitable for pedestrian detection, but perhaps not for objects like flowers. Here, we will look at extending the HOG features to a circular grid. Figure 6.8 shows the HOGs computed for the circular cells. For this particular example (i.e. Passion flower) we can see that the inner cells have strong gradients in the radial direction and the outer bins have very small gradients.

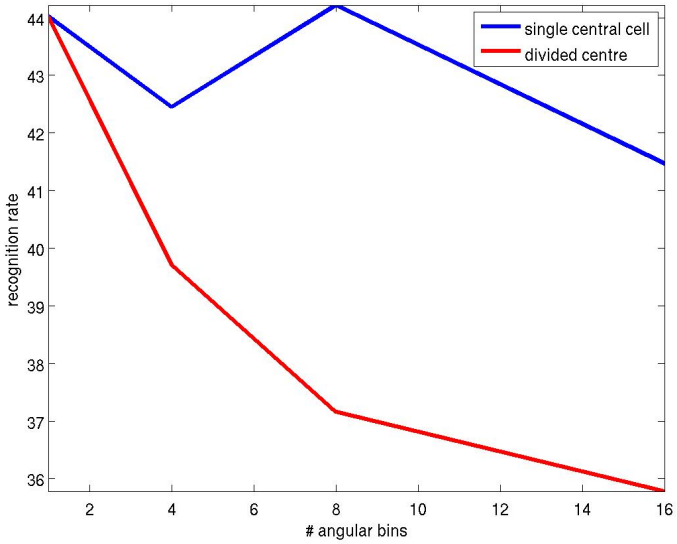


Figure 6.7: Geometry evaluation: Both lines show how the recognition rate varies with the number of angular bins. The blue line shows the performance using one single central cell, and the red line shows how the performance varies with a divided centre.

(a) Original image (b) Segmented image with ellipse (c) Transformed image with HOGs

Figure 6.8: Circular HOGs fitted to flower region – The figure shows a) Original image, b) Segmented flower with ellipse fitted to it and c) the transformed image with HOG features plotted in red for each cell. The magnitude of each orientation bin is proportional to the length of the line. The blue lines mark the angular bins and the green lines the radial bins.

In contrast to the colour layout descriptor, the performance of the circular HOGs does not vary much with the geometry. The performance variation is within 2%. A slight edge is given using a divided central bin and 32 angular bins. Figure 6.9-6.10 show the circular HOG fitted to the normalized images in figure 6.3-6.4. The classification results on the test sets are shown in section 6.5.

6.5 Results

In this section we present classification results on the test set for 102 category dataset. The results are presented for each of the geometric features and the combination with previous features. For simplicity we will refer to the colour layout features as CLAY and the HOG layout features as HLAY. We follow the same classification scheme as in chapter 5. A summary of the recognition rates is shown in table 6.1. The best previously obtained recognition rate is 73.7%, which was obtained using SIFT internal, SIFT boundary, HSV and HOG.

Colour Layout: The CLAY descriptor obtains a recognition rate of 52.7%. This is better than all other individual features except the internal SIFT features. However, when combining the features the results only improve by 1.0% (table 6.1). Note that the same (or even better) performance is achieved by removing the HSV features, 75.0% compared to the 74.7% obtained when including the HSV features. This indicates that the CLAY descriptor has made the HSV feature redundant. Figures

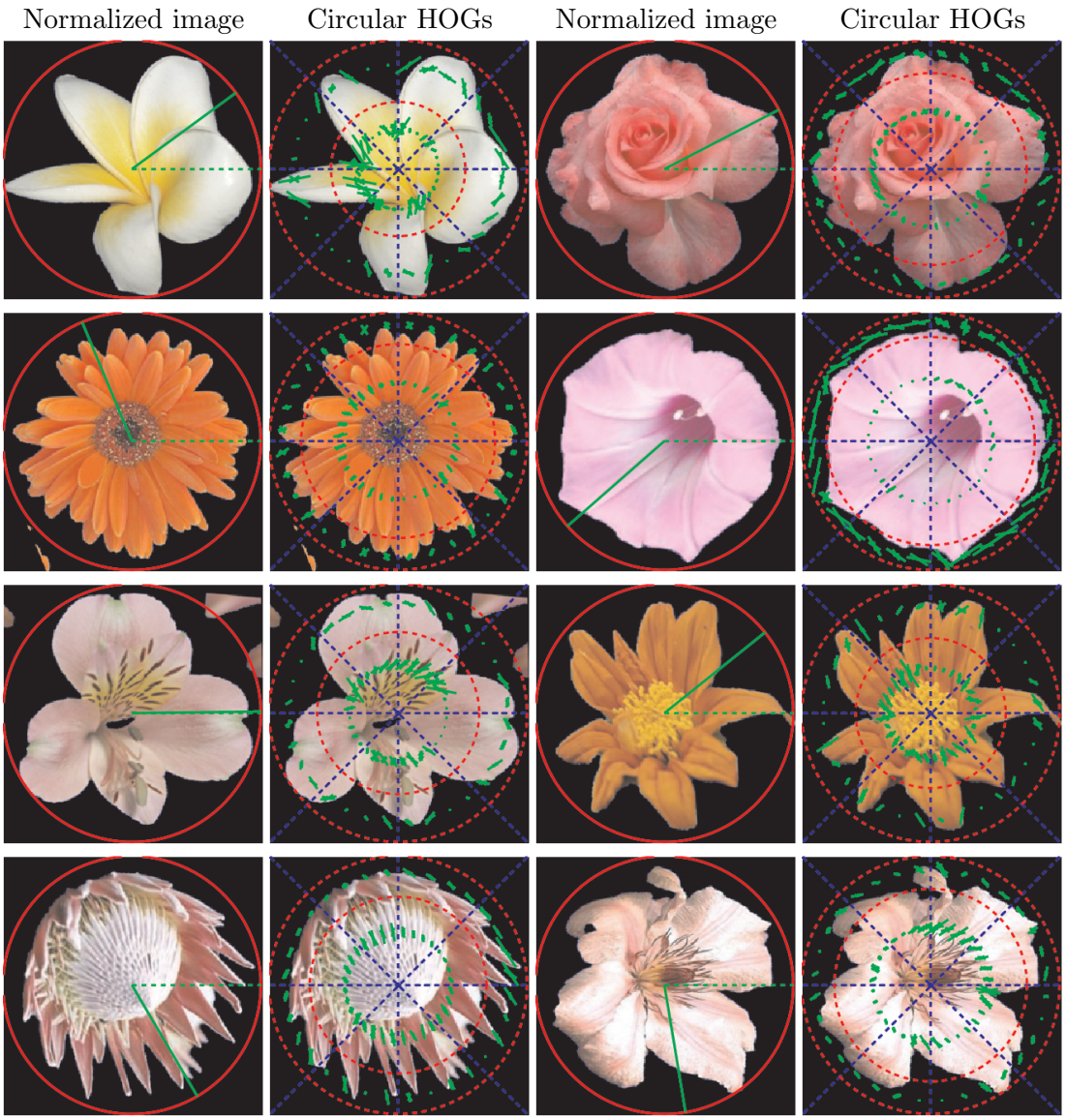


Figure 6.9: Normalized images and Circular HOGs – Part 1. The HOG features are shown in green for each cell, the radial bins in red and the angular bins in blue. For readability only 8 angular bins are shown. The original images can be seen in figure 6.3.

6.11 and 6.12 show the weights learned for each class. Note that the HSV weight is zero for most categories. Removing the HSV features causes the weight for the CLAY features to increase and, as shown in table 6.1, without loss of generalization.

