First Year Report

Learning Class-Specific Segmentation

Jamie Shotton
jdjs2@cam.ac.uk

Department of Engineering
University of Cambridge

Supervisors:

Roberto Cipolla
cipolla@eng.cam.ac.uk

Andrew Blake
ablake@microsoft.com

This work was supported by a Microsoft Research Studentship.

August 27, 2004

ABSTRACT

This report details the work undertaken this year towards class-specific segmentation. The aim is to take an image known to contain an object of a particular class, and return for each pixel a figure-ground segmentation value. A training corpus consisting of images and their ground-truth segmentation masks is used to learn shape and appearance models. Our shape model consists of local shape patches learned using a new translationally-invariant clustering algorithm, together with learned adjacency statistics applied to enforce consistency between neighbouring patches. Our appearance model is a database of patches. Given a novel test image, hypotheses of underlying shape and appearance are constructed, and a final belief-propagation algorithm enforces global consistency.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

This report details the work undertaken this year towards class-specific segmentation. Image segmentation has the very broad definition of dividing an image into regions, and we restrict our interest to *class-specific* segmentation: the aim is to take an image known to contain an object of a particular class, and return a segmentation for the whole image, as illustrated in fig. 1.1. This type of segmentation is often described as *figure-ground* since the binary valued segmentation result associates each pixel with either object (*figure*), or background (*ground*).

1.1 Motivation

There are numerous applications of and prospects for automatic image segmentation. While an easy, though time-consuming, task for humans, at present the best computer-based techniques are rather limited in accuracy without a fair amount of user-interaction. This has so far limited the usefulness of segmentation to mainly image editing and compositing. However, if one could accurately and reliably segment objects from images or video sequences a

(a) (b)

Figure 1.1: An illustration of figure-ground segmentations: (a) the input image; (b) the desired binary valued figure-ground segmentation.

whole realm of applications opens up.

Segmentation is intimately linked with both object detection and recognition. Many computerised recognition systems require pre-processing stages to segment or register training or test data to negate the effects of clutter (spurious background). Conversely as we shall see, knowledge of the class of object being segmented can greatly improve results. By finding those regions of video sequences which correspond to foreground objects, one could effectively track the objects (similar to the tracking as detection paradigm of [69]). The background environment in which an object is situated is also very informative about the types of object likely to be there (e.g. one would expect to see a toaster in a kitchen) and this insight has been used in [71], but this relationship holds both ways: knowledge of the presence of an object (gained through figure-ground segmentation for example) could help with localisation.

The link between segmentation and recognition forms a chicken-and-egg situation: segmentation can aid recognition by removing background clutter, but recognising an object can help image segmentation by providing top-down clues.

1.2 Shortcomings of Existing Approaches

1.2.1 Bottom-Up Segmentation

The traditional, bottom-up, approach to segmentation looks for low-level image cues such as edges, colour and texture (e.g. [66, 50, 11, 58]). For example, areas of near constant intensity or texture, or areas bounded by edge contours, are *possibly* good candidates to segment as contiguous regions. Recent approaches such as [11, 58, 66] have used increasingly sophisticated methods to attempt to get figure-ground segmentations and with some excellent results, though often with significant user interaction.

However, the bottom-up approach will perhaps never be able to segment images as competently as humans, since there is often much potential variation in both the appearance of even a single object and the appearance of the background in which it is situated. As an example, consider an image of someone wearing a red shirt and blue trousers. A naïve bottom-up segmentation algorithm would likely split the image into two regions. While potentially useful, this is not usually what is wanted. Another common failing of bottom-up segmentation approaches is that often there is too much in common between the foreground and background. If the person in our example now stands against a blue background (such

as in the blue-screening segmentation technology used for film and television), it becomes very difficult to distinguish between the trousers and the background, and consequently a simplistic algorithm would mark the trousers as background. This can be used positively for special effects, but for the problem concerning us we wish to delineate the *whole* person regardless of background. Of course, in the limiting case it becomes impossible to segment when the background exactly matches the foreground; but clearly it should be possible to use high-level cues to aid us when only some areas are ambiguous in this fashion.

In addition to these problems, experiments with the human visual system (see [32] for a good introduction to the human visual system) conclude that we use additional high-level criteria to help segment the objects we see, suggesting that it should be possible to use high-level information for good effect in machine vision segmentation.

1.2.2 Top-Down Segmentation

In an attempt to rectify these problems, an alternative, top-down, approach to segmentation has been suggested that uses high-level prior knowledge about shape and appearance for a particular class of objects to guide the image segmentation process. In general, this prior information must be *learned* from a set of training data, and then should be applicable to any image known to contain an instance of the class that has been learned. Continuing our example, if we know in general that most people have four limbs arranged in a certain fashion around a torso, we should be able to look for the limbs and torso in an image and join them into a contiguous segmentation of the person, hopefully relatively independently of background. When certain areas are less distinguishable, such as blue trousers against blue background, the higher-level shape cues should come into play.

One technique ([75]) that has had success recently has been to use small, rectangular fragments (patches) of images for object detection and recognition. The work by Borenstein & Ullman [9] presents a fragment-based approach to top-down segmentation, requiring hand-segmented training examples; their paper [10] extends this to cope with unsegmented training examples.

1.3 Approach

We present in this section a brief outline of our technique. We learn from a training corpus, consisting of images and their ground-truth segmentation masks, models of shape

and appearance. Our shape model consists of local shape patches learned using a new translationally-invariant clustering algorithm, together with learned adjacency statistics applied to enforce consistency between neighbouring patches. Our appearance model is a database of image patches. Given a novel test image, hypotheses of underlying shape and appearance are constructed, and figure-ground boundaries are correctly accounted for. Finally, we run a belief-propagation algorithm to enforce global consistency of our segmentation output.

1.4 Structure of the Report

This report is divided into five chapters, the first of which being this introduction. Chapter 2 presents a survey of some of the literature in the areas of object recognition and image segmentation. Chapter 3 presents our investigation into the methods of Borenstein & Ullman, while Chapter 4 presents our new work towards the goal of class-specific segmentation. Finally, we present our conclusions and future work in Chapter 5. Various Appendices at the end of the document give extra information to accompany the main body of the text.

1.5 Acknowledgements

I am indebted to several people for their help with my research this year and in writing this report. I would like to thank my supervisors, Roberto Cipolla and Andrew Blake, for their guidance and support; all the members of our research group for many useful and interesting discussions, especially George Vogiatzis, Ollie Williams, and additionally for providing me with an implementation of [11], Stefano Bucciarelli; Eran Borenstein for discussing his techniques at ECCV 2004; Bastian Leibe for supplying training and test data of cars and cows; John Winn and Carsten Rother for interesting discussions of Epitome Images; and last but not least, my father, David Shotton, for his help editing this report, and general academic encouragement.

This work has been financially supported by a Microsoft Research Studentship.

CHAPTER 2

LITERATURE SURVEY

In this chapter we present a brief survey of recent literature in computer vision, focusing slightly on the area of object recognition, and introduce a few papers which will be examined in much more detail later on in the report. The chapter is divided into sections covering similar areas of the literature. We start by examining a variety of interest point detectors in §2.1. Next we briefly examine in §2.2 a few local descriptors that can characterise detected interest points. We then move on to object recognition and detection in §2.3, followed by segmentation in §2.4, and finally a few miscellaneous papers in §2.5.

2.1 Interest Point Detectors

A very powerful and widely-used technique in object recognition is interest point detection. This low-level approach involves the automatic selection of a large number of small, local features of interest, which can subsequently be used for matching and indexing.[1] The power of the technique derives from the *locality* of the interest points. Firstly, significant occlusion can be tolerated since only a few good matches (out of potentially thousands of interest points) should be necessary for recognition or other applications. Secondly, local features can be made invariant to a wide variety of image transformations and object deformations.

The field of Computer Vision has, until moderately recently, focused much attention on edge (such as Canny [15]) and corner (such as Harris [33]) detectors, both of which can be implemented extremely efficiently. Advances in the last decade or so, in both computational power and mathematical understanding, have significantly improved these methods, and

[1] The term 'point' is slightly misleading: most interest point detectors actually detect *regions* of interest in the image rather than just a single point.

we now have at our disposal a considerable selection of powerful interest point detectors (e.g. [51, 74, 38, 47, 53]) which have been used to great effect in e.g. [67, 23].

Improvements have largely centred on adding invariance to a variety of transformations, notably photometric and geometric.[2] The photometric invariance often addressed is that of affine illumination changes, both global (e.g. to compensate for different exposure settings on the camera) and local (e.g. to compensate for moving light sources). The geometric invariances achieved have advanced considerably from simple rotation and scale invariances to now full affine invariance, which for sufficiently planar scenes at a distance approximates true perspective invariance accurately.[3]

In this section we present an overview of some of these new interest point detectors. We concentrate here on the actual detectors as presented rather than their applications and results which tend to be very similar. §2.1.1 and §2.1.2 present two similar techniques which analyse the image pixels directly in a similar manner to watershed segmentation ([4]). §2.1.3 presents a novel scheme based on local entropy measures. §2.1.4, §2.1.5 and §2.1.6 present several schemes based on the responses of filters convolved with the image.

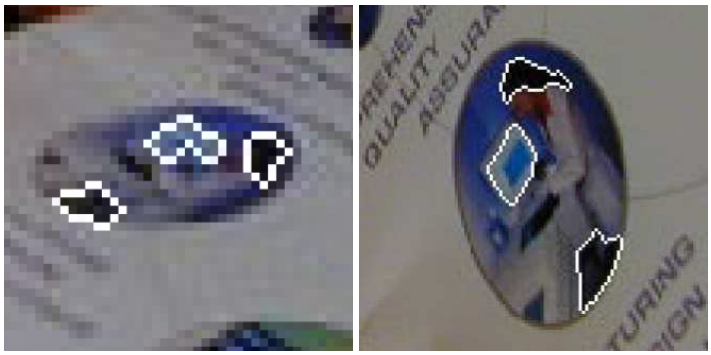
2.1.1 Maximally Stable Extremal Regions
Matas et al [51]

This paper describes a new interest point detector which analyses image pixels directly to select regions of an image satisfying certain 'maximally stable extremal' conditions.

An 'extremal' region is defined as a contiguous region of the image, containing pixels with intensities all greater than the intensities on the boundary of the region. Formally, they denote an image $I : D \rightarrow S$ as a mapping from coordinates D to intensities S, and, using a neighbourhood relation $\diamond$ (here, representing the 4-connected neighbourhood in the image), define a region as a subset $Q \subseteq D$ such that for all $p, q \in Q$ there exists a sequence $p, a_1, a_2, \ldots, a_n, q$ where $\{a_1, a_2, \ldots, a_n\} \subseteq Q$ and $p \diamond a_1, a_1 \diamond a_2, \ldots, a_n \diamond q$. Defining the region boundary $\partial Q = \{q \in D \setminus Q : \exists p \in Q : q \diamond p\}$, they specify an extremal region $R \subseteq D$ as a

[2] For a detector to be invariant to a certain class of transformation means that transformation and detection should be commutative operations. So, for example, a scale invariant detector should find the same points in two images of the same scene taken at different scales (up to occlusion). Of course, achieving exact (i.e. perfect) invariance is very difficult in practice due to artefacts in the imaging process such as aliasing. Therefore invariance is usually used to mean *theoretical* invariance, which in practice means only *tolerance* to certain transformations.

[3] In this report, 'geometric affine invariance' will be synonymous with 'affine invariance'.



(a) (b)

Figure 2.1: Example MSE regions detected in two views of the same scene are highlighted with white outlines. Note correct correspondences despite widely different viewpoints. Note also that it tends to detect regions of high or low intensities relative to their surroundings. [From [51]].

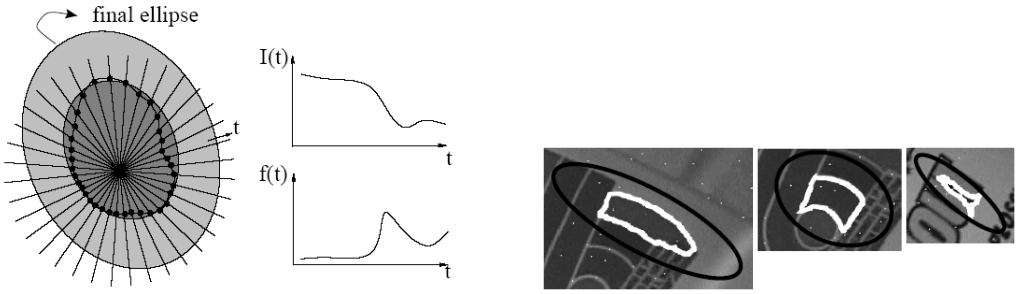
region such that for all $p \in R$ and all $q \in \partial R$, $I(p) > I(q)$.[4]

The 'maximally stable' extremal (MSE) region is then defined as the member of a sequence of nested extremal regions at which the relative change in area achieves a local minimum. Let $R_1, \ldots, R_i, R_{i+1}, \ldots$ denote a sequence of nested extremal regions, i.e. for all i, $R_i \subset R_{i+1}$. Extremal region R_{i^*} is maximally stable iff $q(i) = |R_{i+\triangle} \backslash R_{i-\triangle}|/|R_i|$ has a local minimum at i^*, where $\triangle$ is an integer parameter of the method. In words this says that the maximally stable region is the region in a sequence for which the relative change in area achieves a minimum.

MSE regions are invariant under continuous transformation of image coordinates (including perspective and affine), and under monotonic transformation of image intensities. The MSE region detector tends to find regions of high and low intensities relative to their surroundings, as one would expect from the definitions. Fig. 2.1 shows an object viewed from two different angles with the detected MSE regions indicated with white outlines.

They seem useful (e.g. in [67]) and are very efficient to detect, but they tend to pick out only extreme intensity patterns, i.e. bright and dark patches, an especially bad example of which being specularities which ideally should be ignored as lighting artefacts. One could envisage many situations where this detector would fail to find anything useful, for example highly textured images. It seems therefore that these features, while useful, are far from

[4]These definitions result in bright extremal regions. The dual definition where greater than is replaced with less than is also used to get dark extremal regions.



(a) (b)

Figure 2.2: (a) Rays are projected outwards from local intensity extrema along which the extrema of function $f(t)$ are located. An affinely-invariant ellipse is then fitted to these points. (b) Example regions found: linked points outlined in white; fitted ellipses outlined in black. [From [74]].

sufficient used alone.

2.1.2 Local, Affinely Invariant Regions

***Tuytelaars & Van Gool* [74]**

In a similar vein to the paper presented above, they describe an intensity-based affine invariant interest point detector that does not rely on edge or corner features. The method starts at local extrema in image intensity, reasoning that, while not as accurately localisable as corners, they can withstand large illumination changes and are less likely to be located at object boundaries.[5]

Rays are projected outwards from the intensity extrema, and along each ray the following function is evaluated:

$$f(t) = \frac{|I(t) - I(0)|}{\max(\frac{1}{t}\int_0^t |I(x) - I(0)|\mathrm{d}x, d)} \tag{2.1}$$

where t is the distance along the ray from the origin, and d is a small constant to prevent division by zero. The point for which this function reaches an extremum is invariant under affine geometric and photometric transformations. Hence to create an invariant region these extrema are joined from one ray to the next, forming a closed region (see fig. 2.2). An ellipse is then fitted to the invariant shape such that the resulting region is still affine invariant.

[5]Interest points that are located at object boundaries will straddle both object and background, a common problem that we will investigate further in §4.5.2.

Due to the intensity-extrema based nature, as for the previous paper, this technique suffers problems with specularities. However, their evaluation shows good results up to viewpoint changes of about 60 degrees where scaling due to foreshortening and specularities affect the results.

In conclusion, these features are able to pick up fine detail since there is no pre-smoothing of the image as in many other approaches. But they can only be located at local intensity extrema, a condition which may be too sensitive (highly textured regions) or too insensitive (largely untextured regions) to produce sensible results. They should be considered a useful contribution but seem to have been overshadowed by the newer techniques.

2.1.3 Scale Saliency

***Kadir et al* [38, 37, 39]**

This work presents a completely novel interest point detector, based on a notion of 'saliency'. They note that informative, or salient, features of an image usually exhibit unpredictability. For example, the eye has an unusual appearance compared with the rest of the face and so is probably a good local feature to choose. Using the information theoretic definition for unpredictability as Shannon entropy ([49] is a good text on Information Theory), they suggest a three stage interest point detector to exploit this insight:

1. Calculate the entropy of local appearance as a function of scale.
2. Select scales for which the entropy is achieves a local maximum with respect to scale.
3. Weight the entropy scores by an inter-scale unpredictability measure.

These steps are run for each pixel in the image, and the N regions with highest saliency values are chosen as salient features.

For the first step, the entropy of local appearance is calculated as follows. For a particular scale s and position $\mathbf{x}$, intensity histograms[6] are created from a window of support centred on point $\mathbf{x}$ of size proportional to the scale. These histograms are used to approximate the local probability distribution of appearance $p(I|s,\mathbf{x})$, from which the entropy can be calculated[7] as:

$$H(s,\mathbf{x}) = -\int p(I|s,\mathbf{x}) \log_2 p(I|s,\mathbf{x}) \mathrm{d}I \tag{2.2}$$

[6] Colour or orientation information could also be used to generate the appearance histograms.
[7] Of course, the discrete versions of these equations are used in practice.

The second step chooses for each pixel a set $S(\mathbf{x})$ of candidate scales based on local maxima of the entropy measure with respect to scale:

$$S(\mathbf{x}) = \{s : \frac{\partial H(s, \mathbf{x})}{\partial s} = 0, \frac{\partial^2 H(s, \mathbf{x})}{\partial s^2} < 0\} \tag{2.3}$$

Finally for the third step, a measure of inter-scale unpredictability is calculated for the candidate scales:

$$W(s, \mathbf{x}) = s \int \left| \frac{\partial}{\partial s} p(I|s, \mathbf{x}) \right| \mathrm{d}I \tag{2.4}$$

This W is used to weight the entropy H to produce the saliency measure Y:

$$Y(s, \mathbf{x}) = H(s, \mathbf{x}) W(s, \mathbf{x}) \tag{2.5}$$

For all pixels $\mathbf{x}$ and candidate scales $s \in S(\mathbf{x})$, the saliency measure $Y(s, \mathbf{x})$ is calculated and the N regions (pairs of position and scale) with highest saliency scores Y are taken as the salient regions for the image.

As a motivation for this algorithm, consider fig. 2.3. Here, (a) shows an image of a black circle against a white background. In this example, two points have been picked out for comparison: one centred on the circle (blue) and the other centred at the edge (red). Looking at the image, we would probably expect that the blue circle should be a better local feature than the red circle. Examining the entropy responses in (b) as a function of scale, both achieve a good peak at seemingly sensible scales. But note that entropy on its own would not produce a saliency measure discriminating towards the blue circle, since the peaks occur at roughly equal values. This is because in calculating entropy, all positional information is thrown away and both regions in this example can attain the highest possible entropy at roughly equal mixes of black and white pixels. This justifies the use of the inter-scale unpredictability measure which weights in favour of regions for which the local probability density is changing rapidly with respect to scale. This is born out in (c) where the blue graph gets a much higher saliency value at the chosen scale than does the red graph, meaning that, in the final thresholding, the blue region is much more likely to be selected as a salient region, as desired.

In this formulation from [38], invariance to rotation, scale, and photometric shifts is provided. In [39] they extend the technique to provide affine geometric invariance, by simply extending the search space from circular regions specified by scale, to elliptical regions spec-

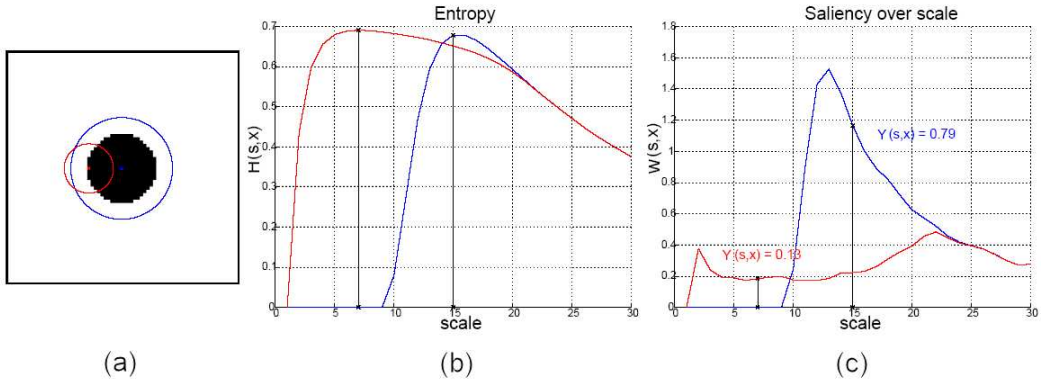


Figure 2.3: (a) An image of a black circle against a white background. Superimposed in blue is a circle representing a salient region centred on the black circle, and in red is a circle representing a salient region centred on the edge of the black circle. (b) The graphs of entropy of appearance as a function of scale for the two centres. (c) The weighted saliency measure $Y(s, \mathbf{x})$ as a function of scale. [From [39]].

ified by scale, axis ratio, and orientation; this large search space consequently makes the affine invariant version of the detector very slow.

Throwing away most of the spatial information and using just local entropy may seem a bad thing to do. However, since pixels can be permuted within the window of support to produce the same result (assuming W is left relatively undisturbed), this they claim provides some invariance to small intra-class deformations. Their feature detector has been used successfully by Fergus et al in [23] for object category recognition where this deformation invariance is very useful; see §2.3.1 for more details.

2.1.4 Difference of Gaussians

***Lowe* [47, 48]**

Here we describe the interest point *detector* used in this paper, though most of the paper is concerned with the 'SIFT' feature point *descriptor*, a way of characterising the interest points (see §2.2). Their system was designed with high efficiency in mind, and hence a very simple interest point detector was used.

They propose using scale-space peaks in the difference-of-Gaussians (DoG) function convolved with the image $I(x, y)$ to localise interest points:

$$D(x, y, \sigma) = (\mathcal{G}(x, y, k\sigma) - \mathcal{G}(x, y, \sigma)) \star I(x, y) \tag{2.6}$$

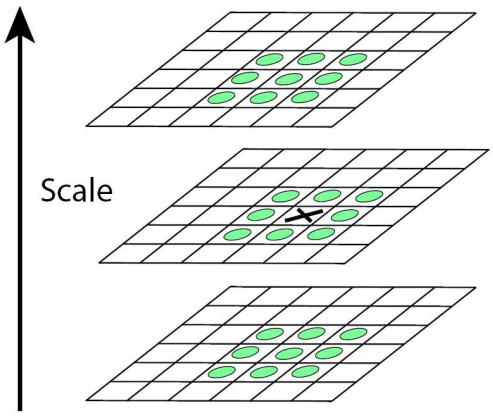


Figure 2.4: Interest points are detected at local maxima in scale space of the difference-of-Gaussian function by comparing the filter response with its 26 neighbours in space and scale. [From [48]].

where $\mathcal{G}(x, y, \sigma)$ is an 2-D isotropic Gaussian convolution kernel (suitably truncated) with standard deviation σ. This can be efficiently computed by pre-smoothing the image as

$$L(x, y, \sigma) = \mathcal{G}(x, y, \sigma) \star I(x, y) \tag{2.7}$$

where $\sigma \in \{s, sk, sk^2, \ldots\}$, for some starting scale s and geometric sequence ratio k.[8] Then the DoG function can be calculated directly, due to the properties of convolution, as:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{2.8}$$

Local extrema of this function are taken as interest points. They are found by comparing each filtered pixel with its 26 neighbours in both space and adjacent levels in scale (see fig. 2.4).

It is worth noting that the DoG function provides a very close approximation to the scale-normalised Laplacian of Gaussian function $\sigma^2\nabla^2 G$, which tends to fire at blob-like structure in the image.