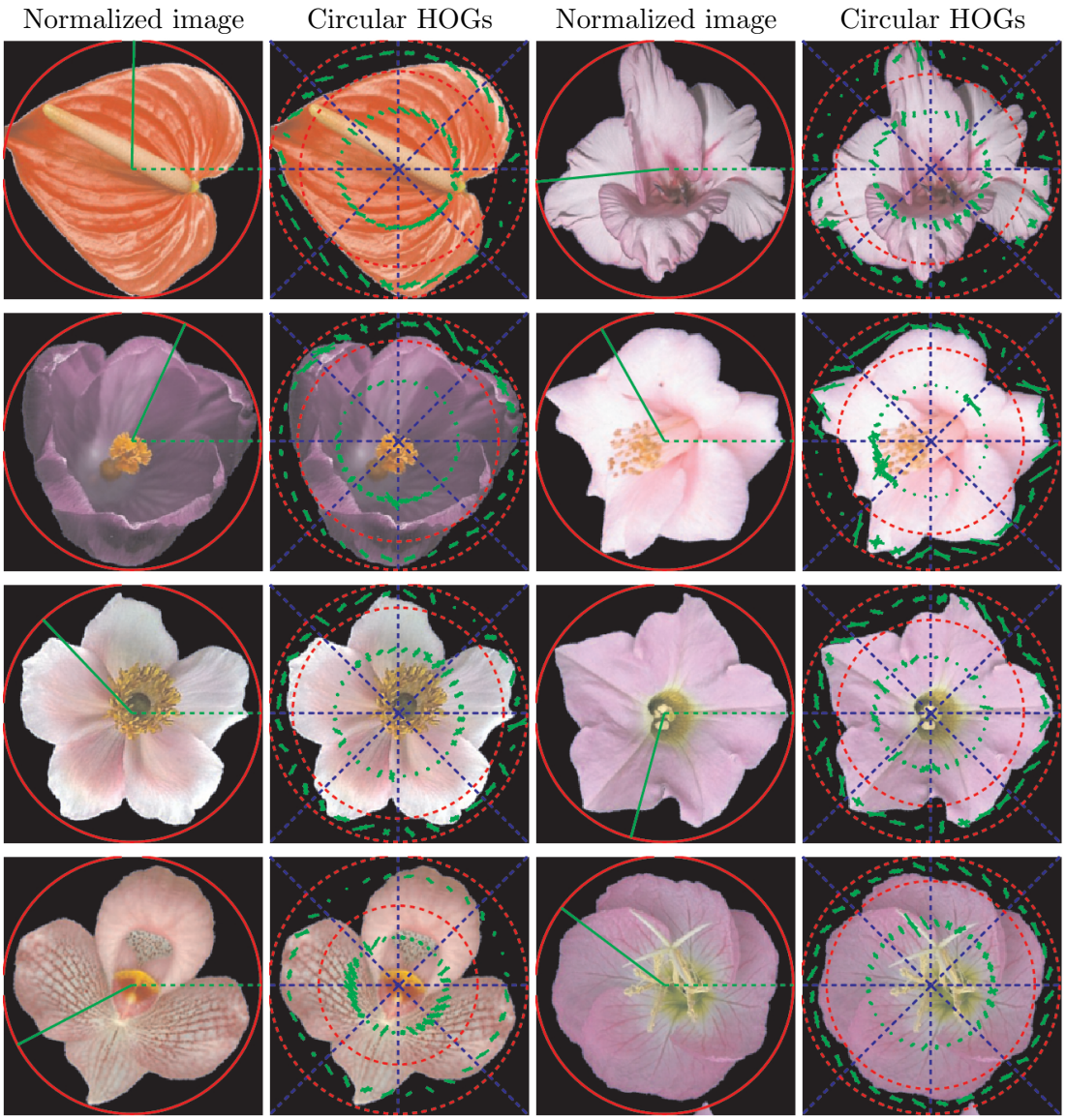


Figure 6.10: Normalized images and Circular HOGs – Part 2. The HOG features are shown in green for each cell, the radial bins in red and the angular bins in blue. For readability only 8 angular bins are shown. The original images can be seen in figure 6.4.

Circular HOGs: Unlike the CLAY features the circular HOG features by themselves give worse performance than any of the previous features and only achieves a recognition rate of 25.5%. However, they give more of an increase in performance

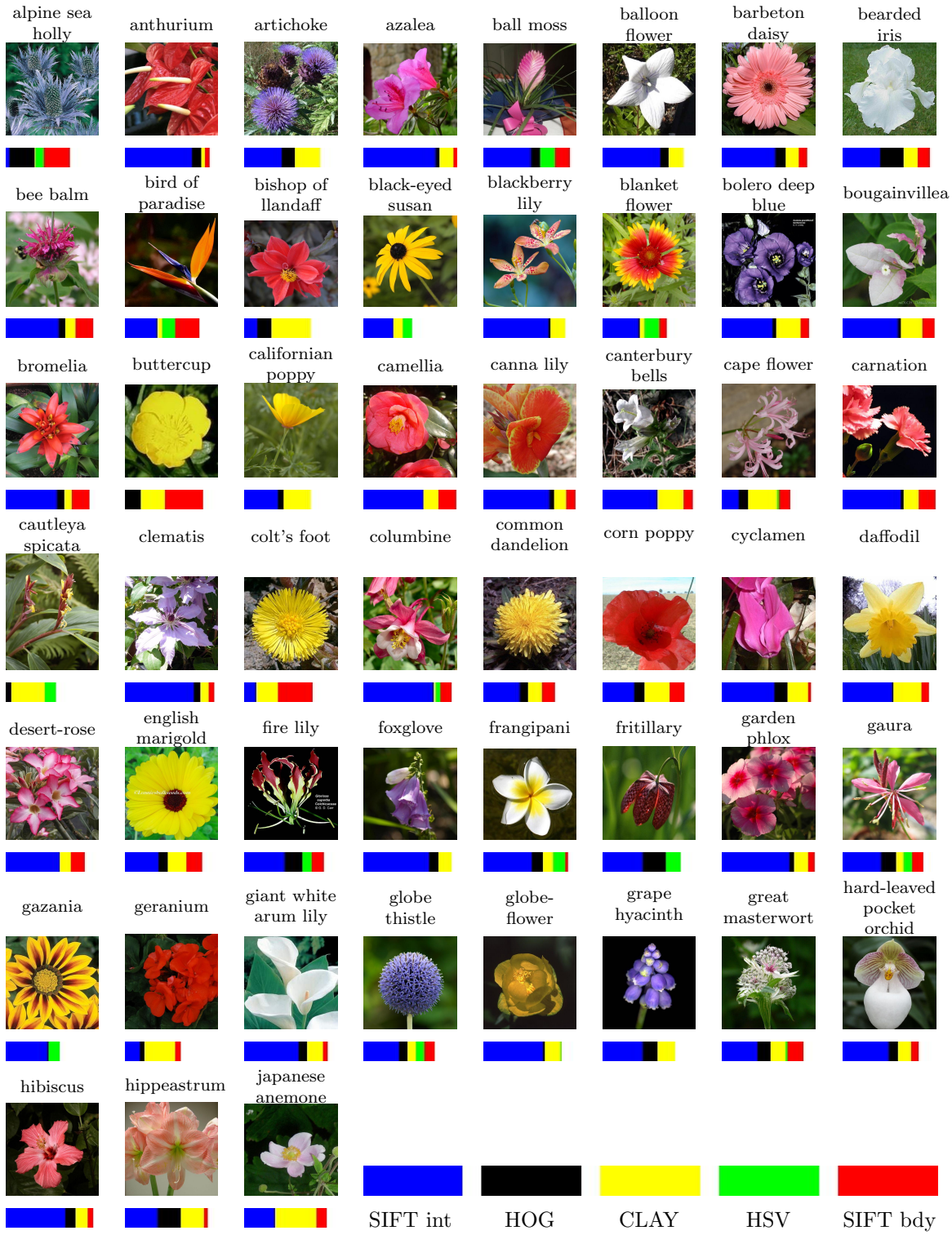


Figure 6.11: Weights for the 102 class flower dataset – Part 1. Each image is an instance of a different class. They are sorted alphabetically. The bars show the weights for the internal SIFTs, boundary SIFTs, HOGs, CLAYs and HSVs.

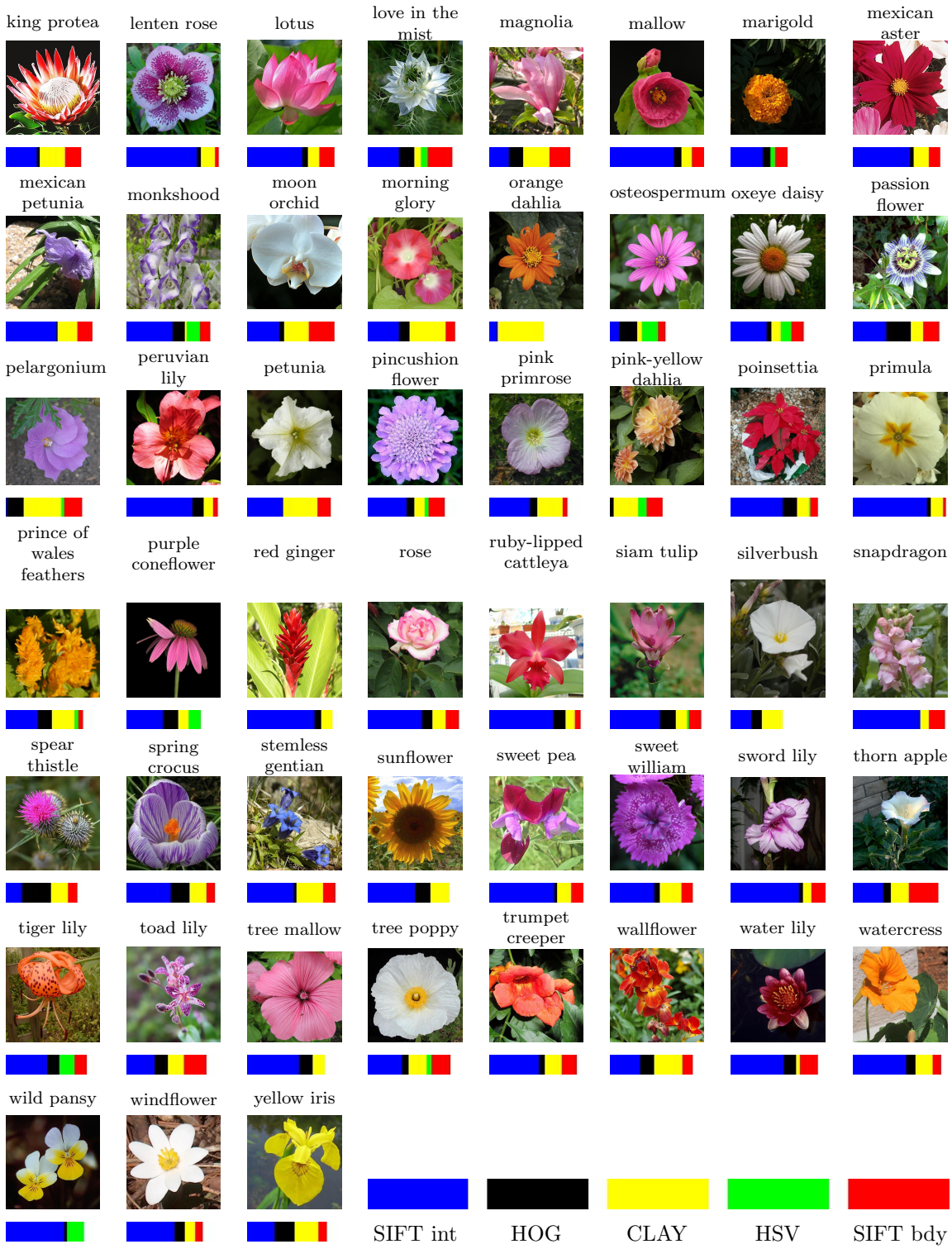


Figure 6.12: Weights for the 102 class flower dataset – Part 2. Each image is an instance of a different class. They are sorted alphabetically.The bars show the weights for the internal SIFTs, boundary SIFTs, HOGs, CLAYs and HSVs.

Features	Local β_f
HLAY	25.5%
CLAY	52.7 %
HSV	44.6%
SIFT internal	57.5%
SIFT boundary	34.6%
HOG	50.9%
HSV + SIFT int + SIFT bd + HOG	73.7%
SIFT int + SIFT bd + HOG + CLAY	75.0 %
HSV + SIFT int + SIFT bd + HOG + CLAY	74.7%
HSV + SIFT int + SIFT bd + HOG + HLAY	75.3 %
SIFT int + SIFT bd + HOG + CLAY + HLAY	76.3 %

Table 6.1: Recognition rates for classification on the 102 category test set. Results are presented using the multiple kernel framework of Varma and Ray [112].

than the CLAY features when combined with the remaining features. The recognition rate obtained is then 75.3%. To boost performance further we replace the HSV features with the CLAY features. The overall recognition rate obtained is then 76.3%. Figures 6.14 and 6.15 show the weights learnt for these features. Note that the weight for the HLAY features are in general quite small, but certain classes like Daffodils and Giant Arum Lilies have relative large weights for these features (see figure 6.13). The HLAY features perform particularly poorly for categories of many small flowers, such as Sweet Peas and Snapdragons.

Figure 6.17(a) shows how the performance is improved over all categories. The bars for each feature combination are ordered by recognition rate, hence the bars for different feature combinations do not necessarily correspond to the same categories. The graph shows that no category has a lower recognition rate than we had before.

Figure 6.13: Categories with large weights for the HLAY features. The HOG features are plotted in green.

Figure 6.17(b) plots the recognition rates of the new feature combination versus the feature combination from chapter 5. We can see that for most categories there is an improvement but for some categories we have a slight deterioration in performance. Figures 6.18 and 6.19 shows a more detailed comparison of categories. The most significant deterioration (-9.2%) is for the Purple Coneflowers and is due to the fact that the flower has a characteristic side profile in most images, which adds to its discriminability (see figure 6.16). It is worth noting that the performance is still

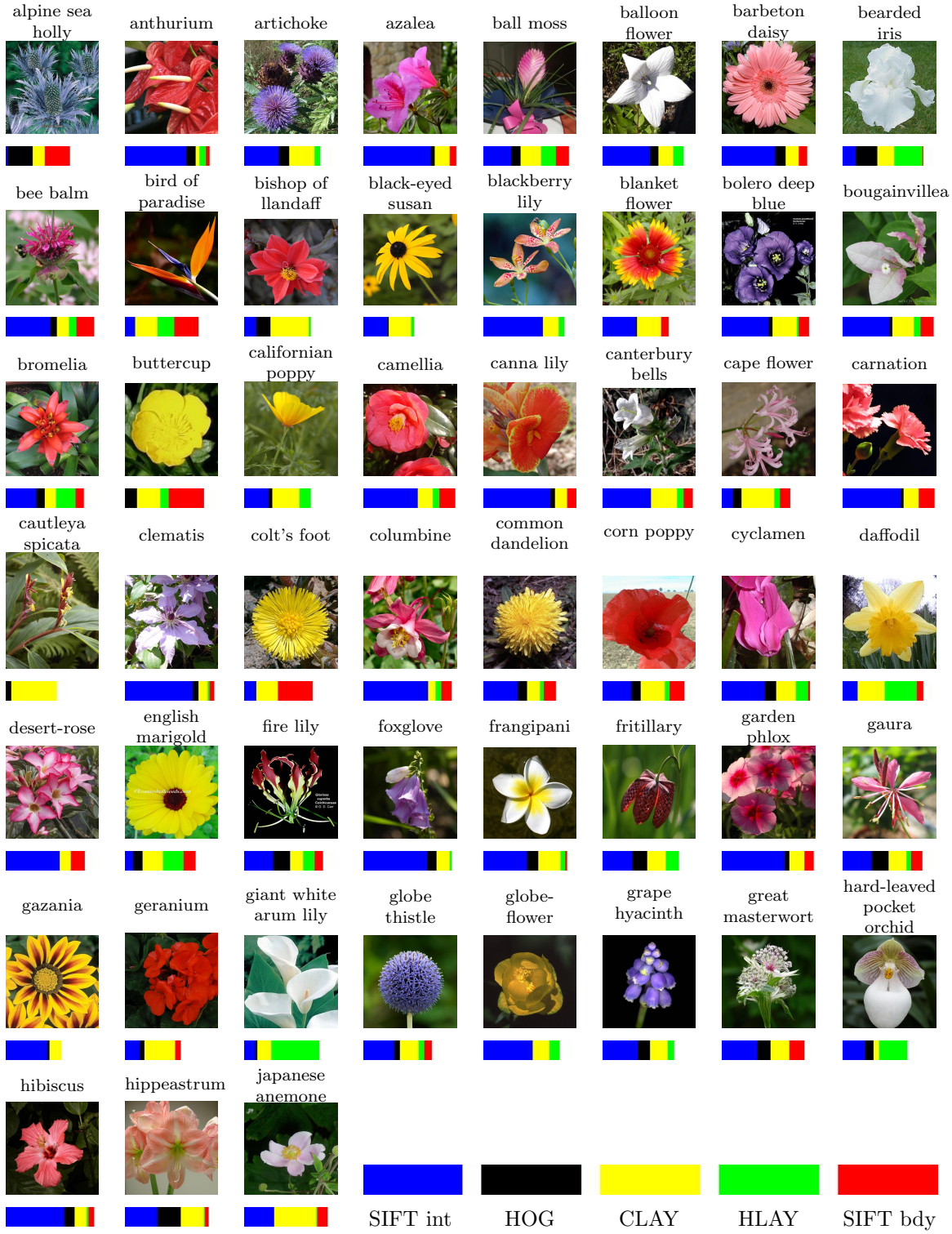


Figure 6.14: Weights for the 102 class flower dataset – Part 1. Each image is an instance of a different class. They are sorted alphabetically. The bars shows the weights for the internal SIFTs, boundary SIFTs, HOGs, CLAYs and HSVs.