While highly efficient, this detector is only scale and rotationally invariant, and cannot cope with significant affine or perspective geometric changes. Additionally, it is fairly low-

[8] In scale space, a geometric sequence of Gaussian blurs corresponds to a linear sequence of information reduction. See [44].

resolution in the scale dimension, and due to initial smoothing cannot detect very fine detail.

2.1.5 The Harris-Laplace Detector

***Mikolajczyk & Schmid* [52]**

This is a precursor to the affine invariant interest point detector described in §2.1.6; their aim here is just scale and rotation invariance. Much of what we present builds on the ideas of scale-space presented in Appendix B.

Interest points are detected using the scale-adapted Harris corner detector:

$$F_{\text{Har}} = \det C - \alpha \text{trace}^2 C \tag{2.9}$$

where

$$C(\mathbf{x}, s_{\text{i}}, s_{\text{d}}) = s_d^2 \mathcal{G}(\mathbf{x}, s_{\text{i}}) * \left[(\nabla L(\mathbf{x}, s_{\text{d}}))(\nabla L(\mathbf{x}, s_{\text{d}}))^T \right] \tag{2.10}$$

$$= s_d^2 \mathcal{G}(\mathbf{x}, s_{\text{i}}) * \begin{bmatrix} L_x^2(\mathbf{x}, s_{\text{d}}) & L_x L_y(\mathbf{x}, s_{\text{d}}) \\ L_x L_y(\mathbf{x}, s_{\text{d}}) & L_y^2(\mathbf{x}, s_{\text{d}}) \end{bmatrix} \tag{2.11}$$

and α is some constant. A detection is defined where $F_{\text{Har}} > t_{\text{Har}}$ for some threshold t_{Har}.

They evaluate a number of functions to be used to find a *characteristic scale* for interest points. They suggest searching for extrema over scale of several scale normalised derivative-based functions, looking for good repeatability and therefore invariance of feature detection with respect to scale changes in the image.

The functions they evaluated are:

Square gradient: $F_{\text{Sq}} = s^2(L_x^2(\mathbf{x}, s) + L_y^2(\mathbf{x}, s))$

Laplacian: $F_{\text{Lap}} = s^2 |L_{xx}(\mathbf{x}, s) + L_{yy}(\mathbf{x}, s)|$

Difference-of-Gaussian: $F_{\text{DoG}} = |I(\mathbf{x}) * \mathcal{G}(s_{n-1}) - I(\mathbf{x}) * \mathcal{G}(s_n)|$

Harris function: F_{Har} (see eq. 2.9)

Given knowledge of the ground-truth scale change between images, it is simple to evaluate each of these for repeatability. Their results show that using the scale normalised Laplacian gave the best repeatability.

With this in mind, they propose a detection strategy as follows:

- Interest points are found with scale-adapted Harris corner detector by taking points in the image at different scales such that F_{Har} is greater than a threshold t_{Har} and at a local maximum with respect to its spatial 8-neighbourhood W. Formalising their notation slightly, we write the set of scale-adapted Harris points as $P_{\text{Har}} = \{(\mathbf{x}, s_n)\}$ such that:

$$F_{\text{Har}}(\mathbf{x}, s_n) > t_{\text{Har}} \tag{2.12}$$

$$F_{\text{Har}}(\mathbf{x}, s_n) > F_{\text{Har}}(\mathbf{x}', s_n), \forall \mathbf{x}' \in W \tag{2.13}$$

- In order to achieve invariance to scale they only take those Harris corners that occur at a sufficiently large maximum of the Laplacian in scale-space. We write $P_{\text{HL}} \subseteq P_{\text{Har}} =$ such that:

$$F_{\text{Lap}}(\mathbf{x}, s_n) > t_{\text{Lap}} \tag{2.14}$$

$$F_{\text{Lap}}(\mathbf{x}, s_n) > F_{\text{Lap}}(\mathbf{x}, s_{n-1}) \tag{2.15}$$

$$F_{\text{Lap}}(\mathbf{x}, s_n) > F_{\text{Lap}}(\mathbf{x}, s_{n+1}) \tag{2.16}$$

2.1.6 The Harris-Affine Detector

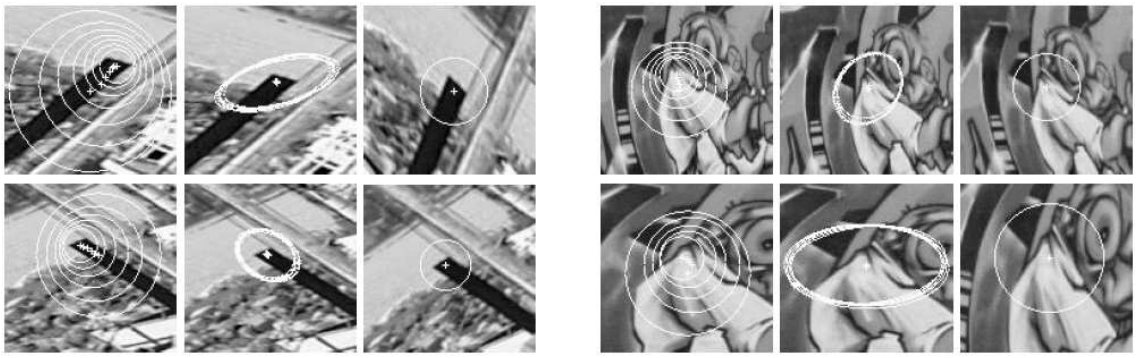
***Lindeberg & Gårding* [45, 46]**

***Baumberg* [3]**

***Mikolajczyk & Schmid* [53]**

This series of papers develops an affine geometric invariant interest point detector. The earliest paper by Lindeberg & Gårding ([45]) was in fact concerned with shape from texture, but forms the backbone of subsequent developments. They aim to estimate a transformation that has warped images of isotropic texture by iteratively adapting the estimate based on the second moment matrix, until convergence.[9] Baumberg in [3] was the first to exploit this technique to detect affine invariant image regions, by first detecting interest points, and then adapting their region of support with the iterative second moment matrix scheme until convergence. This was further generalised in the work of Mikolajczyk & Schmid in [53], which we present here in some detail. The reader is referred to Appendix B for some of the

[9] We have already encountered an isotropic version of the second moment matrix as $C(\mathbf{x}, s_{\text{i}}, s_{\text{d}})$.



(a) (b)

Figure 2.5: Two examples of the Harris-Affine interest point detector. In each, the top and bottom rows show the interest points detected from different views of the same surface. The left column shows the initial isotropic seed points found with the Harris-Laplace detector. The middle column shows the corresponding elliptical regions found once the iterative second moment matrix adaptation algorithm has converged. The right column shows the image normalised so that the ellipse becomes a circle. Note in both examples that the circles in the right column match up well up to rotation. [From [53]].

background mathematics.

They start by finding scale and rotation invariant interest points using the Harris-Laplace detector described in §2.1.5. These form seed interest points and are given an initial isotropic (circular) window representing no affine adaptation. They then iteratively adapt the position, size and most importantly *shape* of the local neighbourhood using the second moment matrix, until convergence at a fixed point, where affine geometric invariance is achieved. Their final result is a set of elliptical regions in an image which are invariant to affine changes, up to a rotation. Fig. 2.5 illustrates the process. There are then many techniques (see §2.2) for computing descriptor vectors which characterise these invariant regions for further invariance such as rotational and photometric invariance.

They restrict the seemingly huge search space by assuming good estimates of position and scale from the seed points, and also that the scale matrices are coupled (eq. B.29 and eq. B.30) such that

$$\Sigma_0 = \frac{1}{s_\mathrm{d}}\Sigma_\mathrm{d} = \frac{1}{s_\mathrm{i}}\Sigma_\mathrm{i} \tag{2.17}$$

for some common shape matrix Σ_0. This results in four degrees of freedom: two for Σ_0 (a symmetric 2×2 matrix normalised with respect to scale) and two for the relative scale parameters s_d and s_i.

The iterative algorithm starts from an isotropic shape matrix Σ_0 which is gradually trans-

formed until a fixed point is found. They work in the transformed image frame so that they can calculate the various derivative-based operators isotropically and therefore efficiently (by using separable filter kernels).

Each region in image $I(\mathbf{x})$ is specified for iteration τ by its centre $\mathbf{x}_c^{[\tau]}$ and its neighbourhood shape $U^{[\tau]}$. At each stage a local image patch $I_N^{[\tau]}$ is transformed and windowed as

$$I_N^{[\tau]}(\mathbf{x}') = W(\mathbf{x}')I((U^{[\tau-1]})^{-1}\mathbf{x}' + \mathbf{x}_c^{[\tau-1]}) \tag{2.18}$$

such that the transformed image patch has its centre at the origin. Here, $W(\mathbf{x}')$ is a windowing function such as the square window of sides $2S_W$:

$$W(\mathbf{x}') = \begin{cases} 1 & \text{if } |\mathbf{x}'| < S_W \\ 0 & \text{otherwise} \end{cases} \tag{2.19}$$

They define the cumulative transformation matrix as

$$U^{[\tau]} = (\mu^{[\tau]})^{\frac{1}{2}}(\mu^{[\tau-1]})^{\frac{1}{2}} \dots (\mu^{[1]})^{\frac{1}{2}}U^{[0]} \tag{2.20}$$

where the initial transformation $U^{[0]} = I$ and $(\mu^{[\tau]})^{\frac{1}{2}}$ (from the second moment matrix μ) is calculated in the local window frame $W^{[\tau]}$.[10] If at iteration τ_e in the algorithm $(\mu^{[\tau_e]})^{\frac{1}{2}} = I$ then this implies that $U^{[\tau_e]} = U^{[\tau_e-1]}$ and convergence to a fixed point of the second moment matrix has been achieved. Note that

$$M^{[\tau]} = (U^{[\tau]})(U^{[\tau]})^T \tag{2.21}$$

corresponds to M_L and M_R in §B.3.2.

At each stage in the algorithm it is natural to normalise the $U^{[\tau]}$ matrix with respect to scale so that s_d and s_i can be considered canonical with respect to this normalisation.[11] To do this, they examine the eigenvalues $\lambda_{max}(U^{[\tau]})$ and $\lambda_{min}(U^{[\tau]})$, and normalise $U^{[\tau]}$ such that $\lambda'_{min} = 1$ and $\lambda'_{max} = \frac{\lambda_{max}}{\lambda_{min}}$. This ensures that the original image is not under-sampled.

[10] Note that our formulation for the U matrix is the transpose inverse of the U matrix in [53]. The transpose is immaterial since we are dealing with square roots of matrices which can be defined in two fashions ($(M^{\frac{1}{2}})^T(M^{\frac{1}{2}}) = M$ or $(M^{\frac{1}{2}})(M^{\frac{1}{2}})^T = M$), and the inverse is used since it more clearly follows the derivations in Appendix B. In their paper they do not clearly specify the direction of the transformation U.

[11] This clearly has no effect on the existence of fixed points (eq. B.28).

To preserve invariance to scale variations in the image, they choose the new characteristic scale for the current estimate of the transformation. As with the Harris-Laplace detector, the Laplacian is used to select the integration scale $s_{\mathrm{i}}^{[\tau]}$ by searching for maxima with respect to scale in the transformed image frame. The differentiation scale is set proportionally:

$$s_{\mathrm{d}}^{[\tau]} = k^{[\tau]} s_{\mathrm{i}}^{[\tau]} \tag{2.22}$$

The factor $k^{[\tau]}$ should not be set too small or the Gaussian averaging would be too great with respect to the scale of differentiation; and it should not be too large otherwise $s_{\mathrm{i}}^{[\tau]}$ would not be able to average the derivatives well. They select the $k^{[\tau]}$ in the range $0.5 \leq k^{[\tau]} \leq 0.75$ such that the local isotropy assumes a maximum. This local isotropy can be measured as the eigenvalue ratio $\lambda_{min}(\mu^{[\tau]})/\lambda_{max}(\mu^{[\tau]})$ by calculating $\mu^{[\tau]}$ for different potential values of k. Convergence occurs at a fixed point where $\mu^{[\tau_\epsilon]} = I$, so maximising this ratio should increase the rate of convergence.

Changing the shape and scales with which the second moment matrix is calculated will slightly alter the spatial location of its local maxima. Consequently they suggest the need to re-detect spatial maxima in the transformed frame $I_N^{[\tau]}(\mathbf{x}')$, finding the local maximum $\mathbf{x}_c'^{[\tau]}$ nearest to the current position. This maximum is back transformed into the original image coordinate frame and so the new location of the point $\mathbf{x}_c^{[\tau]}$ can be found.

For convergence, they look for $(\mu^{[\tau]})^{\frac{1}{2}}$ being sufficiently close to a pure rotation, implying that the eigenvalues are sufficiently close:

$$\frac{\lambda_{\min}((\mu^{[\tau]})^{\frac{1}{2}})}{\lambda_{\max}((\mu^{[\tau]})^{\frac{1}{2}})} < \epsilon_{\mathrm{conv}} \tag{2.23}$$

for some constant ϵ_{conv} set to 0.96 in their paper.

They also stop the iteration and discard the result if the neighbourhood becomes too elongated:

$$\frac{\lambda_{\min}(U^{[\tau]})}{\lambda_{\max}(U^{[\tau]})} > \epsilon_{div} \tag{2.24}$$

Summary of the algorithm

For clarity we summarise the algorithm here. The seed points are taken as the points found by the Harris-Laplacian detector. Given an initial interest point $\mathbf{x}_c^{[0]}$ and an initial isotropic neighbourhood $U^{[0]} = I$, the algorithm proceeds as Algorithm 1.

Algorithm 1 Harris-Affine interest point detector

repeat

Calculate windowed image in transformed frame:

$$I_N^{[\tau]}(\mathbf{x}') \leftarrow W(\mathbf{x}')I((U^{[\tau-1]})^{-1}\mathbf{x}' + \mathbf{x}_c^{[\tau-1]}) \tag{2.25}$$

Select integration scale $s_{\mathrm{i}}^{[\tau]}$

Select differentiation scale $s_{\mathrm{d}}^{[\tau]}$

Re-detect spatial localisation $(\mathbf{x}'_c)^{[\tau]}$ closest to $(\mathbf{x}'_c)^{[\tau-1]}$, giving:

$$(\mathbf{x}_c)^{[\tau]} \leftarrow (U^{[\tau-1]})^{-1}(\mathbf{x}'_c)^{[\tau]} + (\mathbf{x}_c)^{[\tau-1]} \tag{2.26}$$

Compute:

$$\mu^{[\tau]} \leftarrow \mu_{I_N^{[\tau]}}((\mathbf{x}'_c)^{[\tau]}, s_{\mathrm{i}}, s_{\mathrm{d}}) \tag{2.27}$$

Concatenate transformation:

$$U^{[\tau]} \leftarrow (\mu^{[\tau]})^{\frac{1}{2}}U^{[\tau-1]} \tag{2.28}$$

Normalise $U^{[\tau]}$

until convergence

2.1.7 Miscellaneous

We briefly mention two other kinds of interest point detector. Firstly, those of Mikolajczyk et al presented in [55], which found interest points that lie on the edges of objects. These seem to be useful for thin, tubular like objects, such as bicycles and tennis racquets. Secondly, the idea of interest points has been extended to videos by Laptev & Lindeberg in [40], by simply treating time as a third dimension and using the 3-D equivalent of the second moment matrix.

2.1.8 Discussion and summary

This section has presented recent work into interest point detectors, focusing on those with affine geometric invariance properties. We have only described the interest point detectors, not any applications such as object detection and recognition, epipolar geometry estimation for wide baseline stereo, and image segmentation. These applications will be discussed in the coming sections of this chapter.

We have examined a wide variety of interest point detectors, each using different techniques and producing different results. The choice of which to use should be motivated by application. If efficiency is important then the DoG detector of Lowe ([48]) should be considered a good choice. However it does not address affine invariance and so more complex

detectors might be used. The simplest of these seems to be the MSE regions of Matas et al ([51]), and a more complicated scheme based on filter responses is described by Mikolajczyk & Schmid ([53]). Each of these schemes assumes rigid objects, but certain situations demand a degree of flexibility. Kadir & Brady's detector ([39]) proposes one method allowing for slight deformation and intra-class variability.

Given that each of these detectors exploits a different type of image feature it seems sensible to build (in the words of [74]) "an opportunistic system, that exploits a wide diversity of invariant regions depending on what is on offer". At least one such system has been built to great effect and is described in [67].

2.2 Local Descriptors

We discuss in this section the results from [54] by Mikolajczyk & Schmid, who evaluated a variety of local descriptors. These are ways of characterising the local neighbourhoods of interest points in various invariant manners (e.g. rotation and affine photometric) into a low dimensional vector that can be matched efficiently.

When creating local descriptors, it is important to bear in mind the scale of interest points to ensure sufficient discriminability. Smaller interest points are better localised and therefore less likely to cross an object boundary; but they are less discriminating and will match well to many other interest points, both correctly and incorrectly. Larger interest points have exactly the opposite characteristics: they are more discriminating but more likely to straddle an object boundary. Various people (such as [61]) have suggested therefore making local descriptors for various multiples of the scale found by the interest point detector.

Once a scale has been chosen, most interest point descriptors proceed by normalising the detected region to a circle of known radius (using, such as, bi-linear interpolation). For those descriptors that are not inherently rotationally invariant, the circular window can be rotated in the direction of the average gradient. For those descriptors that are not inherently photometrically invariant, the pixel values can be normalised by subtracting off the mean and dividing by the variance.

We will now briefly list the descriptors used for the evaluation of [54]:

- SIFT:

  This paper ([47]) was mentioned in §2.1.4 with regards to its interest point detector based on the DoG function. Here they characterise the local neighbourhood

by creating histograms of local gradient information, from which they form a 128-dimensional vector descriptor.

- Steerable Filters:

  These were suggested by Freeman in [28]. They look at the local image derivatives and steer them in the direction of local gradient, thereby giving rotation invariance.

- Differential Invariants:

  From e.g. [64]. They derive differential invariants, combining components of local derivatives (also known as the *local jet*) to obtain rotation invariance.

- Complex Filters:

  In e.g. [62] is presented a descriptor based on the outputs from a bank of complex filters of the form

  $$K(x, y, \theta) = f(x, y) \exp(i\theta) \tag{2.29}$$

  An image rotation will affect the phase of the output only, and so the modulus of the filter responses is a rotation invariant.

- Moment Invariants:

  Introduced in [76] by Van Gool et al. They define central moments as

  $$M^a_{p,q} = \int\int_\Omega x^p y^q [I(x, y)]^a \mathrm{d}x\mathrm{d}y \tag{2.30}$$

  with order $p + q$, degree a, and over the local neighbourhood Ω.

Their evaluation examines the above descriptors for their ability to cope with various geometric and photometric distortions. They conclude that the SIFT descriptors perform best, followed by Steerable Filters. However, given the much lower dimensionality of the latter (here, 13-dimensional compared with 128-dimensional), these should perhaps be considered a good choice.

2.3 Object Detection and Recognition

In this section we shall outline a few of the recent papers in the areas of object detection and object recognition. Detection usually is concerned with finding an instance of an object in an image, while recognition is used to determine the particular identity of an object. As techniques have improved, there has been a general shift over the last few years away from detection of individual objects to whole classes of objects, and lately much attention has been

focused on detecting deformable objects. There is also a general trend away from supervised techniques which use labeled training data to unsupervised techniques that can learn from a large collection of unlabeled images.

2.3.1 The Constellation Model

Burl et al **[14]**

Weber et al **[77]**

Fergus et al **[23, 24]**

Li et al **[20]**

A Probabilistic Approach to Object Recognition

The 'Constellation of Parts' model was introduced by Burl et al [14] for object detection. As with many if not all Computer Vision problems, their problem was to sensibly combine the detection of local features (here, 'parts') with some prior, high-level information about how the features fit together (here, the geometric 'constellation'). Some previous approaches (e.g. [13]) had involved a two stage process: first, features were detected in the image, and good candidates were chosen based on appearance alone; secondly these candidates were optimised sensibly incorporate the prior information.

However it is clear that this hard, two-stage decision strategy is non-optimal and ideally one should be able to optimise both the selection of parts and their arrangement *simultaneously*. This paper therefore derives a decision theoretic model for object detection and suggests three heuristic approaches to solve it. The technique is *supervised*, requiring hand-labeled part positions in the training images.

For a given image I, they want to detect the presence (hypothesis ω_1) or absence (hypothesis ω_2) of an object . Bayesian decision theory (see e.g. [5, 34, 19]) gives us a principled way of choosing between the two hypotheses, by examining the likelihood ratio of posterior probabilities:

$$\frac{p(\omega_1|I)}{p(\omega_2|I)} = \frac{p(I|\omega_1)p(\omega_1)}{p(I|\omega_2)p(\omega_2)} \tag{2.31}$$

Absorbing the ratio of priors into a constant threshold $\alpha = p(\omega_1)/p(\omega_2)$, one should examine the quantity

$$\Lambda = \frac{p(I|\omega_1)}{p(I|\omega_2)} \tag{2.32}$$

If Λ is greater than $1/\alpha$, then for the minimum Bayes error, hypothesis ω_1 should be chosen,

and conversely, if Λ is less than $1/\alpha$ then hypothesis ω_2 should be chosen.

They condition the numerator on the locations $\mathbf{X}$ of parts:

$$\Lambda = \frac{\sum_{\mathbf{X}} p(I|\mathbf{X},\omega_1)p(\mathbf{X}|\omega_1)}{p(I|\omega_2)} \tag{2.33}$$

and factorise the image into non-overlapping image regions representing the different parts $I^1,\ldots,I^N$ such that

$$\Lambda = \sum_{\mathbf{X}} \left[\prod_{i=1}^{N} \frac{p(I^i|\mathbf{X},\omega_1)}{p(I^i|\omega_2)}\right] p(\mathbf{X}|\omega_1) \tag{2.34}$$

$$= \sum_{\mathbf{X}} \left[\prod_{i=1}^{N} \lambda_i\right] p(\mathbf{X}|\omega_1) \tag{2.35}$$

They present straightforward expressions for the likelihood $p(I|\mathbf{X},\omega_1)$ under certain Gaussian noise assumptions. However, to model the probability distribution of positions $p(\mathbf{X}|\omega_1)$ as a full joint distribution (i.e. non-independent part positions) would result in a combinatorial explosion in evaluating the sum. Hence this paper assumes that all the probability mass is located at a point $\mathbf{X}_0$, and so

$$\Lambda \approx \Lambda_0 = \prod_{i=1}^{N} \lambda_i p(\mathbf{X}_0) \tag{2.36}$$

They present three heuristics for optimising the selection and location of parts, and demonstrate their 'soft'-decision model working better than the hard-decision models in previous work.

Unsupervised Learning of Models for Recognition

The next development of the model was in [77] by Weber et al. The major restriction in the previous paper was the requirement for hand-labeled training data. This paper presents a method for learning in an *unsupervised* manner that does not require any hand-labeling. The idea is to detect a large number of parts throughout the training data and select a subset of these that is sufficiently representative to reliably detect the class of object in question.

To this end, they first select a large number (~ 10000) of interest points from the training

images.[12] Using the standard k-means clustering algorithm [5], they vector-quantise the interest points descriptors to a much smaller set of features (~ 100). The task is then to select a subset of these features to correspond to the 'parts' in the constellation model, and simultaneously to estimate the parameters underlying the probability distributions of the model.

They employ a greedy strategy to find a non-optimal but hopefully good set of parts, and for each set tested, EM is employed to learn the parameters, for which the update equations are derived in the paper.[13]

Object Class Recognition by Unsupervised Scale-Invariant Learning

Another improvement to the constellation model, [23] adds some degree of scale invariance, and additionally models the variability of appearance, learned simultaneously with the other parameters of the model. They illustrate the technique working with six object categories varying from faces to spotted cats.[14]

In their formulation, parts consist of an appearance, a relative scale, and an occlusion flag. The global shape is represented by the mutual positions of the parts. They first run an computationally-intensive learning stage to estimate the parameters of the model, and then use these parameters for recognition which is much faster.