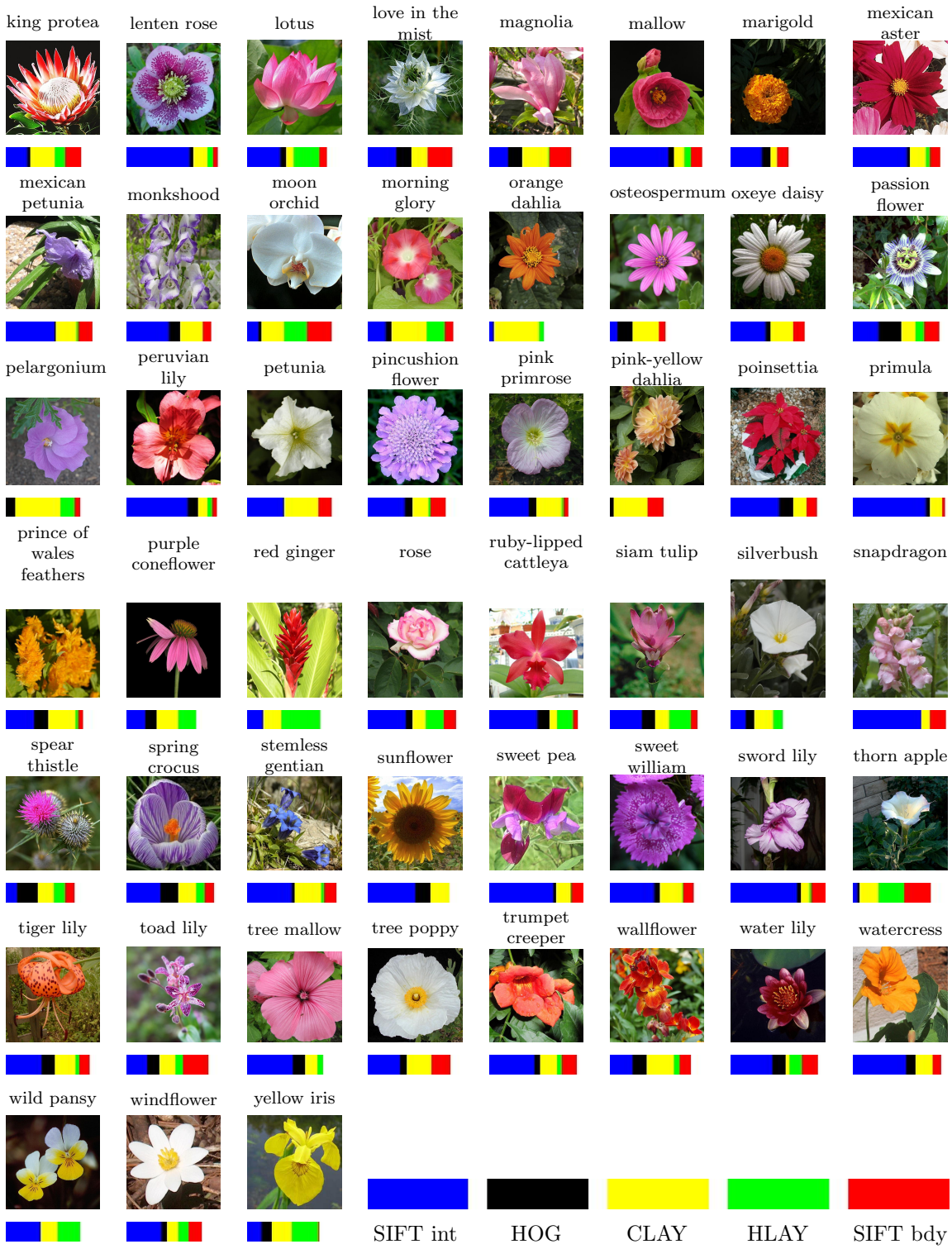
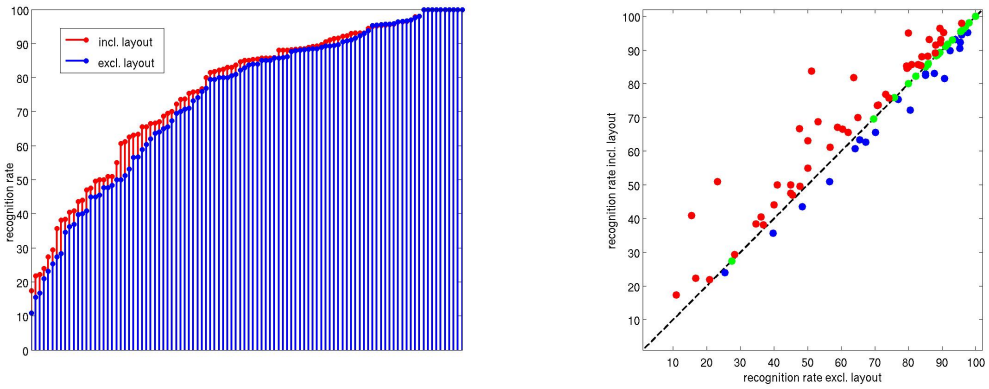


Figure 6.15: Weights for the 102 class flower dataset – Part 2. Each image is an instance of a different class. They are sorted alphabetically. The bars shows the weights for the internal SIFTs, boundary SIFTs, HOGs, CLAYs and HSVs.

high for this category. Many categories are improved. In particular the classification of the Camelias improve by 25.4% from 15.4% to 40.8%, Roses improve by 27.8% from 23.2% to 51.0% and Bromelias improve by 32.5% from 51.2% to 83.7%. These are all categories with large pose variations in the dataset. Note that we also get an improvement for the Sweet Peas, for example, which we did not expect this scheme to work for. Although, the overall recognition rate for the combined classifier is low for categories like the Snapdragons, Sweet Peas and Peruvian lilies. The images that are normalized in 6.4 and 6.10 are all classified correctly if we include the layout features and misclassified if we do not include them. Poor segmentations now account for 14.1% of the misclassifications compared to 9.5% in the previous chapter, but since we are misclassifying fewer images the number of images has not actually increase. This indicates that the layout features has improved classification mainly for the images that were segmented correctly.

Figure 6.16: Purple Coneflower – examples of images in the dataset.



(a) histogram of recognition rates (b) scatter plot of recognition rates

Figure 6.17: Visualization of class improvements for the 102 category test set– a) Histogram of recognition rates. The red bars shows the recognition rates when including the layout descriptors and the blue bar without. The bars are ordered by recognition rate. b) Scatter plot of the recognition rates for each class. Red dots indicate categories where including layout improves performance and the blue dots when it deteriorates performance. The green dots corresponds to equal performance.

Retrieval. An alternative to a single classification is to narrow down the flowers category to a shortlist of categories and view the problem as a retrieval task. Figure 6.20 shows how the performance varies with the length of the short list. Note that within a shortlist of length 10, we have a recognition rate of 95.2%. Figure 6.21 shows some short lists. Each row show a misclassified test image and random samples from the top 5 ranked categories. We can see that there is a strong similarity between the top ranked categories for an image.

Generalization to new images. The goal is to create a system that can classify images taken by anyone. To analyze how the system generalizes to new images, 20 images were collected for 10 of the categories in the database. Most of the images

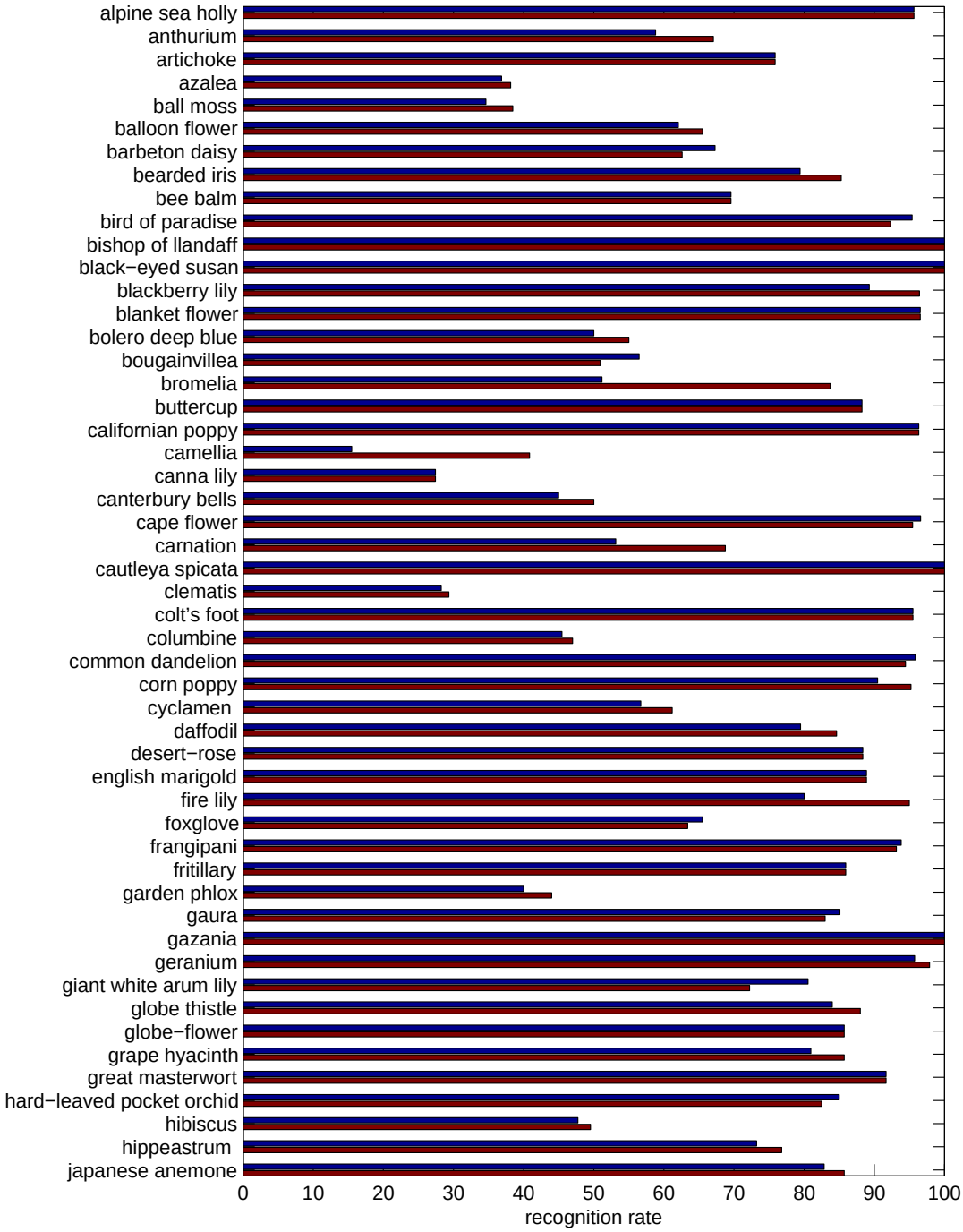


Figure 6.18: Histogram of per-class recognition rates with class labels – Part 1. The red bars shows the recognition rate including the layout features and the blue bar without.

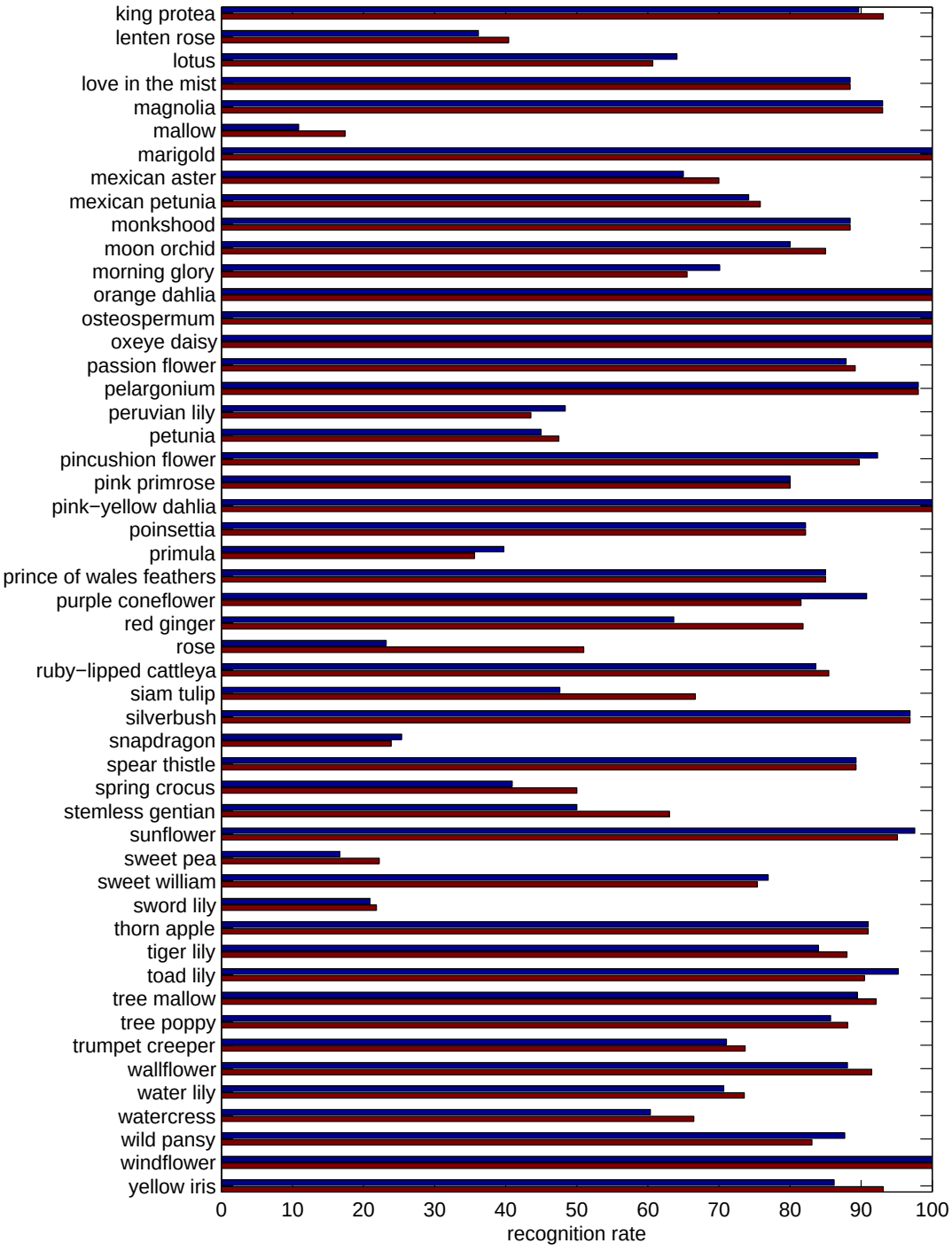


Figure 6.19: Histogram of per-class recognition rates with class labels – Part 2. The red bars shows the recognition rate including the layout features and the blue bar without.

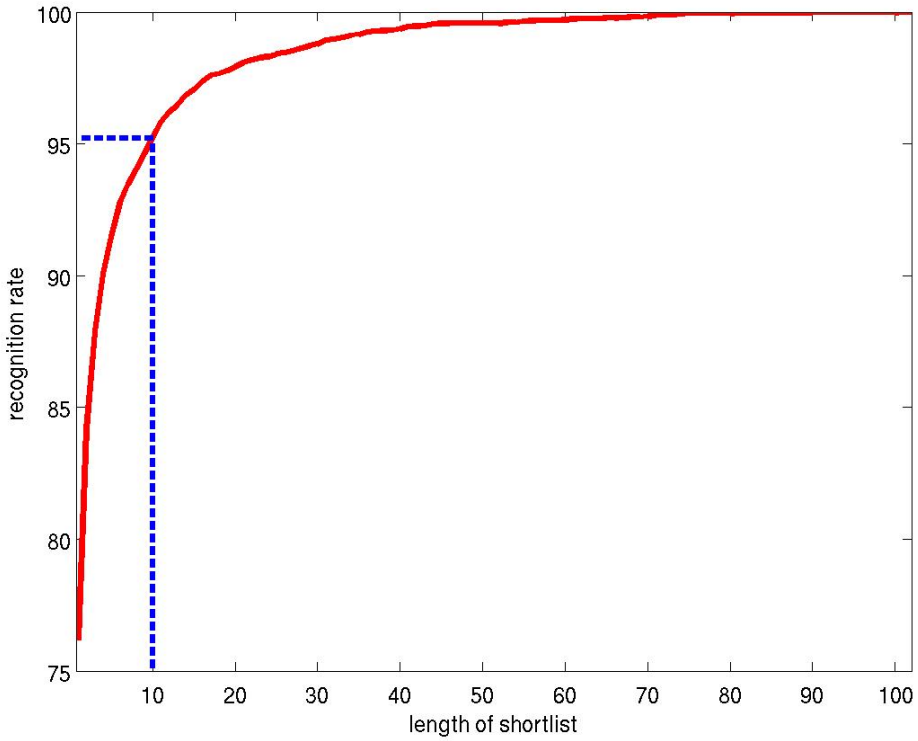


Figure 6.20: Performance variation with shortlist length

were taken in the University of Oxford Botanical Garden. The images were taken by four different people. The instructions given to them was to take pictures of flowers as if they were intending to classify them. Variations in viewpoint, scale and occlusion were encouraged. The selected categories were Japanese Anemone, Waterlily, Black-eyed Susan, Purple Coneflower, Watercress, Tree Mallow, Balloon Flower, Peruvian Lily, Stemless Gentian, Buttercup. These include categories for which both high and low recognition rates are obtained on the 102 category test set. The results are shown in table 6.2. The table shows the recognition rates for each category for both the 102 category database test set and the 20 novel images. For

Figure 6.21: Some classification short lists –The first column shows the test image, and the remaining columns show the top 5 ranked categories for that image. The images shown for the training images is a random sampled training image for that class and not the 'closest' image. The correct class is marked by a green frame.

most classes the recognition rates are similar to those obtained on the 102 category test set. It can be seen that we perform better for some categories, e.g. Japanese Anemone and Purple Coneflower. However, for some categories, e.g. Tree Mallow, Stemless Gentian and Peruvian Lily, the performance is very poor. In particular we are able to classify 92.1% of the Tree Mallows in the 102 category database correctly, but in the new set none are classified correctly. This is because all the Tree Mallows in the dataset are pink and the ones in the new set are all white. The Peruvian Lily is a problem because the texture patterns are important for classifying the images and most of the new images are blurry and have lower resolution than the training images. The mistakes to the Stemless Gentian are confused with the Grape Hyacinth and 60% of the images are classified correctly within the top 2. Overall, the average recognition rate is 58.5% for the closest category and 85% for a shortlist of length 10. Segmentation is also an issue for some of the images. For example, the reflection in the water for the Water lilies and flowers taking up a small portion of the image pose problems for the segmentation.