The number of parts P ($\sim 6-7$) and number of features N ($\sim 20-30$) are specified by hand. The features are detected using the Kadir & Brady interest point detector (see §2.1.3), resulting in locations X, scales S, and appearances A for which principle components ([5]) are used for dimensionality reduction. They denote the complete set of parameters for the model by θ.

[12] They use the Förstner interest point detector of [27].
[13] EM stands for Expectation Maximisation; see e.g. [5, 18].
[14] Their test data are available at `http://www.robots.ox.ac.uk/~vgg/data/`.

As before, a Bayesian decision is made, here R:

$$\begin{aligned} R &= \frac{p(\text{Object}|X,S,A)}{p(\text{No object}|X,S,A)} && (2.37)\\ &= \alpha\frac{p(X,S,A|\text{Object})}{p(X,S,A|\text{No object})} && (2.38)\\ &= \alpha\frac{\int_\theta p(X,S,A|\theta,\text{Object})p(\theta|\text{Object})\mathrm{d}\theta}{\int_\theta p(X,S,A|\theta,\text{No object})p(\theta|\text{No object})\mathrm{d}\theta} && (2.39)\\ &\approx \alpha\frac{p(X,S,A|\theta_{ML})}{p(X,S,A|\theta_{bg})} && (2.40) \end{aligned}$$

where α is the ratio of priors as before, and the approximation in the last line is down to taking the maximum-likelihood point estimate θ_{ML}, as given by EM.

Both the numerator and denominator can then be factored using:

$$p(X,S,A|\theta) = \sum_{h\in H} p(A|X,S,h,\theta)p(X|S,h,\theta)p(S|h,\theta)p(h|\theta) \quad (2.41)$$

Here, h represents a 'hypothesis', an allocation of features detected in the image to parts in the model. This is the most limiting feature in the model since there are $O(N^P)$ such hypotheses, each of which must be summed over. This severely restricts the numbers N and P, especially P the number of parts.

Each of the densities in the above expression is given a parametric form (e.g. Gaussians with means and covariances) to fully specify the model. They employ an EM learning algorithm which due to the large number of hypotheses, takes well over 24 hours to complete despite seemingly small values of N and P. But once the model parameters are learned (which is done independently for each category), recognition can be effected in a few seconds.

A Bayesian Approach to Unsupervised One-Shot Learning of Object Categories

This paper presents a method that aims to incorporate *generic* prior knowledge into the task of learning object categories with the constellation model. The goal is 'One-Shot' learning: only one or very few training images are needed to learn to recognise a new category of objects. Their motivation is that humans can learn to recognise new classes very easily with few training examples, and that to do this we must be using some concept of 'generic' knowledge about objects.

They proceed as follows. First several categories are learned using a slightly simplified

version of the existing technique of [23]. They specify a parametric form for the prior distribution over parameters of shape and appearance, and aim to learn the hyperparameters for this distribution. To do this, they employ a Variational approximation (see e.g. [36]), and use Variational Bayesian EM to fit the prior distribution to the example learned parameters. By construction, the Bayesian decision integral has a closed form solution and hence can be evaluated for an object present/absent classification.

They evaluate their technique over four classes, learning three as in [23], and the fourth with the new one-shot method. Their results seem impressive for the small amount of training data. However it is not clear exactly what generic knowledge they are actually learning; the example priors given seem very weak and the shape prior seems only to indicate some notion of compactness. Whether the method would generalise to different classes and more classes simultaneously is yet to be seen.

A Visual Category Filter for Google Images

In [24], the most recent paper to use the constellation model, they present a sensible application for the object recognition techniques presented above. The Google image search allows the user to search for images using words or phrases.[15] It works by finding matching text near to images on webpages, and hence is very unreliable: in addition to many good matches are many bad results. The problem they try to solve here is to effectively filter the results from Google by re-sorting them from most likely correct to least likely. To do this they must learn an object category model, for which they use an improved version of [23].

As before, they employ the interest point detector of Kadir & Brady [38], but additionally incorporate curve segments by finding Canny edges ([15]) and locating the bi-tangent points. They require either that the user selects a few good example images from the image Google results, or that a RANSAC-like algorithm is run to ensure robust learning amongst the very cluttered input data.

Having learned (in a similar manner to [23]) a model for the object class being searched for, they can re-rank the images based on the likelihood obtained from the model.

[15] `http://www.google.com/imghp`.

2.3.2 Miscellaneous

We briefly mention other papers of note for completeness. Agarwal & Roth in [1] detect interest points and learn a classifier to detect instances of an object class. Schneiderman & Kanade in [65] describe a trainable object detector that can detect faces and cars at any size, location or pose, based on a classifier that uses the statistics of parts derived from wavelet responses in the image. Felzenszwalb & Huttenlocher in [21] describe Pictorial Structures, which model a collection of parts in a deformable configuration.

2.4 Segmentation

As described in §1.2 the traditional approach to segmentation has been a bottom-up one, relying on low-level image cues such as colour, texture, and edges. Example bottom-up segmentation papers include [66, 50, 11, 58]. A recent paper by Ferrari et al in [25] present a combined method of object recognition and by 'image exploration'; they can find and segment *particular* objects in extremely different poses and deformations. Borenstein & Ullman in [9, 10, 8] present work on class-specific segmentation, and this will be investigated in detail in Chapter 3. Tu et al in [73] present a method based on data-driven Markov-chain Monte Carlo (DDMCMC) which parses an image by segmenting it into regions and classifying certain regions as face or text.

2.4.1 Combined Object Categorization and Segmentation

Leibe et al [41, 42, 43]

In [42], they introduce a new method which interleaves object recognition and class-specific segmentation. Clearly these two tasks are intimately linked and combining them is a worthy goal. Their paper [41] includes an improved evaluation, and [43] extends the technique to cope with scale invariance.

As with several class-specific segmentation techniques, this is a supervised technique, requiring training data consisting of images containing an object of the relevant class paired with the corresponding figure-ground segmentation. From this they aim to learn a model of appearance and shape which can be used to generalise to novel examples from the same class.

To build the codebook of local appearance, they start by finding Harris corners ([33]), around which they extract image patches of size 25×25 pixels. These appearances are clus-

tered with an agglomerative clustering algorithm, such that the two most similar clusters are repeatedly merged, using normalised cross correlation as their difference measure (see Appendix C). The centres of the clusters form the appearance codebook. The main problem with this approach is that their appearances will often straddle the object boundary, and in these cases they are learning not just the object appearance but also the background appearance. Additionally, using the Harris detector only finds a small subset of image regions that could be useful for recognition and segmentation. Next, they create an implicit shape model for the object class. For each codebook entry, they locate instances in the training images to build a model of position relative to the object centre of mass.

From the codebook and shape model, they proceed with recognition as follows. Interest points are detected in the novel image, and matching codebook entries are found. The shape model for each matched entry is used to vote for the object centre, and a generalised Hough transform (e.g. [2]) is applied to find (possibly several) hypotheses for location. These are tested in turn by looking at the surrounding areas of the image not previously covered by the interest point detector. A final minimum description length (MDL) step improves results by ensuring that the final hypothesis provides the best description of the image.

They evaluate their approach on two data sets: cars and cows. Their claim recognition results better than existing techniques (e.g. [23]) for the cars, and demonstrate their technique giving good segmentations for the cow test set. However, this approach has a number of drawbacks. Primary is it that actually learns some background appearance as a by-product of the interest point detector. Since an instance of a class could appear in almost any environment, ideally the background should be treated completely independently from the foreground. Another limitation is that the segmentation achieved lies only on and around the object in question, while other areas in the background are marked as 'unknown'.

2.5 Miscellaneous

2.5.1 Epitome Images

***Frey & Jojic* [35]**

We briefly mention this paper that presents a novel image representation called 'epitome' images. They specify a generative model of an image in terms of a much smaller epitome image and a mapping from this back into the image. These are learned using a variational version of EM (Expectation Maximisation). They claim their technique is useful for layer

segmentation and for building databases of patches where it compares favourably with standard clustering techniques in terms of space and time taken to build. It certainly seems to provide translation invariance automatically.

2.5.2 Super-Resolution

***Freeman et al* [29]**

This work on super-resolution aims to reconstruct a high-resolution (hi-res) image from a low-resolution (lo-res) image. As with many Vision problems, this is ill-posed and therefore impossible to do perfectly for an arbitrary image, since there are many hi-res images that down-sample to the same lo-res image. However, for a class of images with similar prevalent textures, some resolution enhancement should be possible; one must make use of *prior* information, here in the form of prevalent textures. They aim to learn these textures by constructing a database of hi-res image patches from a training corpus, with corresponding (down-sampled) lo-res patches.

They divide a given lo-res test image into a regular grid, and for each grid square, select a set of good matches from the lo-res patch database. From the database construction, they now have a corresponding set of hi-res patches that could have generated the observed lo-res image patches.

However, these hypotheses about the underlying hi-res image are local and hence somewhat ambiguous, and so they must choose from amongst the hypotheses those that produce a globally consistent hi-res image. To do this, they form a pairwise Markov Random Field ([31]) over the grid, specify forms for the evidence and consistency functions, and use the Belief Propagation algorithm ([57]) to perform inference. This results in consistent hi-res patches which are stitched together to produce the super-resolution image.

We shall employ a similar technique in Chapter 4 to choose a globally consistent figure-ground segmentation from local hypotheses.

CHAPTER 3

BORENSTEIN & ULLMAN: FRAGMENT-BASED APPROACHES

3.1 Overview

Borenstein & Ullman have presented two very similar fragment based approaches to top-down class-specific figure-ground segmentation [9, 10].[1] Both papers follow a similar methodology (see fig. 3.1), though they differ slightly in the details. The general idea is to apply learned shape characteristics of objects within the class to guide the segmentation process of novel images. Both require a set of training images of both class images and non-class images.[2] The first step is to extract fragments from the class images such that they will hopefully generalise well to novel images. Next, each of these fragments must be segmented into regions of foreground and background. Finally these fragments are detected in a novel test image and arranged in such a way as to create a consistent figure-ground segmentation.

The main difference between the papers is in the second, fragment segmentation step, where their earlier work([9], which we will refer to as 'Borenstein & Ullman 2002') requires hand-segmented mask images. Their more recent paper ([10], which we will refer to as 'Borenstein & Ullman 2004') can automatically estimate the figure-ground segmentations for each fragment.

In this chapter we will first describe their methods in detail, going over each step indi-

[1] In this document we use the terms 'fragment' and 'patch' interchangeably to mean a small rectangular region of image.

[2] 'Class' images are those that contain an instance of the class, whereas 'non-class' images do not. Sometimes non-class images are also known as background images; we choose to avoid this terminology due to the obvious confusion with image backgrounds. Appendix A shows the training and validation data that we used to evaluate their techniques; it very closely matches the data they used for their evaluation.

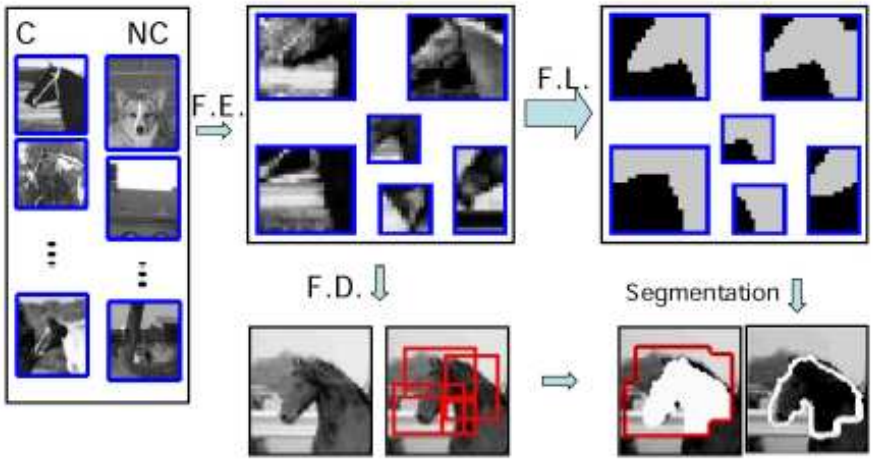


Figure 3.1: General methodology of [9, 10]. From a set of class (C) and non-class (NC) training images, a set of reliable fragments are extracted (F.E.). The figure-ground segmentation mask for each fragment must be learned (F.L.). Then instances of the fragments are detected (F.D.) in novel test images. These instances with their masks together produce a figure-ground segmentation for the whole image. [Figure from [10]].

vidually in §3.2, §3.3, and §3.4. In places we shall tidy up their notation somewhat. In §3.5 we give our evaluation of their methods, and in §3.6 we present our conclusions.

3.2 Fragment Extraction

They aim to extract image fragments from the training set which are later to be used to tile novel test images. They need to be capable of delineating the boundaries of objects in the class, as well as covering their interiors. The best fragments can be chosen by looking for those that are strongly correlated with the class images but not with the non-class images. They must be highly overlapping, well-distributed across the class objects, and such that all foreground parts of the training images can be covered.

Denoting the training set of class images,

$$\mathcal{C} = \{c_1, \ldots, c_{M_\mathcal{C}}\} \tag{3.1}$$

and the training set of non-class images,

$$\mathcal{B} = \{b_1, \ldots, b_{M_\mathcal{B}}\} \tag{3.2}$$

they generate a large number of candidate fragments at random:

$$\mathcal{G} = \{g_1, \ldots, g_K\} \tag{3.3}$$

Each candidate fragment is a small rectangular image region with a random size and aspect-ratio, taken from a random position in a random image in $\mathcal{C}$.

Each candidate fragment g_i is matched (the matching measure is described below) against each image in the training set. We write

$$S(g_i, I) \tag{3.4}$$

to represent the best matching score between g_i and image I, where I ranges over the sets of training images:

$$I \in \{c_1, \ldots, c_{M_\mathcal{C}}, b_1, \ldots, b_{M_\mathcal{B}}\} \tag{3.5}$$

Each fragment g_i is assigned a detection threshold θ_i, i.e. the fragment is detected in image I if $S(g_i, I) \geq \theta_i$ and from this can be defined a variable to indicate the detection of the fragment in the image:

$$\zeta(g_i, I) = \begin{cases} 1 & \text{if } S(g_i, I) \geq \theta_i \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

These definitions are used in Appendix D to define detection probabilities and mutual information of detection that they use.

Borenstein & Ullman 2002

The matching score $S(g, I)$ between a fragment g and an image I is defined in this paper as the maximum[3] value over the image of the normalised cross correlation (NCC), described in detail in Appendix C, between the fragment and the image:

$$S(g, I) = \max_{(x,y)} \mathrm{NCC}_{(x,y)}(g, I) \tag{3.7}$$

[3] A disadvantage of using the *maximum* value is that only one detection ('hit') is possible in an image per fragment. Clearly there are certain fragments (e.g. horse legs) that could match well in several regions of the image simultaneously, but this possibility is ignored here.

To reach a fixed false alarms rate α, they choose θ_i for fragment g_i such that

$$p(X_{g_i} = 1|\mathcal{B}) \leq \alpha \tag{3.8}$$

which consequently fixes the hit rate $p(X_{g_i} = 1|\mathcal{C})$ as well. They order the fragments by their hit rate and select the N best to form the fragment set $\mathcal{F} = \{f_1, \ldots, f_N\}$ which is used in later stages of the algorithm. They additionally assign a measure of 'reliability' to each fragment based on the likelihood ratio L_{f_i} between the detection rate and the false alarm rate.

The size of $\mathcal{F}$ is chosen to be large enough that class representation is over-complete, giving many overlapping fragments which can cover all the training images.

Borenstein & Ullman 2004

They follow the very similar technique of Ullman et al [75]. The correlation score $S(g, I)$ between a fragment g and an image I is defined in this paper as the maximum value over the image of the absolute[4] NCC between the fragment and the image:

$$S(g, I) = \max_{(x,y)} |\text{NCC}_{(x,y)}(g, I)| \tag{3.9}$$

They selects θ_i for fragment g_i such that the mutual information $I(X_{g_i}; Y)$ between the fragment detection and the class is maximised.

They add candidate fragments to the fragment set $\mathcal{F}$ one-by-one so as to maximise the gain in mutual information between the fragment set and the class:

$$f = \arg\max_{g_i}(I(\mathcal{F} \cup g_i; Y) - I(\mathcal{F}; Y)) \tag{3.10}$$

3.3 Fragment Segmentation Learning

Given the set $\mathcal{F} = \{f_1, \ldots, f_N\}$ of fragments, it is necessary to create a corresponding set of binary labeled figure-ground segmentation masks $\mathcal{M} = \{m_1, \ldots, m_N\}$.

[4]They do not specify why the absolute value is taken: we assume that they wish white horses to be able to match black horses. While this may be sensible for this class of images, for others it may be that only particular intensities are valid, and so we suggest that the training data rather than the matching score should be made representative of grey-scale variability.

Borenstein & Ullman 2002

Here they rely on hand-segmented training data. Given segmentations of the original class images, it is trivial to extract $\mathcal{M}$ given knowledge of where the fragment was randomly selected.

Borenstein & Ullman 2004

The main novel contribution of this paper is to automate the figure-ground labeling of the extracted fragments. Their method relies on two criteria, the 'degree of cover', related to the variability of the background, and the 'border consistency', which looks for edges that consistently occur in the training images where the fragment is detected. The algorithm proceeds as follows:

- Initialisation:

  For efficiency reasons, they run an initialisation stage to get a rough bottom-up segmentation which divides each fragment into a small number of regions ($D \approx 9$). The hope is that assigning each region a figure-ground label will be sufficient to accurately segment the whole fragment. This greatly simplifies the computational cost of finding an optimal labeling (see below), since there are now only 2^D combinations of foreground/background allocation.

- Degree of cover:

  Given the over-complete fragment set representing the class, by detecting instances of the fragments in a training image, each region of a particular fragment should be covered by several other overlapping fragments. The insight is that the more fragments that cover a region, the more likely that that region is part of the foreground (see fig. 3.2), because the fragment extraction method chooses fragments that are more likely to be detected in foreground areas than background areas.

  For a fragment f we write the regions into which it is decomposed as $\mathcal{R} = \{r_1, \ldots, r_D\}$. The value $\bar{r}_j, j = 1, \ldots, D$ denotes the degree of cover and can be calculated by counting the number of fragments overlapping with region r_j over all the class images in the training set. They compute $\bar{r}_j$ for the D regions in the fragment, and label as foreground those regions for which $\bar{r}_j \geq \kappa_f$ for some threshold κ_f:

$$Q_f(\kappa_f) = \bigcup_{\{j:\bar{r}_j \geq \kappa_f\}} r_j \tag{3.11}$$

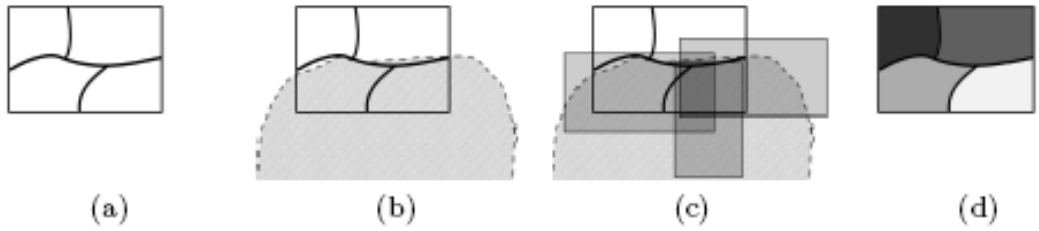


Figure 3.2: Degree of cover. (a) A fragment is split into regions. (b) The fragment is detected in the test image. (c) Other fragments are detected overlapping in the image. (d) The likelihood of foreground for each region can be estimated from the number of fragments overlapping that region. [Figure from [10]].

By sorting the regions by the degree of cover $\bar{r}_j$, they effectively reduce the number of combinations of segmentations from $O(2^n)$ to $O(n)$.

- Border consistency:

  The degree of cover is useful but apparently not sufficient for an accurate segmentation boundary, and they still need a method of choosing the threshold $\bar{r}$. For each fragment f they find the set of image patches where the fragment has been detected which they call 'fragment hits' denoted $\mathcal{H}_f = \{h_1, \ldots, h_{k_f}\}$. They apply an edge detector to each of the hits and obtain the consistent edges of a fragment by averaging:

  $$D_f = \frac{1}{k_f} \sum_{j=1}^{k_f} \text{edge}(h_j) \tag{3.12}$$

  where the sum is pixel-wise over the fragment hits.

  They define three types of edges: *noise edges* which should be averaged out, consistent *border edges* which correctly lie along the edges of the foreground/background borders in the fragment, and consistent *interior edges* that are edges in the foreground. The latter two types of edges are consistent in the sense that they are present in many of the hits h_j.

- Figure-ground optimisation:

  Given the figure-ground labeling $Q_f(\kappa_f)$ for fragment f as a function of threshold, and the border consistency map D_f, they want to find an optimal figure-ground segmentation Q_f by choosing the best value of κ_f. The figure-ground boundary is denoted as ∂Q. All consistent edges should lie within the figure (interior edges) or on the boundary ∂Q.

They therefore maximise the following:

$$Q_f = \arg\max_{Q_f(\kappa_f)} \left(\sum_{(x,y)\in Q_f(\kappa_f)} D_f(x,y) + \lambda \sum_{(x,y)\in \partial Q_f(\kappa_f)} D_f(x,y) \right) \tag{3.13}$$

The first term is maximised when the foreground part of the fragment contains as many of the consistent edges as possible, the second term when the boundary is supported by consistent edges. The parameter λ controls the relative weighting of the two terms.

Since there are only $D \approx 9$ choices of κ_f, this is an extremely simple and efficient optimisation. It is not clear given the linear cost, why then they run the optimisation over regions rather than on a per-pixel basis.

This results in a reasonably accurate learned figure-ground segmentation mask for each fragment. They return later in their paper to present an iterative algorithm for improving these fragment segmentations. They run their segmentation algorithm for all the training images, and use the outputs to estimate more accurately the degree of cover for each region in each fragment, relaxing the regions to individual pixels. This iterative process was shown to have improved the results and they report it converged to a stable state within 3 iterations.

3.4 Image Segmentation Algorithm

Given the fragments and their figure-ground segmentations learned as in the previous section, the task is to cover a novel image I with these fragments and thereby to generate a figure-ground segmentation as output. The two papers present considerably different methods for this as explained below.

Borenstein & Ullman 2002

They denote a cover W as the set of detected fragments with their corresponding positions $p = (x, y)$. For a cover W they can compute its quality as a function of

- Individual Match Score:

  Their measure combines NCC with an edge detection measure. Given the figure-ground mask m_i corresponding to fragment f_i detected at position p, they can exclude background pixels from the correlation measure. To compensate for the consequent loss of edge information, they add an explicit edge similarity measure. The

whole match score can be written:

$$s_p(f, m, I) = w\text{NCC}_p(f, I)|_m + (1 - w)\text{EDGE}_p(m, I) \tag{3.14}$$

It is not stated in the paper which type of edge detector or indeed what similarity measure they use for the edge maps.

- Consistency:

  This is effectively a smoothness prior to ensure that edges match up well. For two fragment masks m_a and m_b at positions p_a and p_b respectively, they define a symmetric consistency function $c(m_a, p_a, m_b, p_b)$ as:

$$c(m_a, p_a, m_b, p_b) = \frac{\text{Number Consistent Overlapping Mask Labels}}{\text{Total Mask Overlap Size}} \tag{3.15}$$

  with sensible checks for small overlap sizes so that if the overlap is too small it will not have too large a contribution.

- Reliability:

  They suggest that the task of piecing together the fragments can be simplified by selecting more reliable fragments for the cover before less reliable ones, similarly to how we might start piecing together a jigsaw puzzle with the more obvious (highly-textured) pieces.