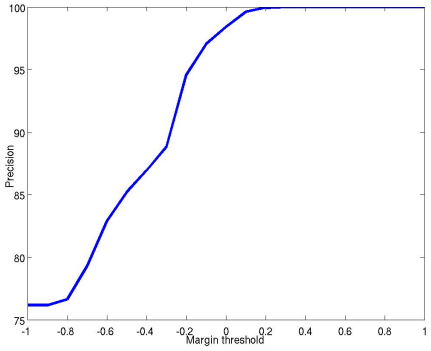
Detecting when image is from unknown category. We want to be able to detect when an image does not belong to one of the categories in the database. This requires some model of what an inlier is. Since we use a discriminative model we do not have a generative model for what a category is, but we can use the margin to get some notion of the confidence in the prediction [85]. Figure 6.22(a) shows the

Category	Test set	New images
Balloon Flower	65.5 %	50 %
Black-eyed Susan	100 %	85 %
Buttercup	88.24 %	70 %
Japanese Anemone	85.7 %	100 %
Peruvian Lily	43.6 %	20 %
Purple Coneflower	81.5 %	90%
Stemless Gentian	63.0 %	30 %
Tree Mallow	92.1 %	0 %
Watercress	66.5 %	65 %
Waterlily	73.5 %	65 %

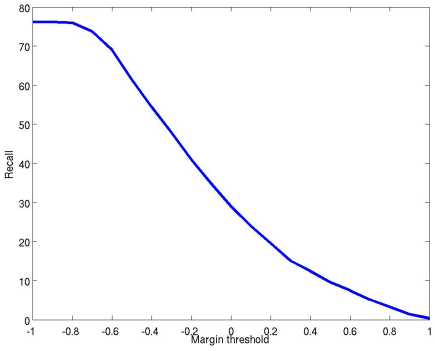
Table 6.2: Recognition rates for classification of new images. For each category the recognition rates are shown for the test set of the 102 category database and for the 20 new images.

precision as we vary the threshold of the margin. We can see that for margins greater than 0.2 we can be confident in the prediction being correct. However, the recall vs margin threshold curve in figure 6.22(b) shows that this also would mean that we miss many correctly classified flowers. Figure 6.22(c) shows the precision-recall trade-off as the margin threshold is decreased.

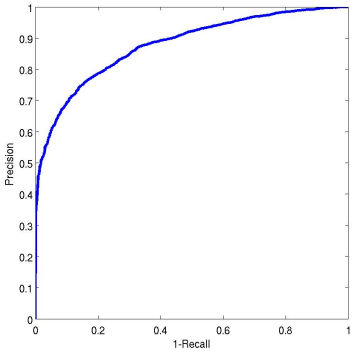
In order to determine the margins for images not in the database, we have randomly collected 200 images of flowers not in the database and another 200 images of other objects. Figure 6.23 shows a histogram of the maximum margins for *i*) correctly classified images in the database *ii*) incorrectly classified images in the database *iii*) flowers not in the database and *iv*) other objects. It can be seen that the margin for the images of flowers not in the database and other object are all smaller than 0.2. It also shows, as seen in figure 6.22(a) as well, that if the margin is greater than 0.2 we classify all the images correctly. Thresholding the margin



(a) Precision vs Margin threshold



(b) Recall vs Margin threshold



(c) Precision vs Recall

Figure 6.22: Precision-Recall curves - The figure shows a)Precision vs Margin threshold, b) Recall vs Margin threshold and c) Precision vs Recall.

at -0.5 would remove 85% of images of both flowers and other objects not in the database. However, as shown in figure 6.22 this would lead to a 15% decrease in recall. We can see that a threshold of -0.6 would increase the precision without a significant loss in recall. In addition, thresholding the margin at -0.6 would remove 70% of images not in the database.

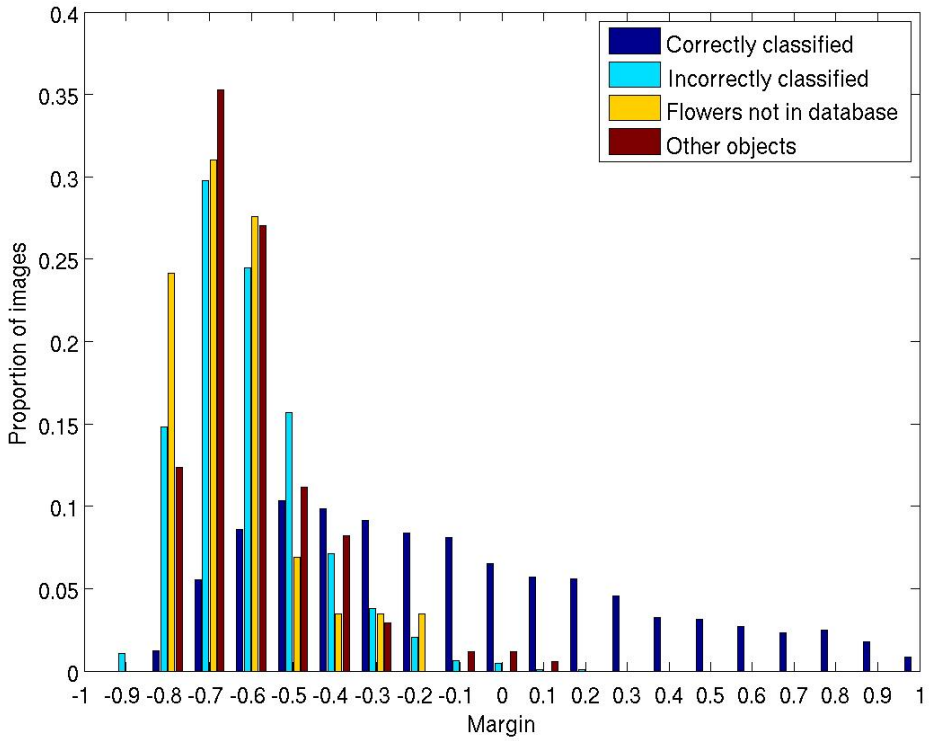


Figure 6.23: Histogram of margins for i) correctly classified images in the database ii) incorrectly classified images in the database iii) flowers not in the database and iv) other objects.

6.6 Conclusion

The layout features were designed to reduce the problem of too much variability in the dataset. We have made use of the segmentation scheme developed in chapter 4 to fit an ellipse and transform the flower to an affine invariant frame. In addition, we have developed two different layout preserving features for the circular regions obtained; one using HSV values and the other one using HOGs. Both features improve classification. We are not able to significantly improve classification of

multiple small flowers, but the features were mainly developed to tackle pose and scale variations of 'single' elliptical flowers. Additional features for different layouts might improve performance, for example the original HOG features. However it is worth noting that already at a short list of length 10, we achieve a 95.2% recognition rate. The system generalizes well to new unseen images, but struggles with images that are significantly different to the samples in the database. Nevertheless, for the new images we are still able to obtain a 85.0% recognition rate for a short list of length 10. We have also shown that thresholding the margin of the SVM can be used as a measure of confidence in order to detect images that are not of flowers in the database.

Chapter 7

Conclusion

This thesis has addressed the problem of flowers classification from images. To this end we have created flower databases and developed schemes for segmenting and classifying flowers. The major contributions of this thesis are:

- Datasets – In chapter 3 we introduced two new datasets for flower classification: a 17 category dataset and a 102 category dataset. Both were chosen to be commonly occurring flowers in the UK. The 102 category dataset poses a recognition challenge somewhat similar to Caltech101 [32], but with the added difficulty of scale and pose variation and also greater inter class similarity. It is also the only available large-scale flower database that can be used straightforwardly for image classification.

- Segmentation and Geometric Modelling – In chapter 4 we proposed an iterative segmentation scheme, which obtains a segmentation by minimizing a CRF using graphcuts. It starts off with a general foreground/background colour distribution and then iteratively learns an image specific distribution by fitting a loose geometric model. We show that our segmentation scheme significantly boosts flowers segmentation both quantitatively on a subset of the 17 category dataset and qualitatively on the 102 category subset. In chapter 5 we showed that the segmentation also boosts classification. In chapter 6 we used the geometric model to develop geometric layout features which are invariant under affine transformations.

- Classification - In chapter 5 we evaluated how tuning a carefully honed vocabulary can improve classification performance. The algorithm and features were first evaluated on the 17 category database and then the best features were applied to the 102 category database. We used a SVM classifier and explored different linear weighted kernel combinations. We found that the multiple kernel learning framework by [112] gave a slight edge, but the performance does not vary much. The misclassifications were largely related to large intra-class variations and class overlap. In chapter 6 we addressed these issues by creating the geometric layout features. We showed that this improved performance further.

7.1 Future Work

We will now present possible direction for future research.