They describe a fairly ad-hoc algorithm to optimise the quality of the cover of the test image. They calculate the total cover score for a particular cover W of image I as:

$$C(W, I) = \sum_{a \in W} [\gamma(a, I) + \frac{1}{\mu} \sum_{b \in W, b \neq a} \beta(a, b, I)] \tag{3.16}$$

where μ is the relative importance of the smoothing versus the data, $\gamma(a, I)$ represents the 'data' term, and $\beta(a, b, I)$ the 'smoothness' term:[5]

$$\gamma(a, I) = L_{f_a} . s_{p_a}(f_a, m_a, I) \tag{3.17}$$

[5]Note that $\beta(a, b, I)$ depends on the image data I, and should not be considered as exploiting purely prior information.

where L_{f_a} is the reliability of fragment f_a (given by the likelihood ratio of detection; see Appendix D). Also

$$\beta(a,b,I) = (c(m_a,p_a,m_b,p_b) - \beta)(L_{f_a}.s_{p_a}(f_a,m_a,I) + L_{f_b}.s_{p_b}(f_b,m_b,I)) \tag{3.18}$$

where β is a constant penalty cost for inconsistent overlap.

Their algorithm is iterative and greedy, picking at each step a subset fragments that will improve the cover score, and removing fragments from the cover which lower the score. It is guaranteed to converge to a local maximum since the score is bounded and increases at each step.

Borenstein & Ullman 2004

They present much simpler algorithm which their results indicate works almost as well as the complex algorithm described above. Fragment hits h_i are detected in the novel image. Since there are likely to be several fragment hits covering each pixel, they use a *voting* scheme based on the labeling of the fragments. Areas of fragment hits labeled foreground in the corresponding learned fragment mask m_i vote positively, and areas labeled background vote negatively. If the score at pixel (x,y) is greater than zero then it is considered foreground, otherwise background.

$$V(x,y) = \begin{cases} +1 & \text{if } \sum_{\{h_i\}} L_{h_i} m_i(x,y) > 0 \\ -1 & \text{if } \sum_{\{h_i\}} L_{h_i} m_i(x,y) \leq 0 \end{cases} \tag{3.19}$$

where L_{h_i} represents the class-specificity of the fragment hit, and $m_i(x,y) \in \{-1,+1\}$ is the binary labeled figure-ground segmentation mask of fragment f_i, centred on the image where the fragment hit was detected.

A post-processing step removes those fragments from the voting that are deemed inconsistent with the segmentation output and recalculates the votes on the reduced subset of fragment hits.

3.5 Evaluation of their methods

In this section we shall first present the evaluations they performed on the two methods. Then we shall present our own findings based on our implementations.

Borenstein & Ullman 2002

They evaluate their algorithm on images of horses viewed side-on. They use a training set of 41 hand-segmented class images and 253 non-class images. All their experiments are performed at 40×30 pixels resolution, and they use only grey-scale images, ignoring colour information. They generate 146 reliable and 339 non-reliable fragments, and tested the algorithm on 176 novel horse images.

They present 21 examples of their output, all of which look reasonable, but there is often considerable error, especially in the detail of the legs and the tail. Their quantitative evaluation is limited to a raw score based on the average consistency between their results and ground-truth: they evaluated the ratio of areas $r = \frac{|S \cap F|}{|S \cup F|}$ where F is the ground-truth segmentation and S is the segmentation output of the algorithm. The average error stated was $\bar{r} = 0.71$, compared to 0.31 for a bottom-up segmentation algorithm ([66]). They give timings of their algorithm on a 600MHz Pentium of about 40 seconds per target image, though learning the reliable fragments apparently takes days.

Borenstein & Ullman 2004

They test their algorithm on three classes: horse heads viewed side-on, cars viewed side-on, and human faces viewed front-on.[6] They compared the hand segmentations used in the old method, against the new method for learning the fragment segmentations which performed nearly as well, though there was still some considerable inaccuracy.

They conclude that a pure top-down segmentation approach will struggle with high resolution detail (e.g. ears), especially given a small training set, whereas a pure bottom-up approach will delineate boundaries well but struggle to coherently group regions together. Perhaps therefore a combined top-down, bottom-up approach is needed (as in e.g. [73, 8]).

3.5.1 Our Experiments

We have run experiments looking at the method presented in Borenstein & Ullman 2002 in detail, and the method in Borenstein & Ullman 2004 apart from the (significant) part that learns fragment segmentations. In other words we learn from a hand-segmented training

[6] It is interesting that they do not apply the method to the class of horse images that was used in their previous paper; perhaps they were unable to get good results.

set, and attempt to produce a good segmentation for novel test images. We have been unable to produce their results in a convincing fashion.

Experiment A: Building the fragment set

The first, and crucial, step in both papers is to extract from the training sets (class and non-class images) a set of reliable image fragments to be used to segment the test images. Here we present our results for their two methods. Both begin by selecting random candidate fragments from the class images, followed by performing a NCC between each fragment and training image (for both class and non-class images). Their first paper ranks them by likelihood ratio of detection, class to non-class, the second by a mutual information measure between detection and class. For this test we use 1000 candidate fragments at random from the database of 40 hand-segmented training images.

Fig. 3.3 and fig. 3.4 respectively show the results obtained from the methods of Borenstein & Ullman 2002 and Borenstein & Ullman 2004. In each, (a) shows the best candidate fragments while (b) shows the worst candidate fragments.

There are several interesting things to note:

- There is a lot of intra-horse similarity. For example, horse front and hind legs look very similar and consequently get matched together. There is also some matching between wildly different parts of horse, e.g. head to tail. This is not necessarily a problem in this model, since there is no notion of global shape to confuse. However more global methods might struggle.

- At least one of the very high ranking fragments has no areas of foreground. This is probably a short-coming of the training data rather than the method: most of the class images have grass and fields in the background but very few of the non-class images. One should additionally exploit the knowledge of backgrounds of the class images to help with training.

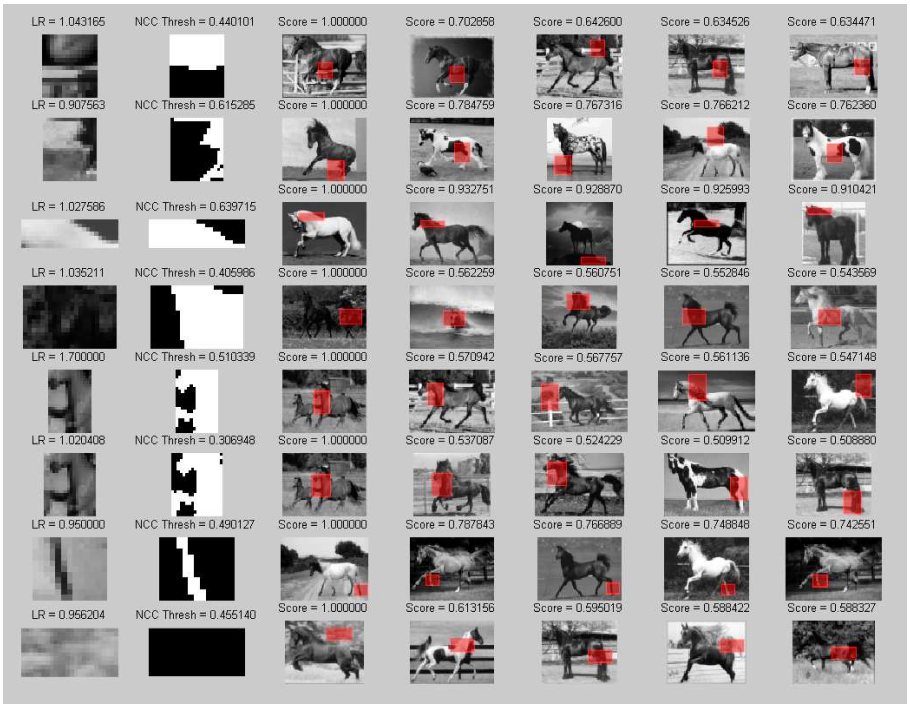
- The use of the *absolute* NCC in Borenstein & Ullman 2004 produces some unfortunate effects (e.g. the 4th fragment hit on the bottom line of fig. 3.4a). As mentioned previously, we believe that matching black to white indiscriminately is probably not a good thing. However, similar effects will also occur due to the inherent ambiguity of local appearance, which we will investigate more fully in §4.2.

(a)

(b)

Figure 3.3: Experiment A, using the method of Borenstein & Ullman 2002. (a) The eight candidates with highest likelihood ratios. (b) The eight candidates with lowest likelihood ratios. In both, the first column is the intensity pattern of the fragment, above which is given its likelihood ratio. The second column shows the fragment mask, above which is given the detection threshold. The final five columns show the five highest scoring fragment hits, highlighted in red, within the class images. Note the first hit is always the fragment in its original position.

(a)



(b)

Figure 3.4: Experiment A, using the method of Borenstein & Ullman 2004. (a) The eight candidates with greatest mutual information. (b) The eight candidates with lowest mutual information. In both, the first column is the intensity pattern of the fragment, above which is given its likelihood ratio. The second column shows the fragment mask, above which is given the detection threshold. The final five columns show the five highest scoring fragment hits, highlighted in red, within the class images. Note the first hit is always the fragment in its original position.

Experiment B: Testing the segmentation algorithms

We were unable to get good results from the method of either paper. Examples are shown in fig. 3.5 of our segmentations obtained on novel validation images. These were the *best* results we could obtain and hence we did not evaluate the whole validation set.

- Borenstein & Ullman 2002:

  Their algorithm seems needlessly complex, and various crucial implementation details (values of parameters, how to generate a final segmentation, how many fragments to use, etc.) seem to be missing from the paper and had to be guessed at. The results we show are from a set of 150 fragments. We did leave out a few minor parts of their algorithm such the multi-scale fragments (as they had barely explained these at all), but these should not be crucial to the method.

- Borenstein & Ullman 2004:

  The second algorithm seems on the other hand almost too simple, and perhaps therefore did not produce good results in our tests. The test data for their evaluation only included simple shapes (horse heads, cars, and faces), and so perhaps the method breaks given a more complex shape such as a full horse used in our evaluation. Our results are only relevant for the last part of the paper, and we have not attempted to evaluate their fragment segmentation learning phase, the main contribution of the paper; we wanted simply to compare the two segmentation algorithms given labeled training data.

3.6 Conclusions

Borenstein & Ullman 2002

The resolution of their test data seems far too low to be of practical use, though could be a starting point for a further segmentation at higher resolution. Their optimisation algorithm seems to be rather unusual and it is unclear and unexplained why a much simpler algorithm was not used; Borenstein & Ullman 2004 produced results that seemed qualitatively no worse.

Our implementation of their work did not produce good output; whether this was because we used different parameter settings, or higher resolution images (80×60 pixels), or because they had missed out important information, we are unsure.

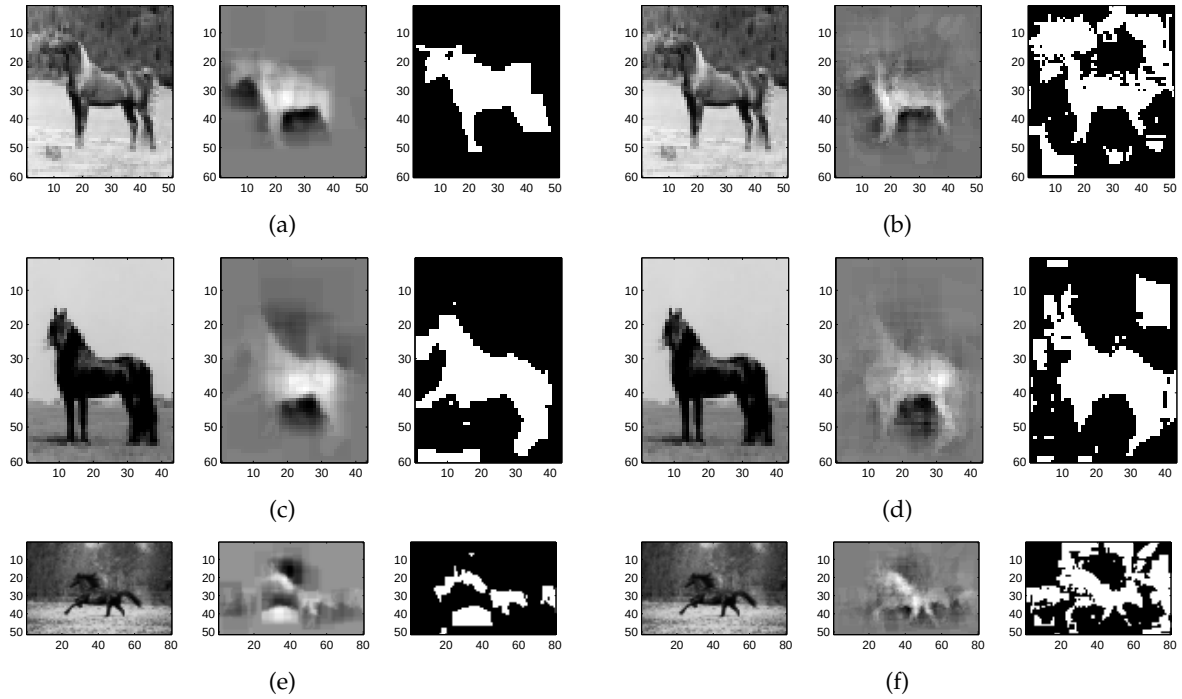


Figure 3.5: Results from Experiment B. (a), (c), (e): results from our implementation of Borenstein & Ullman 2002. (b), (d), (f): results from our implementation of Borenstein & Ullman 2004. In each example, the left image is the input to the algorithm. The middle image is the sum (positive contributions from foreground, negative from background) of all the fragment masks used to cover the image. The right image is an illustration of how one might threshold the middle image to generate a binary valued segmentation.

The paper seems to be missing several crucial implementation issues such as:

- How to actually compute a complete figure-ground segmentation mask from the cover W - should you just pixel-wise OR or AND all the masks together, or something more complicated such as the voting scheme used in Borenstein & Ullman 2004.
- They do not specify a means of ensuring that the candidate fragment set *could possibly* cover a whole image. Just increasing the size of the reliable fragment set, until the training data can be covered, will probably also include a large number of unnecessary candidate fragments.
- What edge detector they use and how the edge images are compared.
- The level of false alarms α, used to calculate thresholds θ for fragment detection.
- How to apply the detection threshold θ when detecting fragments, and whether multiple instances of each fragment are allowed.
- They mention detecting fragments at various scales, but do not explain how or why they do this; they also talk about 'image windows' but this is not explained at all.

Borenstein & Ullman 2004

We were unable to get their segmentation voting algorithm to work well given hand-segmented training images (ignoring the main thrust of the paper to do with fragment segmentation learning in this evaluation). We did not evaluate their fragment segmentation learning algorithm but they claim that this performs almost as well as having labeled training data.

3.6.1 General Conclusions

Both methods do seem to be doing something reasonable: while far from accurate, the segmentations obtained do have some merit, especially for example on the underside of the horse, and the reliable fragments chosen seem to have fairly good intra-class correlation. However, there seem to be many problems with the algorithms.

- They have no background fragment set with which to match background areas of a test image. Their algorithm assumes no knowledge of these areas, but one should hope to be able to make a positive figure-ground classification for every pixel. We will suggest in the next chapter how to learn an appearance model for the background of images.

- In our experiments with matching just the foreground regions of fragments (by ignoring the background regions of fragments when matching) we obtained worse results. This is because the match is then much less discriminative. In §4.5.2 we present a solution for this which mixes image patches based on shape hypotheses from foreground and background appearance models.

From the investigation in this chapter we have seen the limitations of the fragment based approaches of Borenstein & Ullman. The main problem seems to be the lack of global shape consistency. In the next chapter we will investigate our new method for class-specific segmentation.

CHAPTER 4

TOWARDS IMPROVED CLASS-SPECIFIC SEGMENTATION

4.1 Introduction

As described in the introductory chapter, the aim of the work presented in this chapter is to achieve class-specific figure-ground segmentation. That is, given an input image containing an object of a certain type (e.g. horse, car, person), we want to label pixels in that image as belonging to the foreground ('figure') or background ('ground'), illustrated in fig. 1.1. In the previous chapter, we presented an investigation of the techniques of Borenstein & Ullman for solving this problem. As demonstrated, our implementations of their methods in [9, 10] produced very poor results, and consequently we have worked towards a new, improved class-specific segmentation algorithm.

Our technique is similar to that of Borenstein & Ullman in that it relies on local features of appearance and shape. However, we have investigated in detail the effects that occur at the *boundaries* of objects and the technique deals with these correctly. As described below, analysing only *local* appearance and shape is not sufficient to produce a *globally* consistent segmentation. While Borenstein & Ullman presented two fairly ad-hoc optimisation algorithms to achieve this, we have re-formulated the problem into that of a pairwise Markov Random Field (MRF) optimisation (see e.g. [31, 29]), which is amenable to many standard machine learning techniques.

This chapter is organised as follows. §4.2 gives some insight into the problem and motivation for our technique, which is followed in §4.3 by the overall methodology. Then we introduce our models for shape in §4.4 and appearance in §4.5, and describe how these are learned from our training data. §4.6 expands on how local hypotheses are generated, and

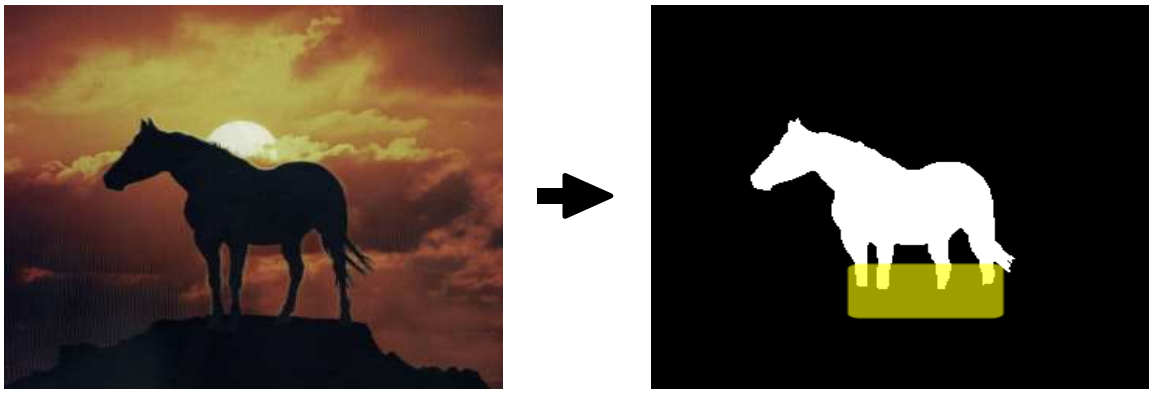


Figure 4.1: Left: Input image. Right: Output from bottom-up segmentation algorithm (from [12]). While generally fairly accurate where there is a clear image gradient, it fails in the highlighted region where the bottom of the legs look very similar to the background.

§4.7 introduces the MRF optimisation method we use to enforce global consistency. The whole technique is evaluated in §4.8, and the chapter is summarised in §4.9.

4.2 Motivation

The difficulty in figure-ground segmentation lies in the relative appearance of foreground and background. Many bottom-up techniques exist that segment images based on colour and texture alone (e.g. [50, 11, 58]), and these work well for many simple examples, though often provide an over-complete segmentation with more than the two figure-ground labels we would like. However, for most object classes it is reasonable to assume that the background can be anything at all, and this is where bottom-up methods can fail: in regions where the foreground and background are very similar it becomes almost impossible to tell figure from background at a local level. For example, finding a black horse on a black background is very hard, even for humans; we must rely on our *a-priori* knowledge of shape and hope that there are at least a few distinctive regions in the image to help guide our search. Fig. 4.1 illustrates this, where the bottoms of the legs of a horse have been marked as background by a bottom-up segmentation algorithm.[1]

We propose a top-down method that *learns* from a set of training data (see Appendix A) generative models for shape and appearance that are representative of the class in question.[2]

[1] This result was generated by an implementation by Stefano Bucciarelli of [12].
[2] A generative model is simply one that can *generate* the input data.

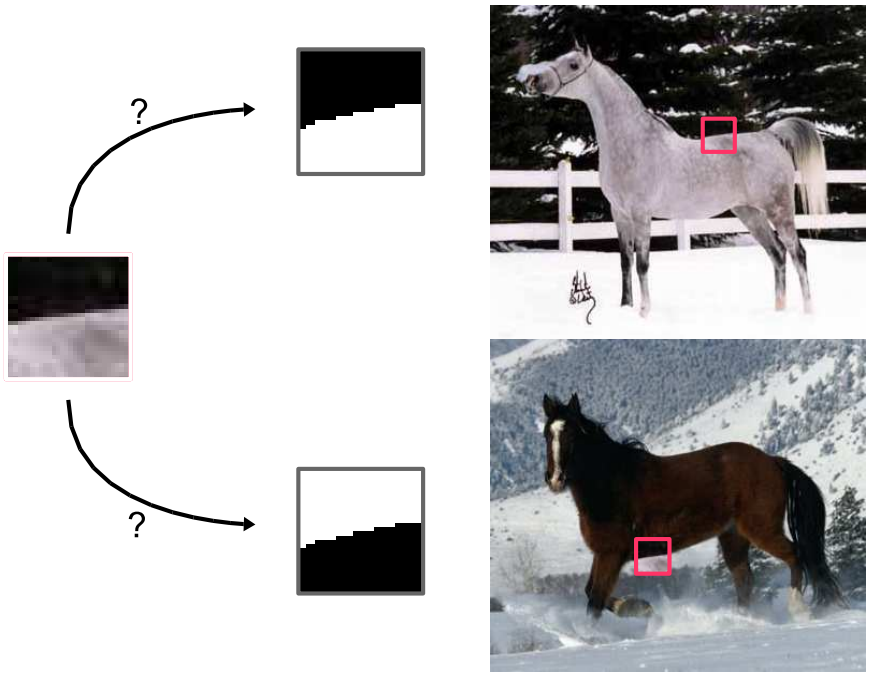


Figure 4.2: An illustration of the inherent ambiguity of using local appearance features. An enlarged appearance patch (left) can be matched well to both the images (right). Note the consequent ambiguity in shape (middle). In this illustration, the enlarged patch was actually extracted from the image of the white horse.

These models are based on *local* features, which afford a lot of flexibility, though often too much: local appearance has an inherent ambiguity which must be recognised and accounted for. Fig. 4.2 gives an example of this problem; note that the enlarged appearance patch could equally well lie on the back of the white horse or the belly of the black horse. Because of this it is necessary to enforce *global* consistency across the image to disambiguate these problems.

4.2.1 Background Models

The recent papers of Borenstein & Ullman ([9, 10]) build background appearance models from a training set of background images. We claim that this is the wrong way of going about things. Unless we are dealing with a specific object class that only occurs in certain environments (such as clouds against the sky), it is reasonable to assume *any* natural (or indeed artificial) background. Hence to learn a representative background model would require a *vast* amount of training data; in theory, every possible image would have to have been seen. Clearly this is not a plausible requirement especially when designing a computationally efficient system.

The only sensible alternative is to learn the background model from the test image itself.

Figure 4.3: Example trimaps. Red indicates foreground and blue indicates background. Note that in our current implementation only the blue areas are used, i.e. only a background appearance model is built using the trimap: the red areas are included for evaluation purposes only.

Unfortunately this results in a chicken-and-egg problem: we need to know where the background is in the image to learn a good background model, yet this is what we are trying to estimate. For the task at hand however, we can permit (minimal) user interaction. The recent 'GrabCut' paper [58] does exactly this: it initialises the background model based on a bounding box of the object provided by the user, and then iteratively improves the model based on the result of their segmentation algorithm. This methodology seems ideal and is one that we should like to emulate in time.

However, our current method makes use of slightly more user interaction: a 'trimap'. A trimap tells us very roughly where in the image is foreground and background, but leaves the border area unmarked (see fig. 4.3). It is important to note that our use of the trimap is entirely 'soft' and is *only* used to learn background appearance models, and not at the segmentation stage. Also, while the trimap does indicate a foreground region, we only used this for comparative evaluation purposes; the foreground appearance model is learned *entirely* from the training database. If we were given some other *a-priori* knowledge about the test image (such as, 'white horse against green field') one could quickly dispense with the trimap completely.

4.3 Methodology

This section presents an overview of our technique and introduces some important concepts that are used throughout. Fig. 4.4 illustrates the whole process. First, models of local shape and foreground appearance are learned from a training set. As discussed above, the test data is labeled with a soft trimap from which the background appearance model is learned. Given the shape and foreground and background appearance models, local hypotheses for

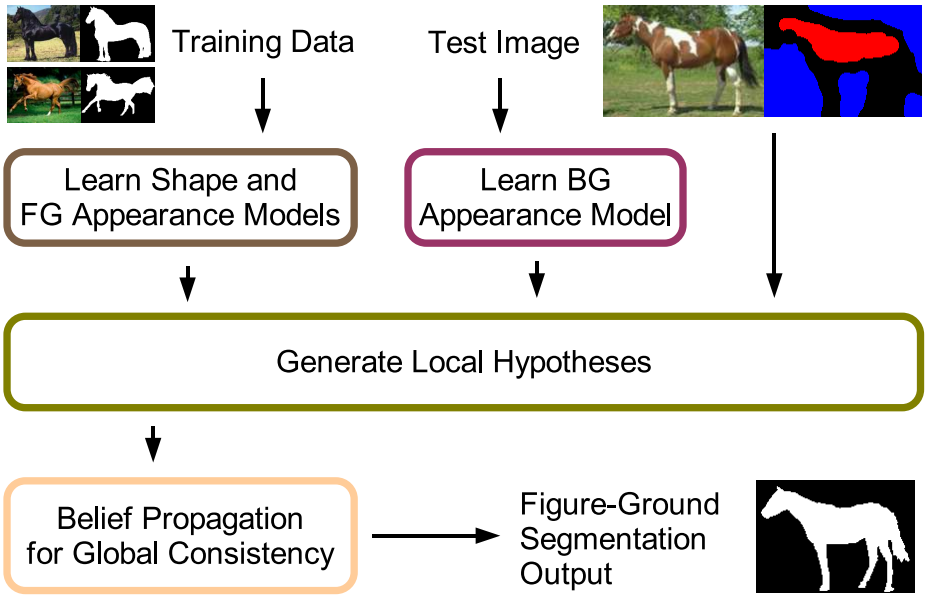


Figure 4.4: Methodology.

segmentation are generated on a regular grid over the test image. Finally, a Belief Propagation algorithm is run to ensure a global consistent figure-ground segmentation for the whole image.

4.3.1 Local Hypotheses

One important consideration is how to ensure a set of local hypotheses that completely cover the input image. The approach taken here is to divide the input appearance image into a regular grid[3] with a total $N = N_x \times N_y$ grid squares, where each grid square (corresponding to an image patch) has size $p \times p$ pixels, and then to formulate hypotheses for each grid square regarding the underlying appearance and shape.

As a preview of §4.4 and §4.5, we represent shape as local, real-valued mask patches, where 1 represents foreground (figure) and 0 represents background, together with learned adjacency likelihoods to enforce global consistency. We represent appearance with two sets of local colour image patches, learned separately for foreground and background.

For each grid square we want to produce a set of hypotheses about what underlying shape generated that image patch. With no prior knowledge, all we have to work with is

[3] Appropriate care is taken when the image is not exactly divisible by the grid size.

the local appearance of the test image at the grid square. Assuming our shape database is sufficiently representative, we know it must contain the real shape underlying the observed image appearance patch. Also assuming our appearance databases are sufficiently representative, we can therefore *generate* for each hypothesised shape an appearance patch that is a maximum-likelihood description for the observed data. Clearly for bad shape hypotheses the description generated will not be very good and have a low likelihood, while for the correct shape hypothesis we should generate an appearance patch with a very high likelihood. Of course, we know that local appearance is inherently ambiguous and so a further global consistency step will be necessary to isolate the correct hypotheses.

This generative model of appearance given shape is expounded in §4.5.2.

4.3.2 Shift-Invariance

The regular grid construction raises a problem of *shift-invariance* for both shape and appearance patches. We must ensure somehow that the patch models we learn can represent image patches at all possible translations ('shifts'), otherwise the quality of match would be dictated by the precise alignment of the input image. A simplistic patch database is unlikely to be shift-invariant, since there must happen to be a suitable patch in the model at exactly the right horizontal and vertical alignment to be able to generate a good match.

A rather messy solution would be to copy each patch in the database at sufficiently many shifts. Fortunately there is a much cleaner solution, based on 'shiftable windows' from the stereo reconstruction literature [63], which allows shift-invariance to be explicitly incorporated into our framework. For this to work, it is necessary to learn shape and appearance patches at a larger size than the grid squares: for full shift-invariance with grid squares of size $p \times p$ pixels, the database patches must have size $q \times q$ pixels where $q = 2p - 1$ (see fig. 4.5).

In our experiments we have found that shift invariance is not so crucial for appearance as it is for shape. This is because the error from misaligning appearance patches is very small compared to the error if the shape is misaligned, since appearance varies slowly and smoothly within the foreground or background regions.

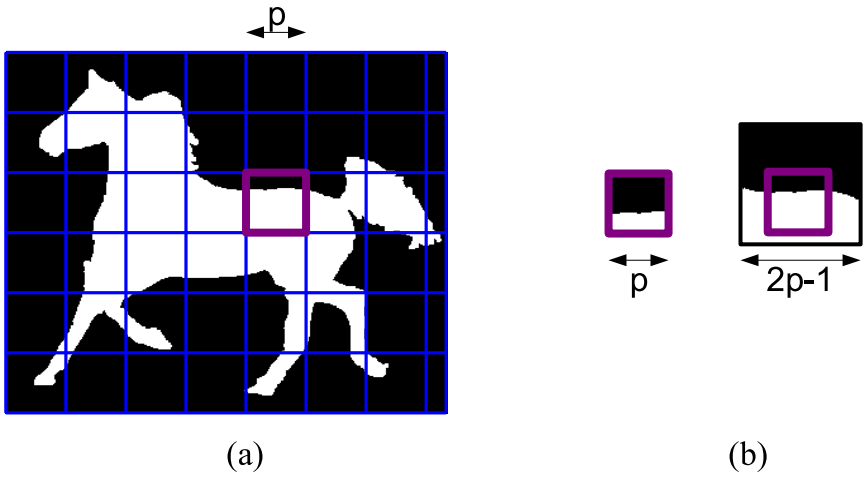


Figure 4.5: (a) A ground-truth mask image has been divided into a regular grid, overlaid in blue. Note that this is an illustration only: normally it is the *appearance* image that is divided into a regular grid. The shape model must contain a good match for the grid position highlighted in magenta. (b) A simplistic shape patch database would have to have seen the patch at exactly the right horizontal and vertical alignment. The left patch shows the best match possible from a non shift-invariant shape patch database: note the misalignment between the mask patch and the best matching database shape. The right example shows that a shape patch database that allows shift-invariance greatly improve the quality of the match since the necessity for precise alignment of image to database is removed.

4.4 A Local Shape Model

Many shape models have been proposed that rely on a set of rigid or deformable templates (e.g. [68, 72]). These global shape models cope well with rigid or semi-deformable objects, but for articulated objects (such as people or animals) it seems sensible to treat shape at a local level combined with a model of spatial arrangement. Most articulated objects have parts that can move largely independently, so it is a huge and unnecessary computational burden to model all possible shapes jointly. For example, a left arm looks very similar to a right arm, and so the same local shape should be able to model both arms independently, though some method of joining the arms to a torso is needed. There is also the question of how restrictive the model should be: should it disallow outright a four- or six-fingered hand, for example, or should the image drive the search.

In [21], Felzenzswalb & Huttenlocher present 'Pictorial Structures' where local parts are connected in a spring-like fashion into a tree-structured model. The constellation model of [14, 77, 23] represents shape with a semi-deformable constellation of parts. Mikolajczyk et al in [55] learn a rigid shape model based on edge features.

We choose to use a very local shape model based on a database of local shape patches, combined with a neighbourhood consistency model to represent global spatial arrangement. Using local patches should cope fairly well with global image deformations (scale, foreshortening, etc.) as well as partial occlusions.

4.4.1 Building the Shape Patch Database

We represent local shape by way of shape *patches*: small square figure-ground masks derived from the training set of mask images. We wish to create a set of shape patches that is compact (for computational efficiency) but representative (for accuracy), and ideally, class-specific (for inter-class discriminability).

Formally, we write the training data as (appearance, mask) image pairs:

$$T = \{(A_1^T, M_1^T), \ldots, (A_{K_T}^T, M_{K_T}^T)\} \tag{4.1}$$

where each colour appearance image A_i is treated as a function $A_i^T : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{R}^3$, and each corresponding mask as the function $M_i^T : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{B}$ where $\mathbb{B}$ represents the range $[0, 1]$, in other words, the mask is a real-valued image, 1 being fully foreground and 0 fully background.[4] Note the training data mask images are purely binary valued; we are not concerned here with alpha value estimation (as in [60, 16, 58]). We allow a real valued mask simply to represent uncertainty in shape estimation.

From the mask images a large number K_M of mask patches $M = \{\mathbf{m}_1, \ldots, \mathbf{m}_{K_M}\}$ pixels are chosen at random such that they contain at least a fraction f of foreground and background, i.e. $f \leq \bar{m}_i \leq (1 - f)$ where $\bar{m}_i$ is the mean value of mask patch $\mathbf{m}_i$. Note that shape patches representing solely foreground ($\bar{m} = 1$) or solely background ($\bar{m} = 0$) are treated specially and therefore excluded at this stage; here we are concerned with learning shapes that lie on the *boundary* of the object.

These mask patches could be used without any further processing as the local shape model. However, while a comparatively large database might be representative of the training data, it probably would not have good generalisation properties; this is investigated further in the evaluation of §4.8.1. To solve both problems we employ a vector-quantisation step. More specifically, we use a modified k-means clustering algorithm (see e.g. [5]) to

[4] In §4.8.2 we evaluate the use of the RGB and YUV colour spaces, as well as grey-scale only.

cluster our K_M random mask patches into a much smaller set of K_S shape patches.

k-means clustering

The k-means clustering algorithm takes as input a set of vectors in some high-dimensional space and groups these together into clusters so as to minimise the total energy in the system, defined as the sum of squared distances from each point to its corresponding cluster centre. Of course, any minimum found will be purely local, since finding the global optimum is an NP-hard problem, and so the quality of resulting clusters depend on the initialisation.

To apply this algorithm for our purposes, we take the random set of mask patches M at size $q \times q$, and then treat these as vectors (of dimension q^2). We employ the Euclidean distance metric, equivalent to assuming an isotropic Gaussian noise process. We initialise the cluster centres to (different) random mask patch vectors, and the algorithm proceeds iteratively as Algorithm 2.

Algorithm 2 k-means clustering

repeat

Assign each mask patch to the cluster with the currently closest centre:

$$z_i \leftarrow \arg\min_k \sqrt{\|\mathbf{m}_i - \mathbf{s}_k\|^2} \tag{4.2}$$

Recompute cluster centres as the mean of all mask patches assigned to each cluster:

$$\mathbf{s}_k \leftarrow \frac{1}{N_k} \sum_{i:z_i=k} \mathbf{m}_i \tag{4.3}$$

until convergence

definitions:
$\mathbf{s}_k$ is the centre of cluster k
z_i is a cluster membership indicator for mask patch i
N_k is the number of mask patches currently assigned to cluster k

The algorithm is guaranteed to converge since the energy in the system, defined as

$$E = \sum_k \sum_{i:z_i=k} \|\mathbf{m}_i - \mathbf{s}_k\|^2 \tag{4.4}$$

is always decreasing and bounded below at zero. Due to the random initialisation however, there is no guarantee that the final solution will be a good one. Therefore, ideally, the algorithm should be run several times with different initialisations, and cluster set with lowest

validation error should be taken.

Translation invariant clustering

It was found by experiment (illustrated in fig. 4.6a) that a naïve k-means clustering produced clusters with very blurry edges and interiors due to the misalignment of the mask patches that were chosen at random. This blurriness can be quantified by an unnecessarily high energy upon convergence. To combat this we have designed an additional step for the clustering algorithm to make it somewhat translation invariant, resulting in lower energy at convergence and cluster centres having much clearer boundaries, giving consequently better performance.

There have been attempts in the literature to deal with transformations while clustering, such as [26] and [30] . However, the former deals only with point sets not vectors of pixels, while the latter only works with transformations that can be represented by invertible matrices acting on the patch vector and this it turns out is inapplicable in our case.

Instead, we treat the task as that of clustering while simultaneously estimating a translation for each patch, and call our algorithm 'TI-clustering' or Translationally Invariant clustering.[5] Firstly, the random mask patch set M is created with a larger patch size ($(2q-1) \times (2q-1)$ pixels) than is needed for the final shape patches ($q \times q$ pixels). This allows us to cluster on a *sub-window* within the large patch while simultaneously estimating the best shift of the sub-window (see fig. 4.7). At each stage of iteration, the position of the sub-window is allowed to shift by a certain amount, such that the vector formed from the pixels in the new sub-window position is strictly closer to the cluster centre than its previous position.

Writing $\hat{\mathbf{m}}_i = \text{subwindow}_{q\times q}(\mathbf{m}_i, \mathbf{x}_i)$ as the smaller sub-windowed copy of mask patch $\mathbf{m}_i$ at shift $\mathbf{x}_i$ and of size $q \times q$ pixels, the algorithm is initialised with (different) random sub-windowed mask patches as the cluster centres, with each shift aligning the sub-window with the centre of the mask patch. It proceeds as Algorithm 3.

This algorithm is again guaranteed to converge to a local minimum since the total energy

$$E = \sum_k \sum_{i:z_i=k} \|\hat{\mathbf{m}}_i - \mathbf{s}_k\|^2 \tag{4.10}$$

[5]TI-clustering is a completely separate step from shift-invariant hypothesis generation, though similar in motivation. The output shapes from the TI-clustering are later used in a shift-invariant manner to generate local hypotheses.

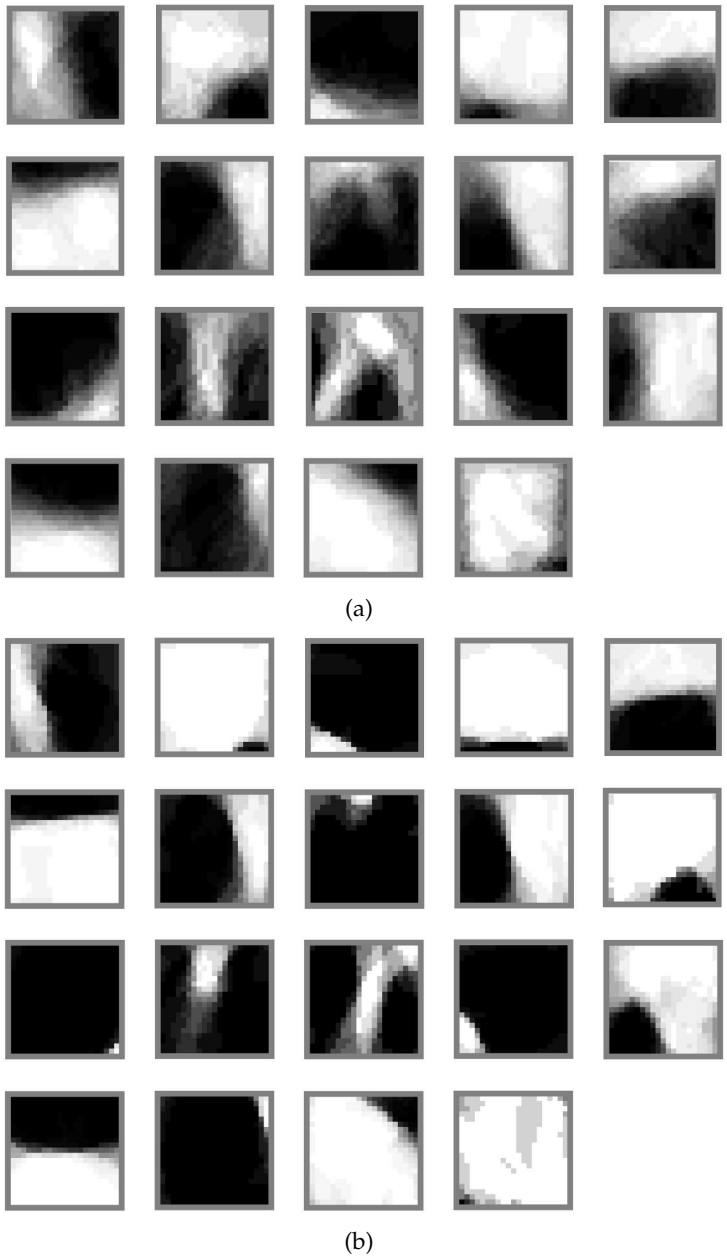


Figure 4.6: (a) Example cluster centres found by the standard k-means clustering algorithm. Note the blurry edges and interiors due to misalignment. The algorithm converged after 14 iterations to an energy of 3090. (b) The corresponding cluster centres found by our Translationally Invariant clustering algorithm given identical initialisation. Note the edges are much sharper and the foreground more solid. The algorithm converged after 22 iterations to an energy of 1500, less than half that of the original algorithm, indicating much tighter clusters.

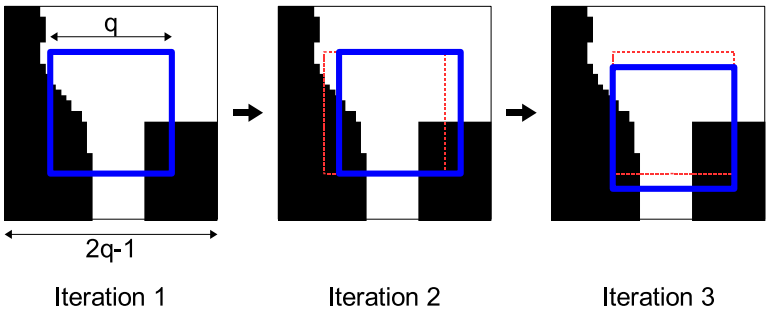


Figure 4.7: The TI-clustering algorithm works on a sub-window patch (shown in blue) which is allowed to shift at each iteration, such that the distance from the cluster centre to the vector representing the new sub-window patch is always reduced. The sub-window position at the previous iteration is shown in dashed red.

Algorithm 3 TI-clustering

repeat

Assign each mask patch to the cluster with the currently closest centre:

$$z_i \leftarrow \arg\min_k \sqrt{\|\hat{\mathbf{m}}_i - \mathbf{s}_k\|^2} \tag{4.5}$$

Recompute sub-window shifts:

for each mask patch $\mathbf{m}_j$ such that $z_j = k$ **do**

Calculate effective new cluster centre without $\mathbf{m}_j$:

$$\mathbf{s}'_k \leftarrow \frac{1}{N_k - 1} \sum_{i:z_i=k, i\neq j} \hat{\mathbf{m}}_i \tag{4.6}$$

Search for sub-window shift that minimises distance to cluster centre:

$$\mathbf{x}_j \leftarrow \arg\min_{\mathbf{x}\in\mathcal{N}(\mathbf{x}_j)} \sqrt{\|\text{subwindow}_{q\times q}(\mathbf{m}_j, \mathbf{x}) - \mathbf{s}'_k\|^2} \tag{4.7}$$

Set

$$\hat{\mathbf{m}}_j \leftarrow \text{subwindow}_{q\times q}(\mathbf{m}_j, \mathbf{x}_j) \tag{4.8}$$

end for

Recompute cluster centres as the mean of all new sub-windowed mask patches assigned to each cluster:

$$\mathbf{s}_k \leftarrow \frac{1}{N_k} \sum_{i:z_i=k} \hat{\mathbf{m}}_i \tag{4.9}$$

until convergence

definitions:

$\mathbf{s}_k$ is the centre of cluster k
$\mathbf{s}'_k$ represents a temporary cluster centre
z_i is the cluster membership indicator for mask patch i
N_k is the number of mask patches currently assigned to cluster k
$\mathcal{N}(\mathbf{x}_i)$ is the set of shifts in the neighbourhood of the current shift $\mathbf{x}_i$.

decreases at each iteration and is bounded below at zero. Note that it usually takes slightly more iterations to converge than k-means, since additional shift parameters are being estimated. Also, the shift re-estimation stage of the algorithm can be time consuming, and hence the search window $\mathcal{N}$ is restricted to a one pixel neighbourhood in our implementation.

A result of this algorithm is shown in fig. 4.6b. This clearly shows the boundaries of the clusters to be sharper, the desired result. Quantitatively in this example, the translation invariance allows the algorithm to converge to a local minimum at less than half the energy of the simple k-means algorithm. §4.8.1 evaluates this algorithm more fully.

Selecting the shape patches

We use the cluster centres resulting from the TI-clustering algorithm to form the shape patch database. We define the set $\mathcal{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_{K_S}, \mathbf{1}, \mathbf{0}\}$ of $K_S + 2$ shape patches as simply the cluster centres returned from the algorithm above, together with shape patches representing foreground ($\mathbf{1}$) and background ($\mathbf{0}$).

4.4.2 Estimating the Shape Neighbourhood consistency

In addition to finding a concise, representative set of shape patches, we need to model how these shapes fit together. We choose again a local neighbourhood consistency model which fits neatly into our grid-based subdivision of the input image. While we generate shape and appearance hypotheses for each grid square independently, we know that this is not sufficient to produce a globally consistent figure-ground segmentation, and hence we formulate the problem into a Markov Random Field (MRF) which we optimise with Belief Propagation as described in §4.7. Part of the specification of the MRF is a neighbourhood consistency function ψ, which represents the marginal likelihood of a particular hypothesis given a neighbouring hypothesis.

We have found that it is useful to decompose this function into two parts: first, a consistency model that is learned from the training data; and second, a consistency model based on a region of overlap between neighbouring hypotheses. The latter, denoted ψ^{overlap}, can only be estimated at a later stage once the hypotheses have been generated; this is discussed in detail in §4.7.1.

The former, denoted ψ^{model}, can however be trained off-line as follows. Once we have learned the set of shape patches S, we can revisit the *training* mask images and find instances of the shapes. These instances are found by treating the mask images in a very similar way to

real input images: each mask image is divided into a regular grid, and for each grid square, the shape patch which best matches the mask image patch is found, using an SSD difference measure and taking due care with shift-invariance as described in §4.3.2.

Given these matches, it is simple to generate histograms of adjacency likelihood based on frequency of neighbouring occurrence. Since we later use an 8-neighbourhood system for our MRF, we choose to build eight adjacency histograms corresponding to the eight neighbourhood directions. Each histogram is build using a frequency table which is initialised with ones to ensure that, for generality, zero probability adjacency never occurs. For each training mask image, the relevant entries in each frequency table are incremented according to the detected instances of shape patches. Finally, probability tables are created from the histograms by normalising the frequencies to give marginal adjacency statistics.

So that we can weight between ψ_{overlap} and ψ_{model}, we raise the adjacency likelihoods to a power $\lambda = 0.1$.

4.5 A Local Appearance Model

The above section developed our local shape model, and it is now necessary to address how we learn and represent generative models of appearance. As previously discussed we wish to model foreground appearance independently of background appearance. With sufficient training data, one supposes that a representative foreground appearance model can be learned from the training data alone. However, we noted that with any sensible amount of training data one cannot hope to learn a representative background appearance model for arbitrary test images, and hence a background model should be learned separately for each test image. As previously mentioned we employ a trimap to indicate regions of background in the test image. Regardless of whence each model is learned, it is sensible to assume they both have the same general form.

4.5.1 Patch Database

We learn an appearance patch database separately for foreground (from the database) and background (from the image, based on the trimap). For each, we extract a large number of random patches of size $p \times p$ pixels, and cluster these to a smaller set of patches using k-means clustering, as described above. This ensures that we get good generality but still have a manageably small database. We have investigated learning appearance for grey-scale, and

the RGB and YUV colour spaces, and our results are shown in §4.8.2. In theory, appearance patches should be made shift-invariant (see §4.3.2) as for shape patches, but in practice this was not found to be very important.