Extend to more categories: We have proposed a framework for classifying visually similar categories which require expert knowledge. This framework can now be easily extended to other categories with similar properties, for example birds, mushrooms, airplanes, cars etc. This can be done by first adapting the segmentation scheme to suit that category. For cars, for example, there are already many geometric models proposed for finding the object (e.g. [57, 83, 97]), these could be incorporated into the iterative segmentation scheme. The next step is then to extract suitable features for that class and learn the best combination.

Hierarchical classification: We know that one of the biggest challenges with flower classification and indeed one of the biggest problems for our system is class overlap. One area which we have not really explored in this thesis is hierarchical classification. Figure 7.1 shows the isomap for the geometric HLAY features. It can be seen that the HLAY features create clusters of tubular flowers and flowers with few large overlapping petals (bottom left), and of flowers with many petals in a circular arrangement (bottom right). At the very bottom right, for example, a Colt's Foot and Dandelion are overlapping. This indicates that even though the HLAY features can separate out these flowers from the rest it is not able to separate

the two. Other features exhibit similar properties for different categories. It would therefore be suitable to classify the flowers hierarchically.

Several approaches have been proposed for hierarchical classification. Zweig and Weinshall [122] and Marszalek and Schmid [70] learn predetermined semantic hierarchies. In [122] the hierarchy is generated manually, which is hardly feasible for a database of our size. In [70] the hierarchy is from an existing semantic taxonomy. This latter approach would lend itself nicely to learning the existing flower taxonomy. Unfortunately, this taxonomy is largely non-visual, i.e. different species with similar appearance might not be very closely related in the taxonomy. Recently, Griffin and Perona [42] and Marszalek and Schmid [71] propose methods that learn an automatic visual hierarchy for the Caltech256 database [41]. Both employ decision tree approaches using a SVM classifier for each decision. In [42] hard decisions are made at each node, whereas in Marszalek [71] the uncertainty of the classifier is taken into account so uncertain decisions are postponed to later. Both approaches are worth exploring as an extension to this work.

Increase training data: Another major issue is the lack of training data. Hierarchical classification might make this less of an issue, but it is also worth investigating how the performance is dependant on the number of training images by changing the data splits and for example using 30 images to train and 10 to test. It also important to ensure that the training data is representative of the class.

Online interface: Finally, in order for this flower classification system to be accessible to the public an online interface needs to be created. This interface should allow people to upload images. The image will then be segmented and classified and the online system will return a short list of possibly 10 categories. In order to avoid segmentation errors for some images it might be necessary to incorporate some user-interface to aid segmentation.

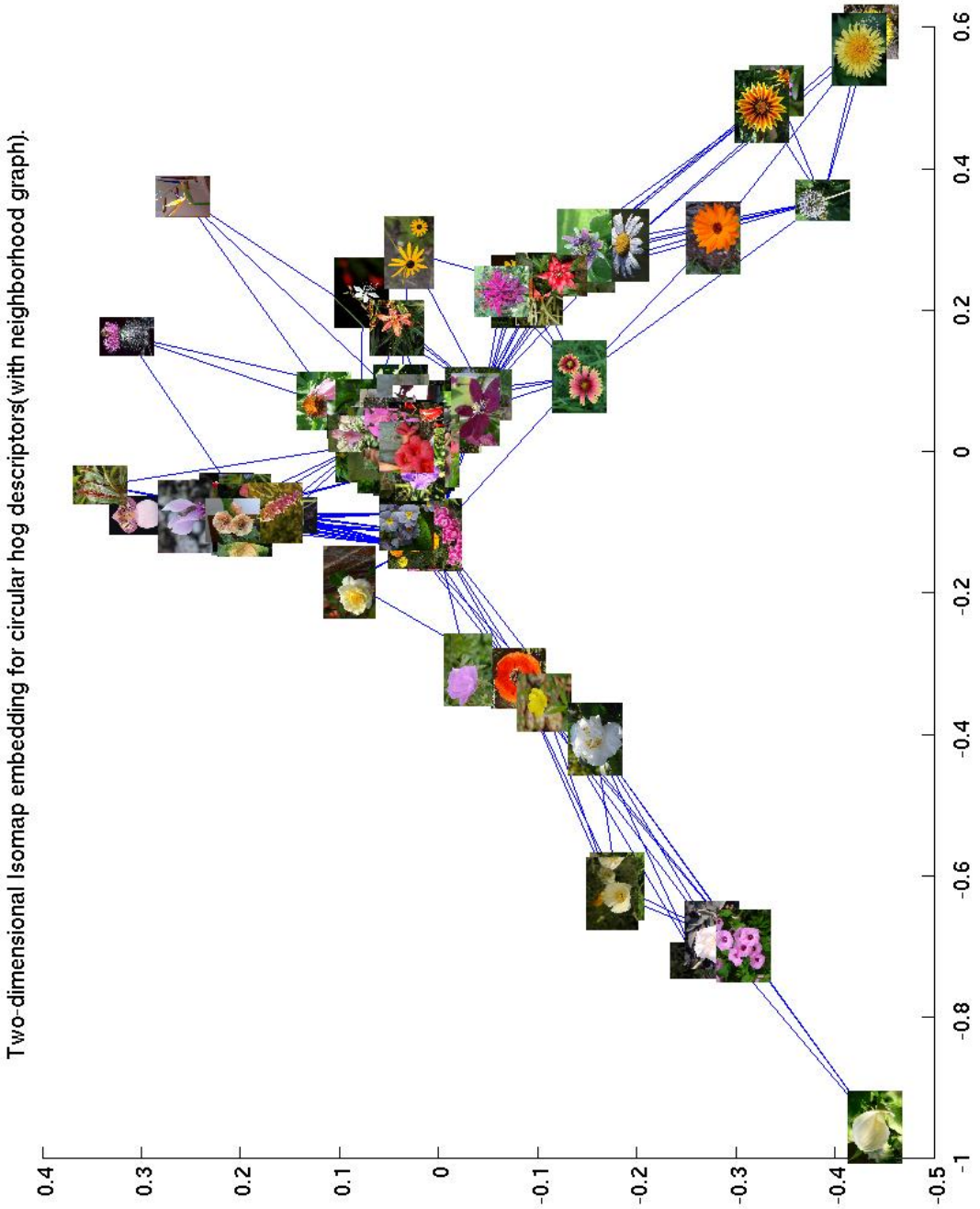


Figure 7.1: Isomap using HLAY features and K nearest neighbour, where $K = 3$. The images displayed are randomly chosen from the class.

Bibliography

[1] Royal horticultural society website. http://www.rhs.org.uk.

[2] Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1300–1305, 1997.

[3] F. Bach, G. Lanckriet, and M. Jordan. Multiple kernel learning, conic duality and the smo algorithm. In *Proceedings of the 21st International Conference on Machine learning, Alberta, Canada*, 2004.

[4] A. Baumberg. Reliable feature matching across widely separated views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 774–781, 2000.

[5] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[6] S. Belongie and J. Malik. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24), 2002.

[7] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[8] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003.

[9] F. Bookstein. Fitting conic sections to scattered data. *Computer Vision, Graphics and Image Processing*, 9:56–71, 1979.

[10] A. Bosch, A. Zisserman, and X. Munoz. Scene classification via pLSA. In *Proceedings of the European Conference on Computer Vision*, 2006.

[11] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *Proceedings of the 11th International Conference on Computer Vision, Rio de Janeiro, Brazil*, 2007.

[12] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the International Conference on Image and Video Retrieval*, 2007.

[13] Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada*, volume 2, pages 105–112, 2001.

[14] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:1124–1137, 2004.

[15] L. Breiman. Random forests. *ML Journal*, 45(1):5–32, 2001.

[16] M. Burl, T. Leung, and P. Perona. Face localization via shape statistics. In *Int. Workshop on Automatic Face and Gesture Recognition*, 1995.

[17] M. Burl, M. Weber, and P. Perona. A probabilistic approach to object recognition using local photometry and global geometry. In *Proceedings of the European Conference on Computer Vision*, pages 628–641, 1998.

[18] M.C. Burl and P. Perona. Recognition of planar object classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 223–230, 1996.

[19] J. F. Canny. Finding edges and lines in images. Master's thesis, MIT, 1983.

[20] V. Caselles, F. Catte, T. Coll, and F. Dibos. A geometric model for active contours in image processing. *Numer. Math.*, 66:1–31, 1993.

[21] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10:266–277, 2001.

[22] H. A. Chipman, E. L. George, and R. E. McCulloch. Bayesian ensemble learning. In *Advances in Neural Information Processing Systems*, 2006.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.

[24] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[25] G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[26] N. Dalal and B Triggs. Histogram of Oriented Gradients for Human Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 886–893, 2005.

[27] M. Das, R. Manmatha, and E. M. Riseman. Indexing flower patent images using domain knowledge. *IEEE Intelligent Systems*, 14(5):24–33, 1999.

[28] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39 B:1–38, 1977.

[29] T. Deselaers, A. Criminisi, J. Winn, and A. Agarwal. Incorporating On-demand Stereo for Real-Time Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[30] G. Dorkó and C. Schmid. Selection of scale-invariant parts for object class recognition. In *Proceedings of the International Conference on Computer Vision*, 2003.