There is a subtly regarding the foreground patch database: a large amount of texture information occurs at the boundary of objects, due to shading from the curvature of the object. Hence it is not sufficient to learn foreground patches from just the centre of the object (i.e. where the shape is purely foreground); we must take 'midground' patches into account. We define midground patches as those for which the underlying shape is a mix of foreground and background. To learn appearance for midground patches therefore, we look at the appearance patches from the database corresponding to members of each midground shape patch cluster. These are clustered with k-means, as before, but using a difference metric (weighted SSD) that reflects the *shape*, as explained in Appendix C, so that midground appearance patches are clustered in their foreground regions only. These clusters are then used as described below to generate maximum likelihood descriptions of observed patches by mixing the midground appearance with background appearance learned from the test image.

4.5.2 Mixing Appearance According to Shape

An important insight is that every grid square in the test image must lie either completely over foreground or background, or over a mixture of foreground and background along the boundary of the object. This third type we will call midground. For foreground and background patches it is necessary only to examine the relevant appearance model; however, for easy of explanation, foreground and background patches can be treated as special midground patches with shapes all ones or all zeros respectively.

Midground image patches have been generated in the original imaging process as a mixture of foreground and background. Since we assume that the image background is independent of its foreground, we must treat the background region independently of the foreground region. Consequently for each midground shape hypothesis we must find the best mix, according to shape, of foreground and background appearance patches. Thankfully the independence property implies we can search for the best matching foreground and background appearances sequentially rather than factorially, which makes this far more computationally tractable.

Recall that for each grid square in the test image, we wish to generate a set of hypotheses

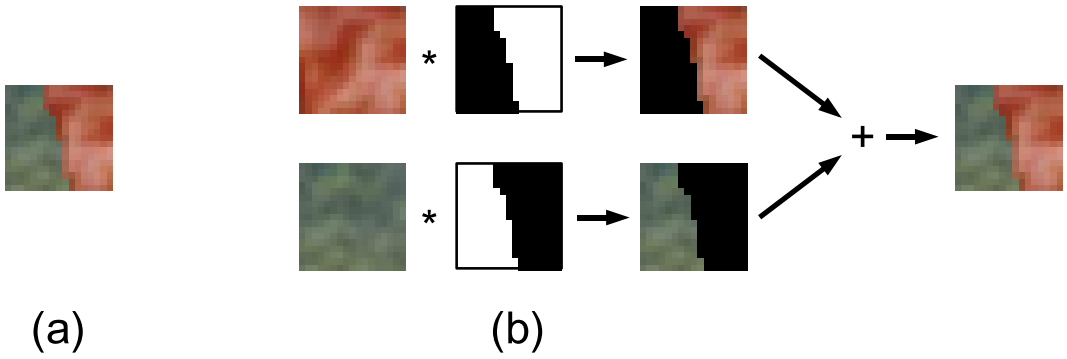


Figure 4.8: Generating midground appearance patches. (a) The patch from a grid square on the test image. (b) A midground appearance patch hypothesis is generated using a patch from the foreground appearance model (top row) and a patch from the background appearance model (bottom row). The hypothesised shape is used as a mask, shown in the top row second column, and its inverse, shown in the bottom row second column. The hypothesised midground appearance patch is then the sum of hypothesised foreground and background appearances.

of underlying shape, based on the maximum-likelihood description of the observed data that our models of shape and appearance can generate. For those hypotheses corresponding to foreground or background shapes it is a simple matter to generate the appearance corresponding hypothesis as the maximum likelihood appearance patch from the model.

For midground shapes however, more care must be taken. Since we cannot directly estimate the match between the hypothesised *shape* and the image patch, we must base the likelihood exclusively on an appearance hypothesis. As illustrated in fig. 4.8, midground appearance hypotheses are generated by mixing the maximum likelihood foreground and background appearance hypotheses, using a mask derived from the shape hypothesis.

Suppose for a particular grid square g on the test image with appearance $\mathbf{a}_g$ we are examining shape hypothesis $\hat{\mathrm{s}}$ (which, recall, is a mask). Given some function $f_{\mathbf{s}}(\mathbf{a}, \mathbf{a}_g)$ representing the dissimilarity in appearance in region $\mathbf{s}$, between generated patch $\mathbf{a}$ and observed image patch $\mathbf{a}_g$, we seek the maximum likelihood (i.e. minimum dissimilarity) generated appearance hypothesis $\mathbf{a}$. We search through the foreground patch database looking for the best match according to the shape:

$$\hat{\mathbf{a}}^{\diamondsuit} = \arg\min_{\mathbf{a} \in A^{\diamondsuit}} f_{\hat{\mathbf{s}}}(\mathbf{a}, \mathbf{a}_g) \tag{4.11}$$

Similarly for background we search using the inverse of the mask:

$$\hat{\mathbf{a}}^{\blacklozenge} = \arg\min_{\mathbf{a}\in A^{\blacklozenge}} f_{(\mathbf{1}-\hat{\mathbf{s}})}(\mathbf{a}, \mathbf{a}_g) \tag{4.12}$$

Finally given these two separate hypotheses, we mix them together:

$$\hat{\mathbf{a}} = \hat{\mathbf{a}}^{\diamondsuit} \star \hat{\mathbf{s}} + \hat{\mathbf{a}}^{\blacklozenge} \star (\mathbf{1} - \hat{\mathbf{s}}) \tag{4.13}$$

where $\star$ represents pixel-wise multiplication (as in `.*` in MATLAB).

In our evaluation of §4.8.3 we use the weighted SSD function (described in Appendix C) for f. We also define the likelihood by assuming a zero mean Gaussian noise process:

$$p(\hat{\mathbf{a}}|\hat{\mathbf{s}}) = \mathcal{G}(v; 0, \sigma_i^2) \tag{4.14}$$

where the mixed score v is described in §C.4, and σ_i is a constant set by hand.

4.6 Generating Hypotheses

For clarity, we summarise here the process of generating hypotheses for a test image. The test image is divided into a regular grid with squares of size $p \times p$ pixels. We assume a shape model as a set of shape patches each of size $q \times q$ pixels (where $q = 2p - 1$) to enable shift-invariant matching. We also assume foreground and background appearance models as sets of appearance patches. Algorithm 4 is employed to generate the hypotheses $\hat{\mathbf{h}}_g^k$.

Once we have generated for a particular grid square g a set of hypotheses

$$\hat{H}_g = \{\hat{\mathbf{h}}_g^1, \ldots, \hat{\mathbf{h}}_g^{K_S+2}\} \tag{4.18}$$

(one for each shape patch), we in fact disregard the subset $\tilde{H}_g \subset \hat{H}_g$ of hypotheses with lowest likelihood to leave a final set of K_H hypotheses $H_g = \hat{H}_g \backslash \tilde{H}_g$, such that for all pairs of triples $(\mathbf{s}, \mathbf{a}, p) \in H_g$ and $(\tilde{\mathbf{s}}, \tilde{\mathbf{a}}, \tilde{p}) \in \tilde{H}_g$, $p > \tilde{p}$. We also enforce that the hypotheses representing foreground and background are also included in H_g to ensure the possibility of a globally correct segmentation. We do this simply for computational efficiency, bearing in mind that the hypotheses with lowest likelihood are rarely (if ever) going to be used in the final result; however K_H should be large enough to allow for the inherent ambiguity in appearance. We used a value of $K_H = 10$ in our evaluation. Fig. 4.9 illustrates the set of

Algorithm 4 Generate shape and appearance hypotheses

Divide test image into regular grid with squares of size $p \times p$
for each grid square $g \in \{1, \ldots, N\}$ **do**
  for each shape $\mathbf{s}_k, k \in \{1, \ldots, K_S + 2\}$ **do**
    for each shift $\mathbf{x}_i$ **do** [to enable shift-invariance]
      Compute sub-windowed shape

$$\hat{\mathbf{s}}_i \leftarrow \text{subwindow}_{p \times p}(\mathbf{s}, \mathbf{x}_i) \tag{4.15}$$

      Generate best matches $\hat{\mathbf{a}}_i^{\diamondsuit}$ and $\hat{\mathbf{a}}_i^{\blacklozenge}$
      Combine matches

$$\hat{\mathbf{a}}_i \leftarrow \hat{\mathbf{a}}_i^{\diamondsuit} \star \hat{\mathbf{s}} + \hat{\mathbf{a}}_i^{\blacklozenge} \star (1 - \hat{\mathbf{s}}) \tag{4.16}$$

      Calculate combined appearance likelihood p_i
    end for
    Find maximum likelihood shift index $j \leftarrow \arg\max_i p_i$
    Store hypothesis

$$\hat{\mathbf{h}}_g^k \leftarrow (\hat{\mathbf{s}}_j, \hat{\mathbf{a}}_j, \hat{p}_j) \tag{4.17}$$

  end for
end for

definitions:
$S = \{\mathbf{s}_1, \ldots, \mathbf{s}_{K_S}, \mathbf{1}, \mathbf{0}\}$ is the set of shape patches
$A^{\diamondsuit} = \{\mathbf{a}_1^{\diamondsuit}, \ldots, \mathbf{a}_{K^{\diamondsuit}}^{\diamondsuit}\}$ is the set of foreground appearance patches
$A^{\blacklozenge} = \{\mathbf{a}_1^{\blacklozenge}, \ldots, \mathbf{a}_{K^{\blacklozenge}}^{\blacklozenge}\}$ is the set of background appearance patches

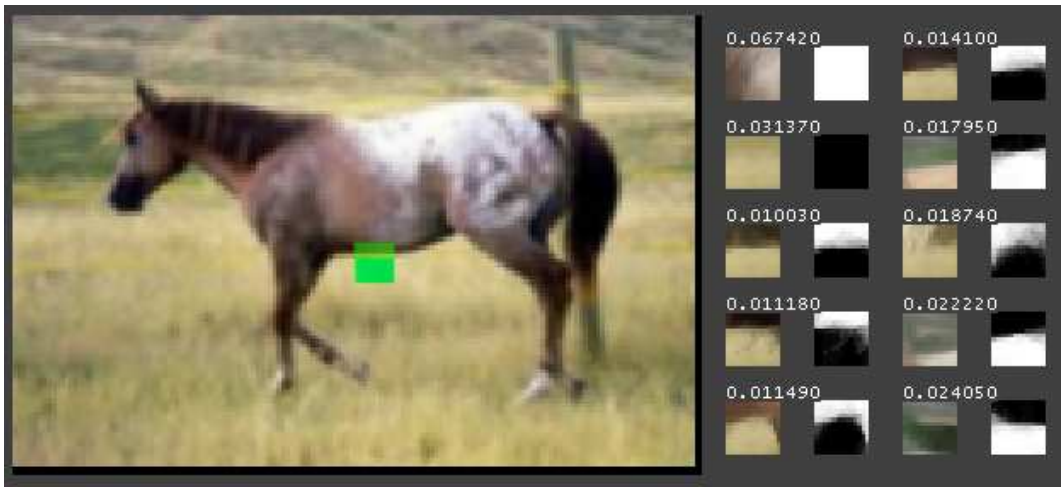


Figure 4.9: Example hypotheses. Left: input image, with a grid square highlighted in green. Right: the set H_g of hypotheses has been generated for the grid square as a pairs of appearance and shape. Note the ambiguity in shape. The numbers above each pair indicate the SSD similarity score. Note that foreground and background hypotheses are always included regardless of score.

hypotheses that has been found for a particular grid square in an image.

It will become important later (to generate ψ^{overlap} in §4.7.1) to be able to consider pixels in a *border* region of b pixels, around both the shape and appearance hypothesis patches, as illustrated in fig. 4.10.[6] With care in implementation, it is easy to ensure these border pixels are available when needed without interfering with the other calculations described above. Note also that a sensible value for b should satisfy $2b < p$ since otherwise there would be overlap with non-neighbouring grid squares. We shall denote the hypothesised shape and appearance patches with the border included as $\breve{\mathbf{s}}_g^k$ and $\breve{\mathbf{a}}_g^k$ respectively.

4.7 Enforcing global consistency

We now have for each grid square in the image a reduced set

$$H_g = \{\mathbf{h}_g^1, \ldots, \mathbf{h}_g^{K_H}\} \tag{4.19}$$

$$= \{(\mathbf{s}_g^1, \mathbf{a}_g^1, p_g^1), \ldots, (\mathbf{s}_g^{K_H}, \mathbf{a}_g^{K_H}, p_g^{K_H})\} \tag{4.20}$$

[6] Note that the border region is not used in shift-invariant matching or calculating appearance likelihoods.

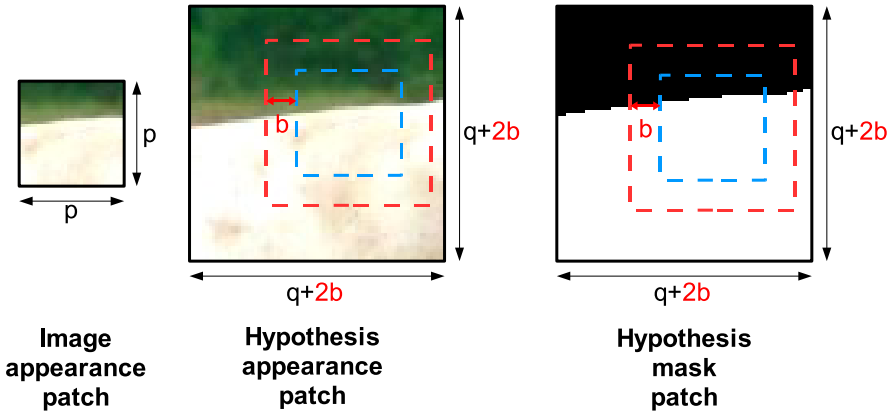


Figure 4.10: Left: an example image patch for which we are generating hypotheses. Middle and right: illustrations of shift-invariant matching with borders of size b pixels. In both cases, the sub-window is only allowed to move over the middle $q \times q$ pixels (where $q = 2p - 1$), ensuring a border is always available. The border region is not used in finding the maximum likelihood appearance hypothesis.

of shape and appearance hypotheses together with their likelihoods (**s**, **a** and p respectively). We need to enforce global consistency since purely local hypotheses can be ambiguous. We do this by embedding the problem onto a pairwise MRF (introduced in [31]) and optimising it using Loopy Belief Propagation (BP).[7] See e.g. [57, 29, 79, 80] for more information on BP.

Our paradigm for global consistency was derived from the super-resolution work of Freeman et al [29] (described in §2.5.2) where they aim to reconstruct a high-resolution image from a low-resolution image. Clearly this is impossible to do perfectly for an arbitrary image since there are many high-resolution images that down-sample to the same low-resolution image. But for a class of images with similar prevalent textures, some resolution enhancement should be possible. Their technique is very similar to ours in concept: they aim to generate high- and low-resolution hypotheses analogously to our shape and appearance hypotheses. They first construct a database of high-resolution image patches from a training corpus, which are down-sampled to produce corresponding low-resolution patches. Dividing the low-resolution test image into a regular grid, they search for good matches of low-resolution patches, thereby generating their hypotheses. They then form a pairwise MRF over the grid, where the labels at each node are indices into the hypothe-

[7] The term 'loopy' is used here to simply indicate that the BP algorithm is being applied to a graph with loops (cycles). This means, unfortunately, that the solution is only a local optimum, whereas for graphs without cycles, BP is guaranteed to converge to the global optimum.

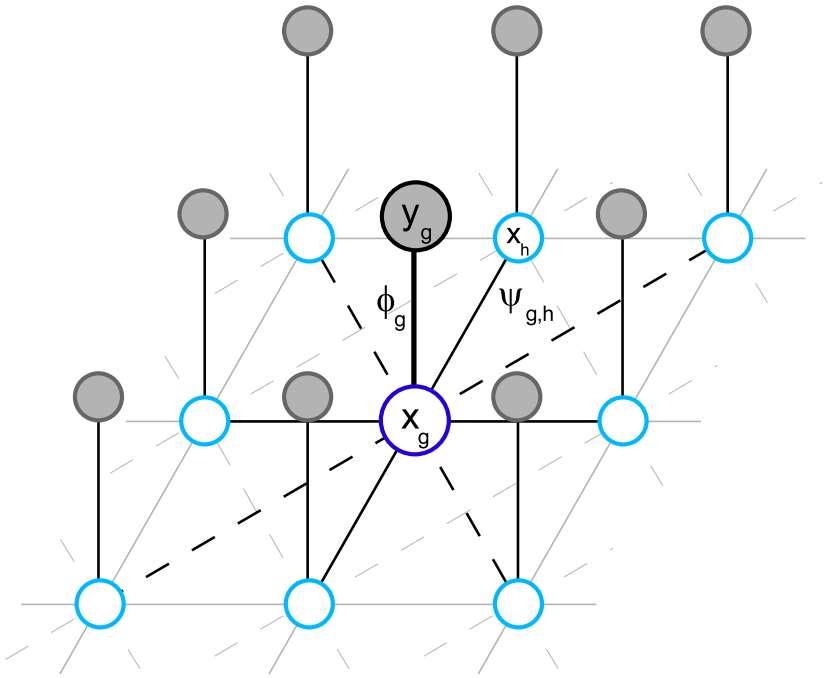


Figure 4.11: A graphical model representing a section of the Markov Random Field. The shaded circles are the observed nodes y_g, each representing the fixed, known appearance of the image at grid square g. The unshaded circles are the hidden nodes x_g, each representing a random variable over the hypotheses of underlying shape and appearance. Vertical lines show the linkage between observed and hidden nodes, while other lines show neighbourhood relations (solid for 4-connected, or solid and dashed for 8-connected). Nodes y_g and x_g are related by a likelihood function ϕ_g representing the local evidence for the hypothesis. Neighbouring nodes x_g and x_h are related by a likelihood function $\psi_{g,h}$ representing the consistency between neighbours.

ses, the evidence functions derive from the quality of match between the image and the low-resolution hypotheses, and the consistency functions derive from the quality of match between neighbouring high-resolution hypotheses. They employ Loopy BP to optimise the MRF, resulting in a globally consistent super-resolution output.

4.7.1 Form of the MRF

We overlay the MRF onto the test image such that there is one site per grid square. Each site $g \in \{1, \ldots, N\}$ consists of two nodes in the graphical model as illustrated in fig. 4.11: an observed node y_g representing the fixed, known appearance of the image at grid square g, and a hidden node x_g representing a random variable over our hypotheses of underlying

shape and appearance.[8] The neighbourhood of the MRF corresponds to the neighbourhood of the grid over the image (either 4-connected or 8-connected).[9] We will write $p_g(x_g = k)$ (and similarly for other distributions) to indicate the probability at site g of hypothesis number k, or for compactness, simply $p_g(k)$.

The MRF is specified by the joint probability distribution over the set of observed image patches $Y = \{y_1, \ldots, y_N\}$ and the hidden variables $X = \{x_1, \ldots, x_N\}$:

$$p(X, Y) = p(x_1, \ldots, x_N, y_1, \ldots, y_N) \tag{4.21}$$

$$= \frac{1}{Z}\left[\prod_g \phi_g(x_g)\right]\left[\prod_{(g,h):g\sim h} \psi_{g,h}(x_g, x_h)\right] \tag{4.22}$$

where the ϕ_g (*evidence*) functions measure how well the hypotheses at site g represent the corresponding observed image patch, and the $\psi_{g,h}$ (*consistency*) functions measure how well neighbouring hypotheses fit together. The neighbourhood relation is denoted $g \sim h$, and Z is a normalisation constant known as the *partition function*.

Our goal will be to infer at each site either the marginal pdf over hypotheses or the maximum a-posteriori (MAP) choice of hypothesis, so that we can estimate the 'best' choice of hypothesis and thereby produce a globally consistent segmentation. First we must specify the forms of the evidence and consistency functions for the MRF, as described next.

Evidence functions

The evidence $\phi_g(k)$ represents the quality of match between hypothesis $\mathbf{h}_g^k = (\mathbf{s}_g^k, \mathbf{a}_g^k, p_g^k)$ and the observed image patch. This likelihood has already been calculated when generating the hypothesis, as p_g^k.

The $\phi_g(k)$ function can be viewed as a fixed vector of likelihoods, each element representing one hypothesis:

$$\phi_g = \begin{pmatrix} \phi_g(1) \\ \vdots \\ \phi_g(K_H) \end{pmatrix} == \begin{pmatrix} p_g^1 \\ \vdots \\ p_g^{K_H} \end{pmatrix} \tag{4.23}$$

[8] Note that the probability distributions we will calculate are over *indices* into the hypotheses rather than over any meaningful values. This unfortunately means that many of the optimisations that have been developed for Loopy BP such as [22] are inapplicable since they require e.g. a convex distance function definable over the labels. For the same reason the MRF cannot be solved with a Graph Cut algorithm ([12, 11]).

[9] In all our experiments we use the 8-connected neighbourhood. We briefly experimented with 4-connected neighbourhoods but often got a checkerboard artefact.

Consistency functions

The consistency functions $\psi_{g,h}(k,l)$ represents the quality of match between neighbouring hypotheses $\mathbf{h}_g^k$ and $\mathbf{h}_h^l$ given by the probability $p(x_g = k, x_h = l)$. The segmentations of neighbouring nodes should match up well to ensure local and thereby global consistency when the BP algorithm converges. We introduced the consistency function in §4.4.2 where we decomposed ψ into the product of two terms:

$$\psi_{g,h}(k,l) = \psi_{g,h}^{\text{model}}(k,l) \times \psi_{g,h}^{\text{overlap}}(k,l) \tag{4.24}$$

The former ψ^{model} we learned as histograms of adjacency likelihoods in §4.4.2. We are now in a position to estimate the latter, ψ^{overlap}.

Recall that we have larger versions of the shape and appearance patches that include a border region. Note that for neighbouring grid squares the larger versions will overlap, and we can use this region of overlap to help ensure consistency, both for shape and appearance. We write $\|\breve{\mathbf{s}}_g^k \ominus \breve{\mathbf{s}}_h^l\|$ as the vector of absolute pixel-wise difference of shape patches in the region of overlap, and similarly $\|\breve{\mathbf{a}}_g^k \ominus \breve{\mathbf{a}}_h^l\|$ for appearance. Assuming a Gaussian noise process of zero mean and isotropic covariance, we decompose the consistency likelihood into:

$$\psi_{g,h}^{\text{overlap}}(k,l) = \mathcal{G}(\|\breve{\mathbf{s}}_g^k \ominus \breve{\mathbf{s}}_h^l\|; \mathbf{0}, \frac{1}{\sigma_s^2} I) \times \mathcal{G}(\|\breve{\mathbf{a}}_g^k \ominus \breve{\mathbf{a}}_h^l\|; \mathbf{0}, \frac{1}{\sigma_a^2} I) \tag{4.25}$$

Note this has introduced two parameters σ_a^2 and σ_s^2 which we currently estimate by hand.

The $\psi_{g,h}$ function can be viewed as a fixed matrix of likelihoods:[10]

$$\psi_{g,h} = \begin{pmatrix} \psi_{g,h}(1,1) & \dots & \psi_{g,h}(1,K_H) \\ \vdots & \ddots & \vdots \\ \psi_{g,h}(K_H,1) & \dots & \psi_{g,h}(K_H,K_H) \end{pmatrix} \tag{4.26}$$

4.7.2 Inference on the MRF

Our goal is to use the above definitions to infer at each grid square g either the marginal pdf over hypotheses or the MAP hypothesis as our globally consistent estimate of segmentation.

[10] With these formulations, the message passing rules can be efficiently implemented as matrix algebra.

Note that the y_g are observed variables, so we have the conditional distribution

$$p(X|Y) = \frac{1}{C}p(X,Y) \tag{4.27}$$

where $C = p(Y)$ is constant with respect to X and therefore can be ignored as it does not affect the optimisation.

Two standard alternatives for inference on MRFs are to find either the marginal distribution or MAP estimate:

- The marginal pdf can be calculated by summing the conditional distribution over all nodes other than the one in question:

$$\begin{aligned} p_g(x_g|Y) &= \sum_{\{x_h, h\neq g\}} p(X|Y) && (4.28)\\ &= \sum_{x_1}\cdots\sum_{x_{g-1}}\sum_{x_{g+1}}\cdots\sum_{x_N} p(x_1,\ldots,x_N|Y) && (4.29)\end{aligned}$$

  where each summation is from $x_h = 1 \ldots K_H$.

  From the marginal we can compute the minimum mean squared error (MMSE) estimate $\bar{\mathbf{s}}_g$ of the underlying shape patch at grid square g as the expectation of the shape patch with respect to the marginal:

$$\bar{\mathbf{s}}_g = \sum_{k=1}^{K_H} \mathbf{s}_g^k p_g(x_g = k|Y) \tag{4.30}$$

- The MAP estimate can be calculated as the maximum of the max-marginal. The max-marginal is defined similarly to the marginal:

$$\begin{aligned} \hat{p}_g(x_g|Y) &= \max_{\{x_h, h\neq g\}} p(X|Y) && (4.31)\\ &= \max_{x_1}\ldots\max_{x_{g-1}}\max_{x_{g+1}}\ldots\max_{x_N} p(x_1,\ldots,x_N|Y) && (4.32)\end{aligned}$$

  where each maximisation is over $x_h = 1 \ldots K_H$.

  From these, the MAP estimate of hypothesis at grid square g follows immediately. Writing:

$$\hat{k}_g = \arg\max_{k=1\ldots K_H} \hat{p}_g(x_g = k|Y) \tag{4.33}$$

the MAP estimate for the shape patch is clearly:

$$\hat{\mathbf{s}}_g = \mathbf{s}_g^{\hat{k}_g} \tag{4.34}$$

Unfortunately, to evaluate either estimate naïvely has exponential cost in the number of hidden nodes. As image size or resolution increases, this cost rapidly becomes infeasible.

Belief Propagation

The tractable technique we employ to perform this inference *approximately* is Belief Propagation (BP). This is an iterative message-passing algorithm where, at each iteration, each node in the graph passes a message to each of its neighbours concerning its 'belief' about their state based on its current information. Several iterations are needed so that the local messages can propagate globally.

For graphs that are singly connected the BP algorithm is guaranteed to converge to the exact solution.[11] When applied to graphs with loops, it is not guaranteed to converge, and even if it does the result is usually only a local optimum. However, for certain problems loopy BP has in practice proved an invaluable tool for machine learning (see e.g. [56, 79]). In our case the MRF is full of loops and so we can only hope for an approximate answer, though it performs well as demonstrated below.

At each iteration a message $\mathbf{m}_{g\to h}(x_h)$ is calculated and sent from node x_g to node x_h, regarding the current belief node x_g has about the distribution over hypotheses at node x_h.[12] The message can be considered a vector where each element represents a different hypothesis. The algorithm is initialised by settings all message distributions to be uniform, which states that we have no initial knowledge about which hypotheses to choose. For our MRF, this means that $\mathbf{m}_{g\to h}(x_h = k) = \frac{1}{K_H}$, for all hypotheses $k = 1 \dots K_H$, and all grid squares g and h. The exception to this rule is around the edges of the image, which we weight heavily in favour of the background hypothesis. Note that this weighting in no way prevents a solution with non-background hypotheses being found by belief-propagation; it only encourages the algorithm to converge to a local minimum that has background around the edge.

[11] A singly connected graph contains no undirected loops.

[12] So as to be consistent with the literature, we here reuse the symbol $\mathbf{m}$ which previously represented mask patches.

If BP converges, the belief $b_g(x_g)$ approximating the marginal pdf $p_g(x_g|Y)$ or the max-marginal $\hat{p}_g(x_g|Y)$ as appropriate is:

$$b_g(x_g = k) = \phi_g(x_g = k) \prod_{x_h \in N(x_g)} \mathbf{m}_{h \to g}(x_g = k) \tag{4.35}$$

The two inference schemes described above give rise to two different sets of BP message passing rules as follows:[13]

- To compute the BP approximation to the marginal distribution, the 'sum-product' rule is used:

$$\mathbf{m}_{g \to h}(x_h = l) \leftarrow \sum_{x_g=1}^{K_H} \left[\psi_{g,h}(x_g, x_h = l)\phi_g(x_g) \prod_{x_f \in N(x_g) \backslash x_h} \mathbf{m}_{f \to g}(x_g) \right] \tag{4.36}$$

  where $N(x_g)\backslash x_h$ denotes the set of neighbours of x_g other than x_h.

  Upon convergence, the MMSE estimate can be calculated by taking the beliefs as approximations to the marginals:

$$p_g(x_g = k|Y) \approx b_g(x_g = k) \tag{4.37}$$

  and then applying eq. 4.30.

- To compute the BP approximation to the max-marginal, the very similar 'max-product' rule is used:

$$\mathbf{m}_{g \to h}(x_h = l) \leftarrow \max_{x_g=1 \ldots K_H} \left[\psi_{g,h}(x_g, x_h = l)\phi_g(x_g) \prod_{x_f \in N(x_g) \backslash x_h} \mathbf{m}_{f \to g}(x_g) \right] \tag{4.38}$$

  Upon convergence, the MAP estimate can be calculated by taking the beliefs as approximations to the max-marginals:

$$\hat{p}_g(x_g = k|Y) \approx b_g(x_g = k) \tag{4.39}$$

  and then applying eq. 4.33 and eq. 4.34.

[13] It is assumed for notational simplicity that all messages are calculated simultaneously for each grid square at each iteration.

Note that to prevent overflow or underflow, after each iteration all messages are normalised to sum to one, to ensure they form proper probability distributions.

The two schemes have different convergence properties. For singly connected graphs, both sets of rules are guaranteed to converge to their respective correct results. For arbitrary graphs with loops, the BP algorithm is not guaranteed to converge. If it does converge, the max-product rules lead to only a local maximum of the posterior, and, for Gaussian processes, the sum-product rules lead to distributions with correct means but incorrect covariances. These results are summarised in the following table [from [29]]:

BP Rules	**Graph Topology**	
	Singly connected	Loopy
Sum-Product	Correct marginals	For Gaussian processes, correct means, incorrect covariances
Max-Product	Correct MAP estimate	Local maximum of posterior

See [78, 29] for more details.

4.8 Evaluation

In this section we present the evaluation and results obtained using the above method. As they are crucial to a good result, we first investigate the performance of our shape and appearance models, and then look into actual figure-ground segmentation results. For all the evaluations we use a validation set of images with their ground-truth segmentations and trimaps, similarly to the training set. Both the training and validation data are described in Appendix A. Similarly to the training images, we write the validation data as image triples of appearance, mask, and in this case, trimap:

$$V = \{(A_1^V, M_1^V, R_1^V), \ldots, (A_{K_V}^V, M_{K_V}^V, R_{K_V}^V)\} \tag{4.40}$$

4.8.1 Testing the Shape Model

It is important to ensure that the final set S of shape patches we use later for segmentation does in fact meet the requirements we set out at the offset: compactness and completeness. For both of these we need a measure of how well a set S represents the object class.

We quantify the quality of a set of shape patches S as follows. First, each mask image M_j^V in the validation set is divided into a regular grid of N mask squares $\mathbf{m}_g^j$ of size $p \times p$ pixels,

as if we were running the full segmentation algorithm. For each grid square, the shape patch sub-window (to account for shift-invariance) with lowest sum-of-squared differences (SSD) to the image mask patch is found.[14] This gives us a lower bound on the error we can achieve in representing the validation mask image by our shape model. The quality measure $\mathcal{Q}(S)$ is then defined as the average error over all midground patches in the validation set:

$$\mathcal{Q}(S) = \frac{1}{p^2 D} \sum_{j=1}^{K_V} \sum_{g=1}^{N} \min_{\mathbf{s} \in S} \min_{\mathbf{x}_i} \|\mathbf{m}_g^j - \text{subwindow}_{p \times p}(\mathbf{s}, \mathbf{x}_i)\|^2 \tag{4.41}$$

where D is the total number of midground patches in the validation set; note that the SSD measure is normalised by the patch size, and that mask patches that are purely foreground or background are excluded from this measure as they can be matched exactly with the corresponding foreground or background shape.

Number of Shape Patches

We investigated the effect of the number of shapes $|S|$ in the shape patch database, and the results are shown in fig. 4.12. All shape patch databases were generated from an initial random set of mask patches of size $K_N = 2000$. The important points to note from the graphs are the following:

- As the number of shapes increases, the error decreases, as expected.
- The larger patch size evaluated (19×19) produced a larger error. This is because larger patches are more specific; the limiting case of this would be using masks of whole objects as the shapes: which very class specific they do not generalise well to validation data. We investigate the class-specificity as a function of patch size below.
- Enabling shift-invariant matching (fig. 4.12(c,d)) greatly reduces the error compared to using non shift-invariant matching (fig. 4.12(a,b)). This proves the efficacy of shift-invariant matching, despite its high computational cost.
- When shift-invariance is enabled the TI-clustering algorithm produces shapes with the lowest error, k-means clustered shapes in fact perform the worst with the highest error, due to the excessive blurring of the shapes, while selecting shapes at random performs

[14] Lowest SSD is equivalent to lowest Euclidean distance.

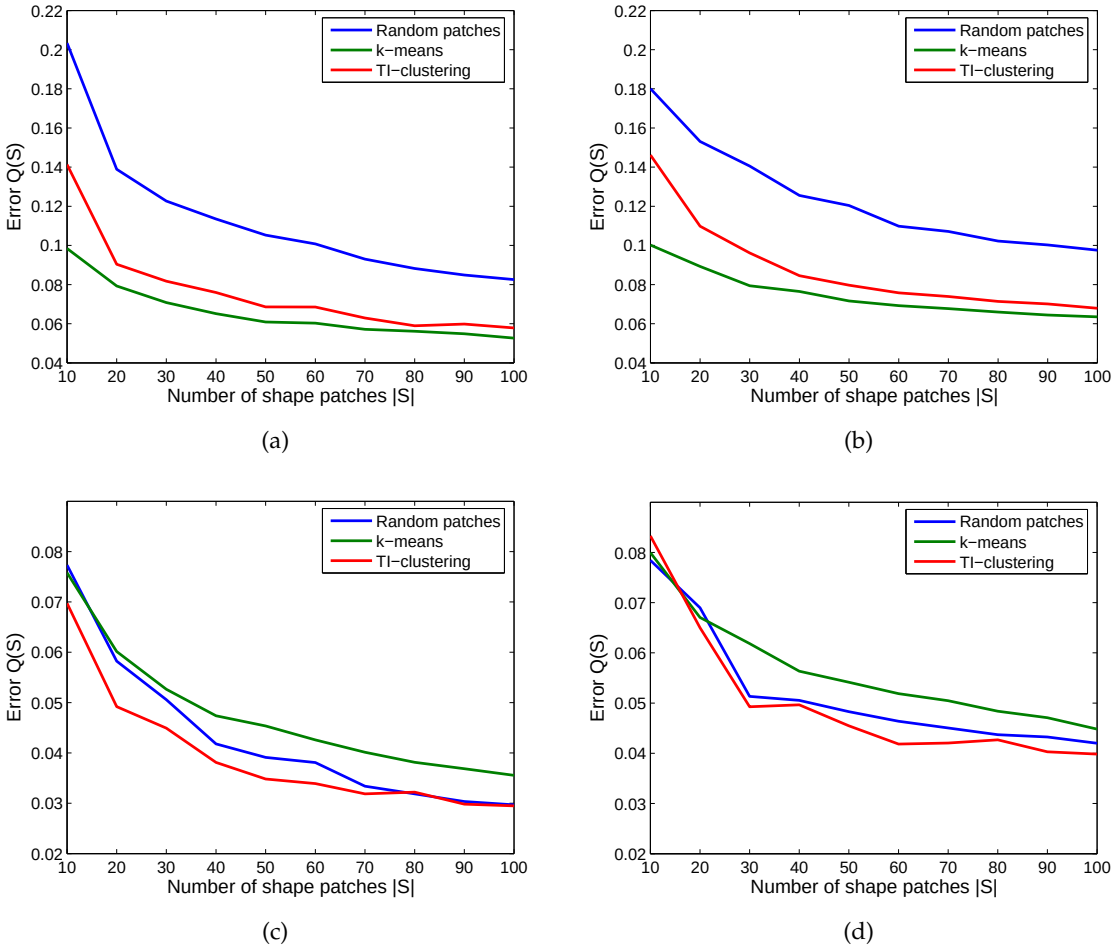


Figure 4.12: Graphs of shape patch database error $Q(S)$ as a function of number of shape patches $|S|$ in database. Blue graphs are results for randomly selected shape patches. Green graphs are results for k-means clustered shape patches. Red graphs are results for TI-clustered shape patches. Top row: shift-invariant matching disabled. Bottom row: shift-invariant matching enabled. Left column: patch size $p = 11$. Right column: patch size $p = 19$.

comparatively well. However, the most surprising result occurs when shift-invariant matching is disabled, as the ranking changes completely: random shapes perform the worst, followed by TI-clustered shapes, but simple k-means performs the *best*. This is because when shift-invariance is not used, using blurry shape patches is a good thing and does reduce the overall error; whereas, the clear, sharp edges that result from TI-clustering or selecting random patches would need to be aligned exactly to give a low error.

These results suggest that the best quality shape database results from our TI-clustering algorithm and shift-invariant matching. However, the closeness of using random patches

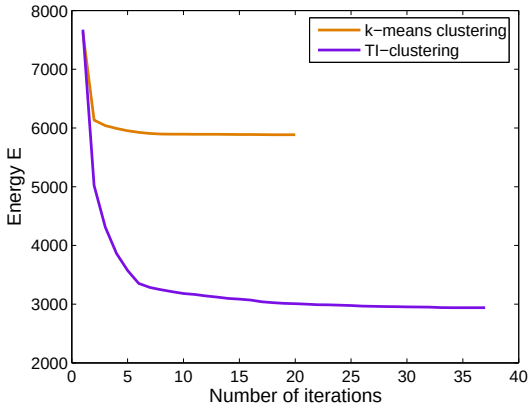


Figure 4.13: Example graph showing convergence of k-means clustering and TI-clustering algorithms.

combined with shift-invariant matching suggests that for computational efficiency using random patches might work fairly well.

Convergence of TI-clustering

Here we look briefly at the convergence of the TI-clustering algorithm, compared with k-means clustering. In fig. 4.13 we plot the total energy of the system as a function of number of iterations, up to convergence for both algorithms. The parameters we used were $q = 21$, the number of random mask patches was $K_M = 1000$, and the number of clusters was $K_S = 30$. Note that k-means clustering converges faster, but to a much worse local-minimum than TI-clustering, since the latter must estimate translational offsets as well as cluster memberships for each input patch.

Class specificity

We briefly examine the class specificity of the shape model by learning the shape database from a set of training images of horses for a range of grid sizes ($p \in \{7, 11, 15, 19\}$), and validating against three classes of image: horses, cows, and cars. Fig. 4.14 shows the graphs of error $Q(S)$ against p: (a) shows the errors obtained when shift-invariance is disabled, and (b) shows the errors obtained when shift-invariance is enabled. For this experiment $K_N = 2000$ and $K_S = 40$. TI-clustering was used to train the shape patches.

The graphs show a clear increase in error with grid size for each class, as one would expect. Also, as expected, enabling shift-invariance greatly reduces the error. However,

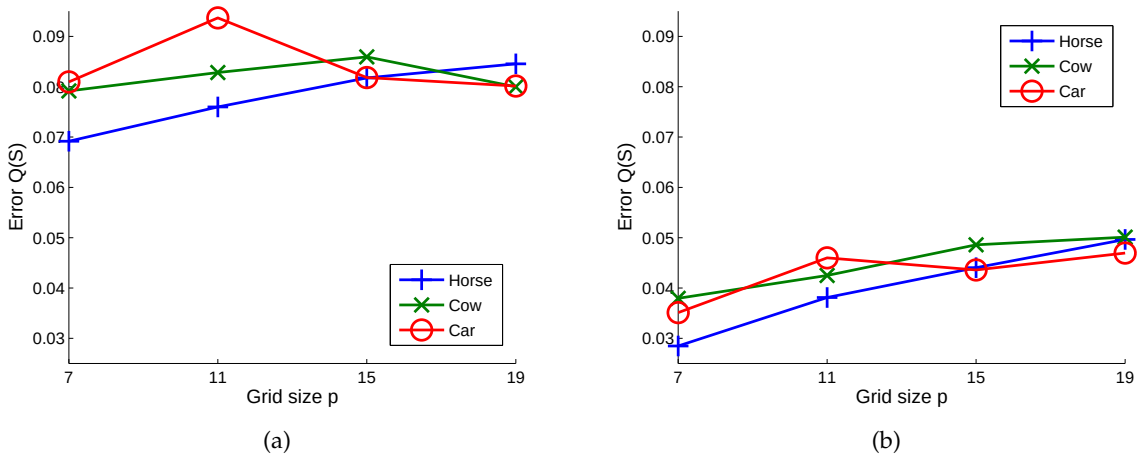


Figure 4.14: Graphs of errors for cross-validation. Shape patches were trained on horse image masks and evaluated against horse, cow and car image masks. (a) Evaluation with shift-invariant matching disabled. (b) Evaluation with shift-invariant matching enabled.

surprisingly, the cross-class difference in error seems to shrink as p increases, and in fact the horse shapes trained for larger values of p seem to represent cars and cows better than horses!

We believe this is caused by two things. Firstly, real structure in the image masks exists at several different scales and limiting our patches to only one size tends to make the clustering find structures at that particular scale. Indeed when we looked at the clusters generated with $p = 19$ and shift-invariance enabled, there were no 'leg' shapes were found since they were too small scale to be picked up, but a good 'head' shape was obtained, and for $p = 7$ just the opposite was found. This highlights the need for some form of scale invariance to be built into the system as future work. Secondly, the error measure does not take into account the complexity of the shapes being examined; a limiting case example would be the class of squares, which could be perfectly represented by only 8 shapes (edges and corners), but would have low error with shapes trained on almost any class that had some straight edges.

It is clear that the *patches* that have been learned are not very class-specific. However, this evaluation has not looked at the learned neighbourhood adjacency likelihoods which may improve the discrimination between classes. As it stands though, this work is not so immediately concerned with object recognition; as long as the resulting segmentation is good, it is not a high priority to ensure class specificity.

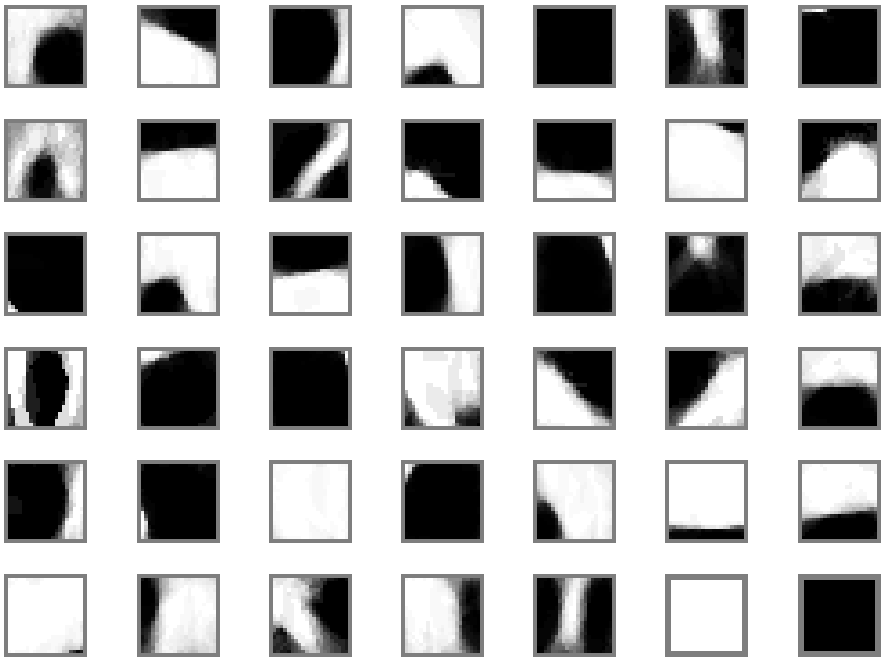


Figure 4.15: The final shape patches to be used with shift-invariant matching. A set of $K_N = 2000$ random patches is clustered with our TI-clustering algorithm to a set of $K_S = 40$ shape patches, with $p = 9$ and $b = 2$, and so the width of each is $q = 2(p + b) - 1 = 21$ pixels.

Final shapes

Based on the results presented above we have chosen to use a shape patch database of $K_S = 40$ with $p = 9$ and $b = 2$, learned with TI-clustering, and applied with shift-invariant matching. This gives a fairly low error on the training data compared with computational expense. The resulting shape patches from the TI-clustering algorithm are shown in fig. 4.15.

4.8.2 Testing the Appearance Model

To evaluate the several types of appearance model we might consider, it is necessary to specify a quality metric. While we do not use a discriminative framework in our actual segmentation algorithm, it is clearly a good idea for 'correct' shape hypotheses to be able to generate appearances that match better than the best appearances that can be generated by 'incorrect' shapes.

First then, it is necessary to define correct shapes. Using the set of shapes defined above,

we divide the validation images up into a regular grid, following our standard technique, and search for the sub-shape $\mathbf{c}_g^j$ in the database which matches best (i.e. has lowest SSD score) to the ground truth mask patch from validation image mask M_j^V and grid square g, just as we did in testing the shape model:

$$(\mathbf{s}_g^j, \mathbf{x}_g^j) = \arg\min_{\mathbf{s}\in S} \min_{\mathbf{x}_i} \|\mathbf{m}_g^j - \text{subwindow}_{p\times p}(\mathbf{s}, \mathbf{x}_i)\|^2 \quad (4.42)$$

$$\mathbf{c}_g^j = \text{subwindow}_{p\times p}(\mathbf{s}_g^j, \mathbf{x}_g^j) \quad (4.43)$$

We then choose to define *in*correct shapes $\mathbf{i}$ as those that have an SSD score to the real shape that is greater than 10% more than the correct shape:

$$1.1 * \|\mathbf{m}_g^j - \mathbf{c}_g^j\|^2 < \|\mathbf{m}_g^j - \mathbf{i}\|^2 \quad (4.44)$$

Those shapes $\mathbf{s}$ that are not the correct shape but still not too different, i.e. $\|\mathbf{m}_g^j - \mathbf{c}_g^j\|^2 < \|\mathbf{m}_g^j - \mathbf{s}\|^2 \leq 1.1 * \|\mathbf{m}_g^j - \mathbf{c}_g^j\|^2$, are simply ignored for this evaluation.

It is now fairly trivial to generate the best matching appearance our appearance model permits, as described in §4.5.2, both for correct shapes and incorrect shapes. We hope that the best match for the correct shape will on average have a much better score than the best matches for incorrect shapes. We collect matching scores for each validation image and each grid square, for both the best appearance that can be generated using the correct shape, and also the *best* appearance that can be generated using a random selection from the database of incorrect shapes. Note that we do not expect these two classes to be very distinguishable due to the inherent ambiguity in appearance; but however we can aim to maximise their separation to help ourselves somewhat.

We tested two matching measures: sum-of-squared differences (SSD) and normalised cross correlation (NCC). The relationship of these two, and how to calculate these correctly when shapes are used to mix appearances, is described in Appendix C. We also tested three colour spaces: red-green-blue (RGB), luminance-chrominance (YUV), and grey-scale (the Y, luminance, channel from YUV). For each we collect matching scores, from which histograms of score frequency can be generated and normalised to produce approximate probability densities. In addition, we can plot ROC curves to compare each scheme.[15]

[15] Receiver-operator characteristic (ROC) curves plot the sensitivity (true-positive rate) against $(1-\text{specificity})$ (false-positive rate).

Results are shown in fig. 4.16. These were generated using the shape patch database learned above, a set of foreground appearance patches learned from the training images, and for each image separately a set of background appearance patches learned from the validation image based on the mask. For most of the histograms plotted in fig. 4.16, there is a clear separation between the distributions of scores for correct and incorrect shapes. The plots where this is not the case are (b) for grey-scale NCC, and (d) for YUV NCC. The latter is simply the resolution of the histogram. The former can be explained by the fact that NCC tries to make the two patches as similar as possible in terms of mean and variance; by construction, most generated midground patches will have very low variance within the foreground regions and within the background regions, which are matched *separately* with NCC, and consequently there will be little to discriminate correct and incorrect shapes.