[31] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, 2nd edition, 2001.

[32] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *IEEE CVPR Workshop of Generative Model Based Vision*, 2004.

[33] L. Fei-Fei and P. Perona. A Bayesian hierarchical model for learning natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*, June 2005.

[34] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient matching of pictorial structures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2066–2073, 2000.

[35] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 264–271, June 2003.

[36] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computer*, c-22(1):67–92, January 1973.

[37] A. Fitzgibbon and R. Fisher. A buyer's guide to conic fitting. In *Proceedings of the 6th British Machine Vision Conference, Birmingham*, pages 513–522, 1995.

[38] P.-E. Forssén. Maximally stable colour regions for recognition and matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[39] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings*, 1996.

[40] K. Grauman and T. Darrell. The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. In *Proceedings of the International Conference on Computer Vision*, 2005.

[41] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

[42] G Griffin and P. Perona. Learning and Using Taxonomies For Fast Visual Categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[43] R. Halir and J. Flusser. Numerically stable direct least squares fitting of ellipses. In *In Proc. 6th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media*, 1998.

[44] C. G. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference, Manchester*, pages 147–151, 1988.

[45] T. Hofmann. Probabilistic latent semantic indexing. In *Special Interest Group on Information Retrieval*, 1999.

[46] F. Jurie and C. Schmid. Scale-invariant shape features for recognition of object categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC*, pages 90–96, 2004.

[47] T. Kadir and M. Brady. Scale, saliency and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001.

[48] T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. In *Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic*. Springer-Verlag, May 2004.

[49] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4), 1987.

[50] M. Kirby and L. Sirovich. Applications of the Karhunen-Loeve procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):103–108, January 1990.

[51] M. P. Kumar, P. H. S. Torr, and A. Zisserman. OBJ CUT. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*, volume 1, pages 18–25, 2005.

[52] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson. Object recognition by affine invariant matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 335–344, 1988.

[53] G. R. G. Lanckriet, N. Cristianini, P. Barlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[54] S. Lazebnik, C. Schmid, and J Ponce. Beyound Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York*, 2006.

[55] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[56] Y. LeCun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard. Handwritten digit recognition: Applications of neural net chips and automatic learning. *IEEE Communication*, pages 41–46, 1989.

[57] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on Statistical Learning in Computer Vision, ECCV*, May 2004.

[58] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, 2006.

[59] T. Leung. Texton correlation for recognition. In *Proceedings of the European Conference on Computer Vision*, 2004.

[60] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44, June 2001.

[61] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):77–116, 1998.

[62] T. Lindeberg and J. Gårding. Shape-adapted smoothing in estimation of 3-D depth cues from affine distortions of local 2-D brightness structure. In *Proceedings of the 3rd European Conference on Computer Vision, Stockholm, Sweden*, LNCS 800, pages 389–400, May 1994.

[63] C. Linneaus. *Systemae Naturae*. Impensis Direct. Laurentii Salvii, 1759.

[64] D. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 1150–1157, September 1999.

[65] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[66] T. Malisiewicz and A. Efros. Recognition by association via learning per-exemplar distances. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[67] C. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[68] R. Marée, P. Geurts, J. Piater, and L Wehenkel. Random Subwindows for Robust Image Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*, 2005.

[69] M. Marszalek and C. Schmid. Spatial weighting for bag-of-features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2118–2125, 2006.

[70] M. Marszalek and C. Schmid. Semantic hierarchies for visual object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, jun 2007.

[71] M. Marszalek and C. Schmid. Constructing Category Hierarchies for Visual Recognition. In *Proceedings of the European Conference on Computer Vision*, pages 479–491, oct 2008.

[72] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference*, pages 384–393, 2002.

[73] G. McLachlan and T. Krishnan. *The EM algorithm and extensions*. Wiley series in probability and statistics. John Wiley & Sons, 1997.

[74] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada*, 2001.

[75] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*. Springer-Verlag, 2002.

[76] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[77] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004. submitted to PAMI.

[78] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic*. Springer-Verlag, May 2004.

[79] F. Moosman, B. Triggs, and F. Jurie. Fast discriminative visual codebook using randomized clustering forests. In *Advances in Neural Information Processing Systems*, 2006.

[80] E. Mortensen and W. A. Barrett. Intelligent scissors for image composition. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, pages 191–198, 1995.

[81] H. Murase and S. Nayar. Visual learning and recognition of 3D objects from appearance. *International Journal of Computer Vision*, 14(1), 1995.

[82] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *Proceedings of the European Conference on Computer Vision*, 2006.

[83] A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. In *Proceedings of the European Conference on Computer Vision*, 2006.

[84] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton–jacobi formulations. *Journal of Computational Physics*, 79(12–49), 1988.

[85] John C. Platt and John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[86] J. Ponce, T. L. Berg, M. Everingham, D. A. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B. C. Russell, A. Torralba, C. K. I. Williams, J. Zhang, and A. Zisserman. Dataset issues in object recognition. In J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, editors, *Toward Category-Level Object Recognition*, volume 4170 of *LNCS*, pages 29–48. Springer, 2006.

[87] X. Ren and J. Malik. Learning a classification model for segmentation. In *Proceedings of the International Conference on Computer Vision*, volume 1, pages 10–17, 2003.

[88] L. G. Roberts. Machine perception of three-dimensional solids. *Optical and electro-optical information processing, J. Tippet et al., Ed.*, 1965.

[89] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: interactive foreground extraction using iterated graph cuts. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, 23(3):309–314, 2004.

[90] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, January 1998.

[91] T. Saitoh, K. Aoki, and T. Kaneko. Automatic recognition of blooming flowers. In *Proceedings of the International Conference on Pattern Recognition*, volume 1, pages 27–30, 2004.

[92] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "How do I organize my holiday snaps?". In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, volume 1, pages 414–431. Springer-Verlag, 2002.

[93] R. E. Schapire and Y. Singer. Improove boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[94] C. Schmid, R. Mohr, and C. Bauckhage. Comparing and evaluating interest points. In *Proceedings of the International Conference on Computer Vision*, pages 230–235, 1998.

[95] H. Schneiderman and T. Kanade. A statistical method for 3D object detection applied to faces and cars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[96] B. Scholkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

[97] J. Shotton, A. Blake, and R. Cipolla. Contour-Based Learning for Object Detection. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, China*, 2005.

[98] J. Shotton, M. Johnson, and R. Cipolla. Semantic Texton Forests for Image Categorization and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[99] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering object categories in image collections. In *Proceedings of the International Conference on Computer Vision*, 2005.

[100] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, October 2003.

[101] M. Sonka, V. Hlavac, and R. Boyle. *Image processing, analysis and machine vision*. Chapman and Hall, London, 1993.

[102] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large Scale Multiple Kernel Learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[103] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, November 1991.

[104] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[105] A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiele, and L. Van Gool. Towards multi-view object class detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2006.

[106] E. Tola, V. Lepetit, and P. Fua. A Fast Local Descriptor for Dense Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[107] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC*, pages 762–769, 2004.

[108] M. Turk and A. P. Pentland. Face recognition using eigenfaces. In *CVPR*, pages 586–591, 1991.

[109] T. Tuytelaars, A. Turina, and L. Van Gool. Noncombinatorial detection of regular repetitions under perspective skew. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):418–432, 2003.

[110] I. Ulusoy and C. Bishop. Comparison of generative and discriminative techniques for object detection and classification. In J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, editors, *Toward Category-Level Object Recognition*, LNCS. Springer, 2006.

[111] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1999.

[112] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *Proceedings of the International Conference on Computer Vision*, Rio de Janeiro, Brazil, October 2007.

[113] M. Varma and A. Zisserman. Classifying images of materials: Achieving viewpoint and illumination independence. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, volume 3, pages 255–271. Springer-Verlag, May 2002.

[114] M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 691–698, June 2003.

[115] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 511–518, 2001.

[116] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *Proceedings of the European Conference on Computer Vision*, pages 18–32, 2000.

[117] J. Winn, Criminisi, A., and T. Minka. Object Categorization by Learned Universal Visual Dictionary. *Proceedings of the 10th International Conference on Computer Vision, Beijing, China*, 2005.

[118] J. Winn and J. Shotton. The Layout Consistent Random Field for Recognizing and Segmenting Partially Occluded Objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York*, 2006.

[119] P. Yin, A. Criminisi, J. Winn, and I. Essa. Tree-based Classifiers for Bilayer Video Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis*, 2007.

[120] H Zhang, A. C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2126–2136, 2006.

[121] J. Zhang, M. Marszalek, S. Lazebnik, and Schmid C. Local features and kernels for classification of texture and object categories: a comprehensive study. *International Journal of Computer Vision*, 2007.

[122] A. Zweig and D. Weinshall. Exploiting Object Hierarchy: Combining Models from Different Category Levels. In *Proceedings of the International Conference on Computer Vision*, pages 1–8, 2007.