From the raw matching scores we created ROC curves for each of the six matching functions. These are shown in fig. 4.17. The best curves are obtained from using YUV and RGB colour spaces with the SSD measure, the worst, for reasons explained above, from grey-scale and NCC.

4.8.3 Segmentation Results

We present in figs. 4.18-4.25 the final segmentation results of the validation images given by our algorithm, ranked from best to worst. The results were generated with a database of shapes learned for $p = 9$ with TI-clustering (as shown in fig. 4.15), and hypotheses generated in our shift-invariant manner. The foreground and background appearance patch databases were generated by k-means clustering and we used the RGB colour space. The foreground appearance model was learned exclusively from the training data, not using the foreground area of the trimaps. Background appearance models were learned for each test image using the relevant trimap, as justified in §4.2.1. The variance parameters were set to $\sigma_i = 0.05$, $\sigma_s = 0.3$ and $\sigma_a = 0.15$.

To quantify the performance we evaluated two measures. For both, the resulting MMSE segmentation (as defined by eq. 4.30) is thresholded at a value of 0.5 to convert the output into a binary-valued segmentation mask. For the measure used by Borenstein & Ullman, $r = \frac{|S \cap F|}{|S \cup F|}$ where F is the ground-truth segmentation and S is the thresholded segmentation output, our algorithm gave a mean value of r over the whole validation set of $\bar{r} = 66.2\%$ which compares reasonably to the stated result of Borenstein & Ullman given that we could not repeat their results. The best result gave a value of $r = 94.7\%$. Alternatively, if we look

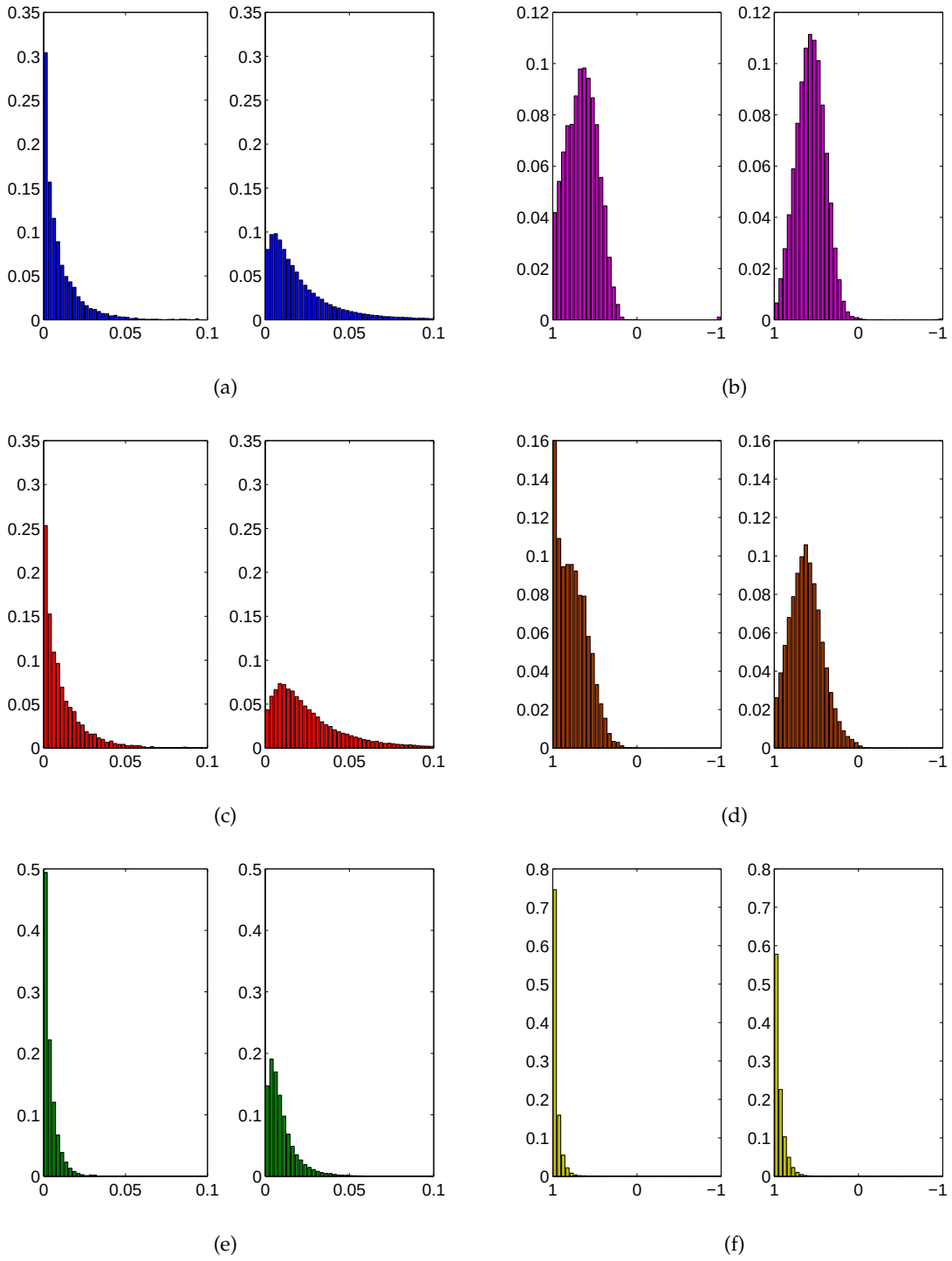


Figure 4.16: Normalised histograms of matching score frequency for correct and incorrect shapes. In each pair, the left graph shows the distribution of matching scores for correct shapes, and the right graph shows the distribution for incorrect shapes. (a,c,e) show scores for SSD. (b,d,f) show scores for NCC. Top row uses grey-scale only; middle row uses RGB colour; bottom row uses YUV colour. In each graph, the horizontal represents matching score and the vertical represents normalised frequency (i.e. probability).

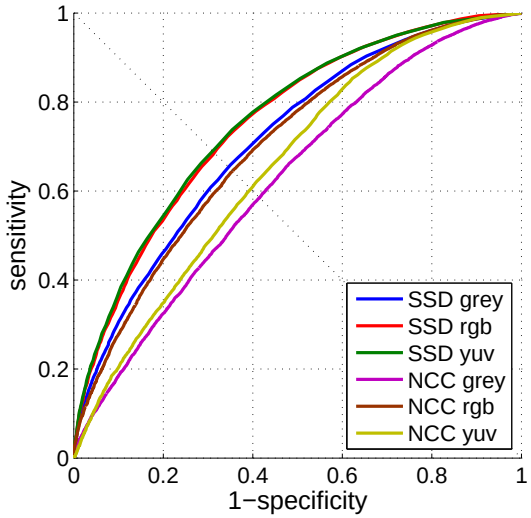


Figure 4.17: ROC curves plotted for each pair of histograms in fig. 4.16. Equal error rates can be found along the dashed diagonal line.

at the proportion of correctly classified pixels $c = \frac{|S \cap F| + |S' \cap F'|}{S \cup S'}$ where S' and F' are copies of S and F respectively with the mask values reversed, we get a score of $\bar{c} = 88.2\%$ successful segmentation classification for the whole validation set, the best result giving $c = 98.0\%$.

A good example of where a bottom-up segmentation algorithm would struggle is the top row of fig. 4.19, where the white back of the horse merges into the background and yet our algorithm has learned to continue the horses back despite very poor information in the image itself.

The method seems to work well for about half the example images. The main problems seem to be highly textured horses, where the foreground appearance model learned from the database is not sufficient to cope with all observed variation, probably because the training data are not sufficiently general in appearance. We tried learning foreground appearance models from the test images themselves (using the trimaps), and got slightly better results, but it was considered too great a restriction to require the foreground of test images to be roughly labeled when we can obtain almost as good results without.

4.9 Summary and Conclusions

In this chapter we have presented our research into class specific segmentation. The method learns models for local shape and appearance from a labeled set of training data. Given a novel test image, the algorithm generates local hypotheses of shape and appearance that

Figure 4.18: Final segmentation results, ranked 1 (top), 2 (middle) and 3 (bottom). First column is input image; second column is generated appearance; third column is ground-truth segmentation; fourth column is our segmentation result. Note third and fourth columns are given a grey border to distinguish foreground at the edge of the image.

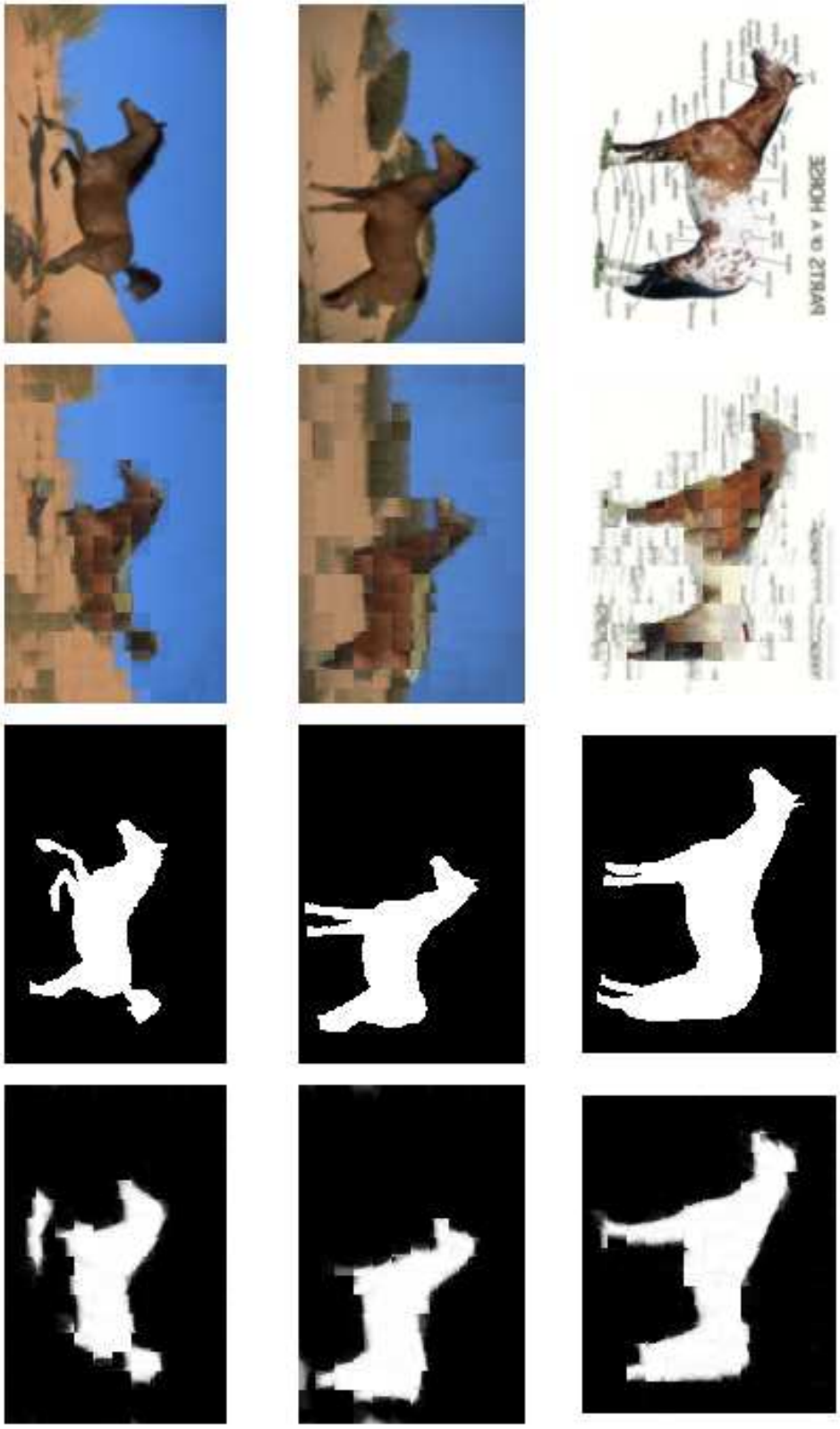


Figure 4.19: Final segmentation results, ranked 4, 5 and 6. Columns as in fig. 4.18.

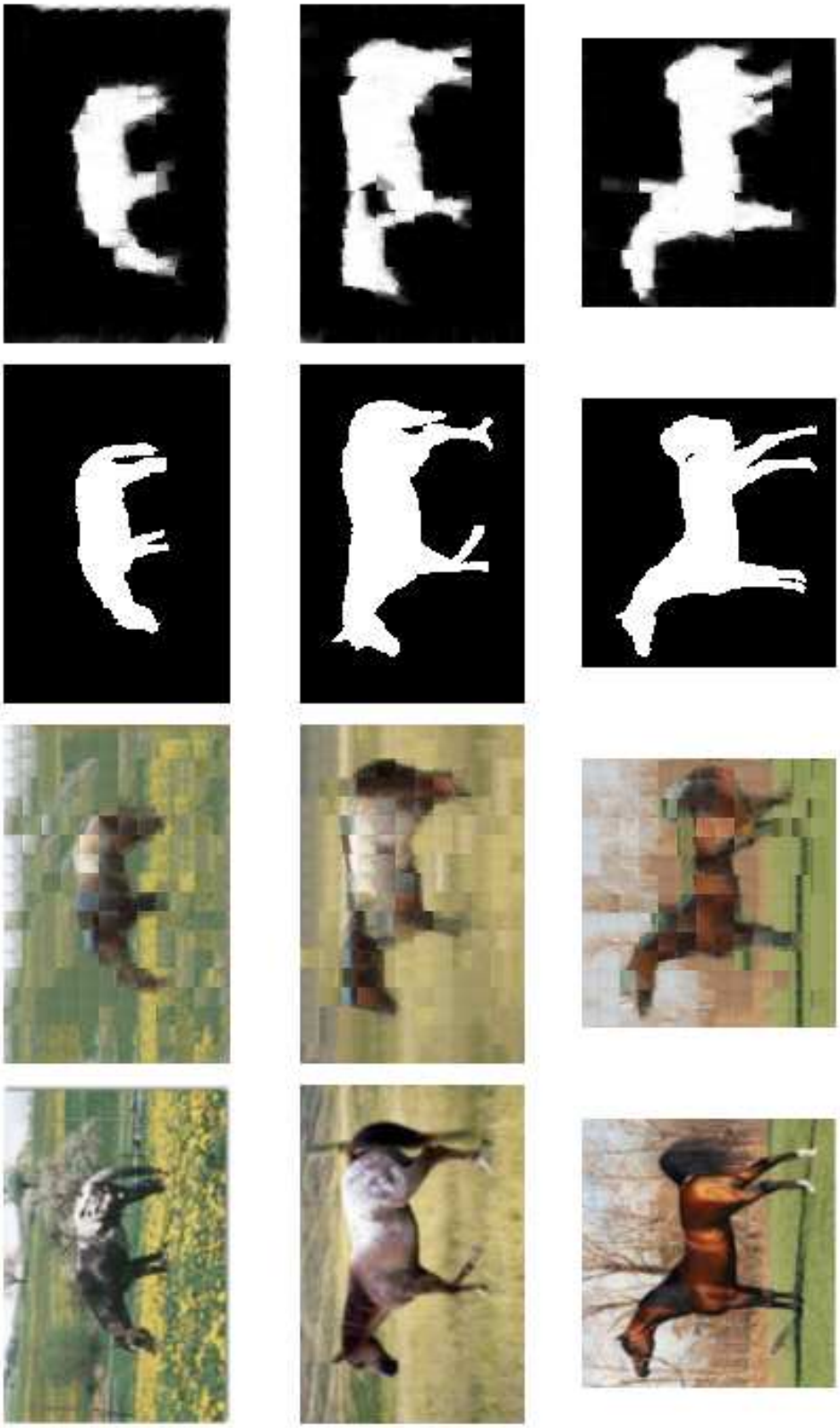


Figure 4.20: Final segmentation results, ranked 7, 8 and 9. Columns as in fig. 4.18.

Figure 4.21: Final segmentation results, ranked 10-18. Columns as in fig. 4.18.

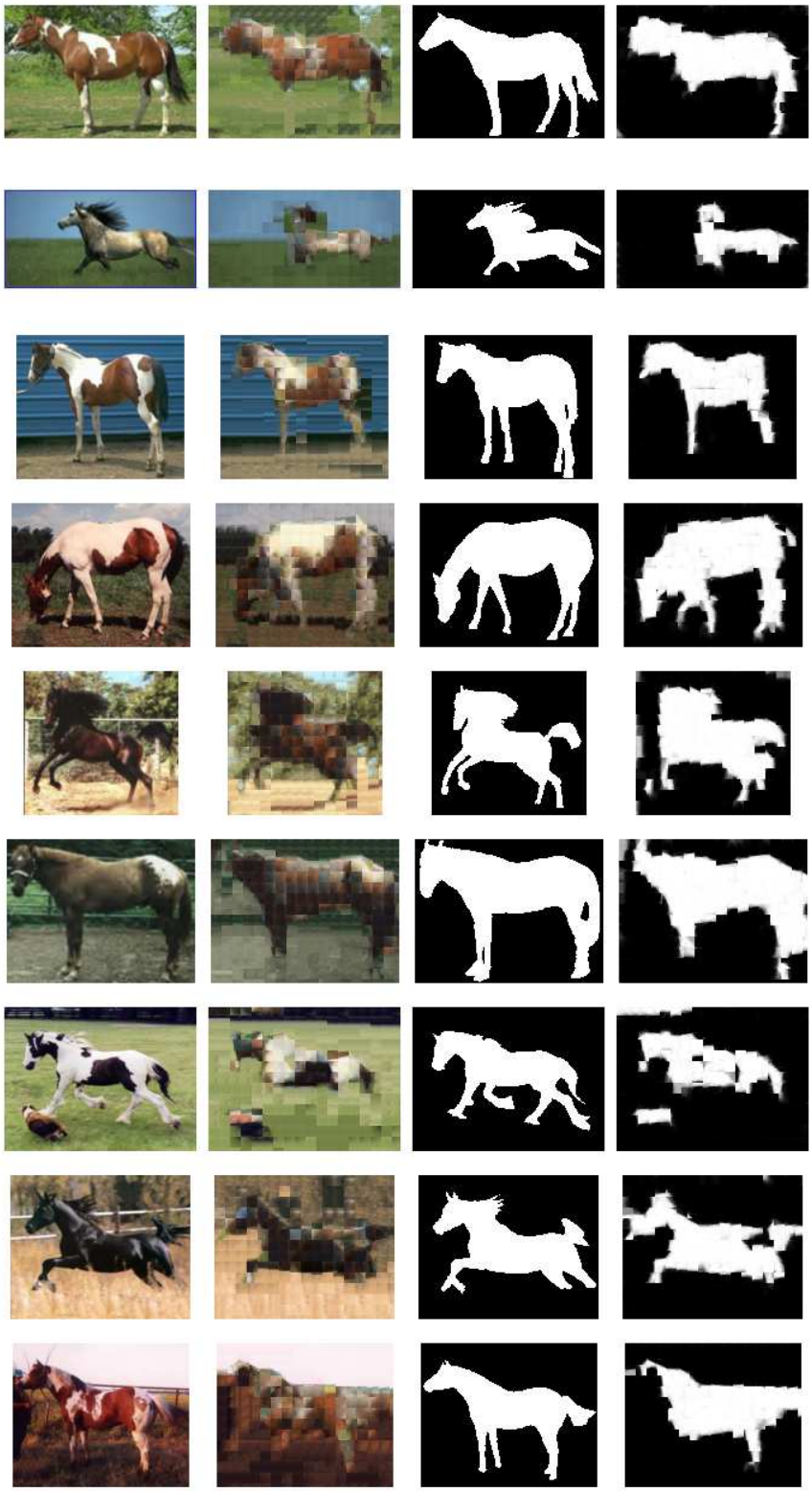


Figure 4.22: Final segmentation results, ranked 19-27. Columns as in fig. 4.18.

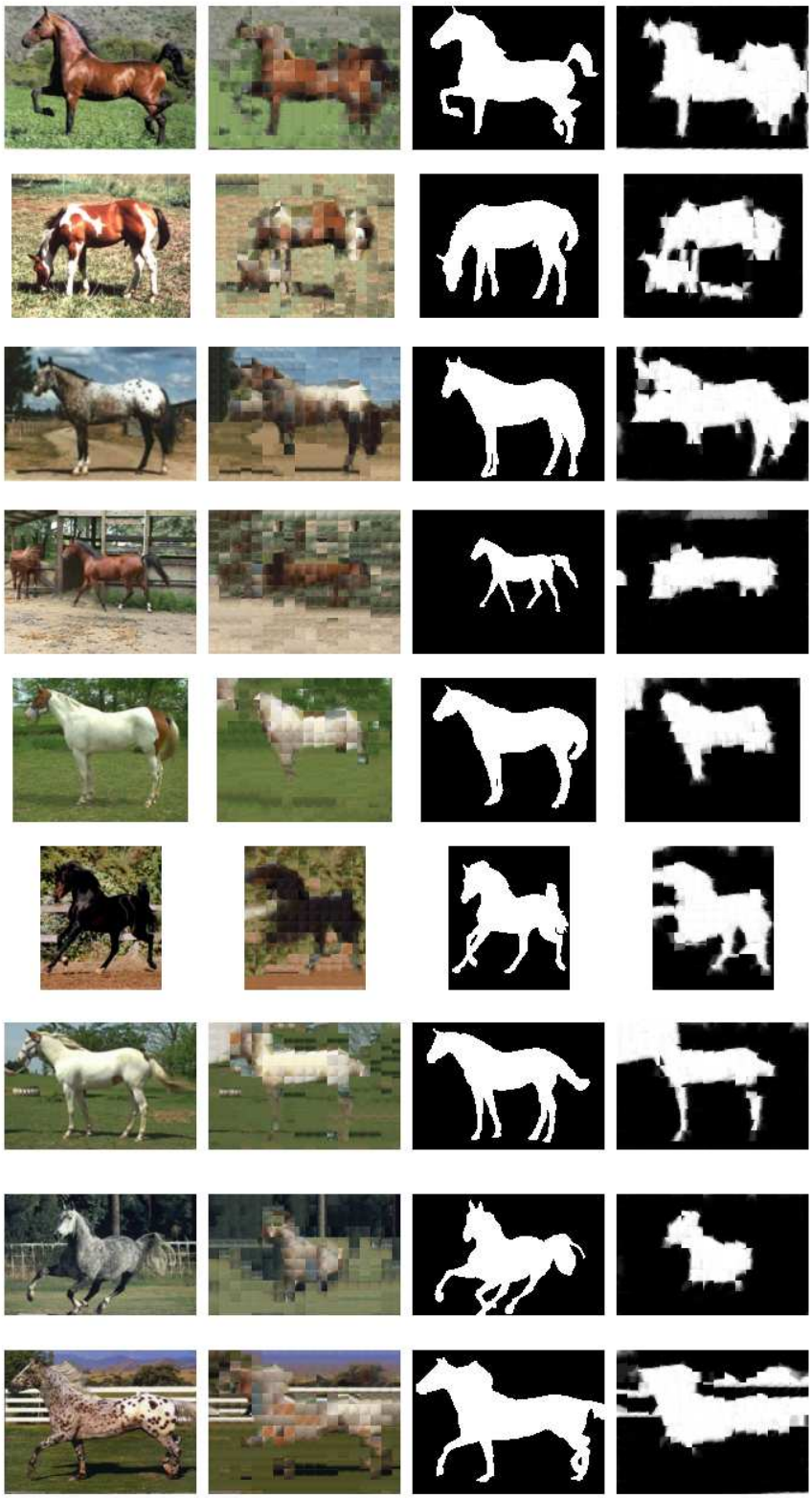


Figure 4.23: Final segmentation results, ranked 28-36. Columns as in fig. 4.18.

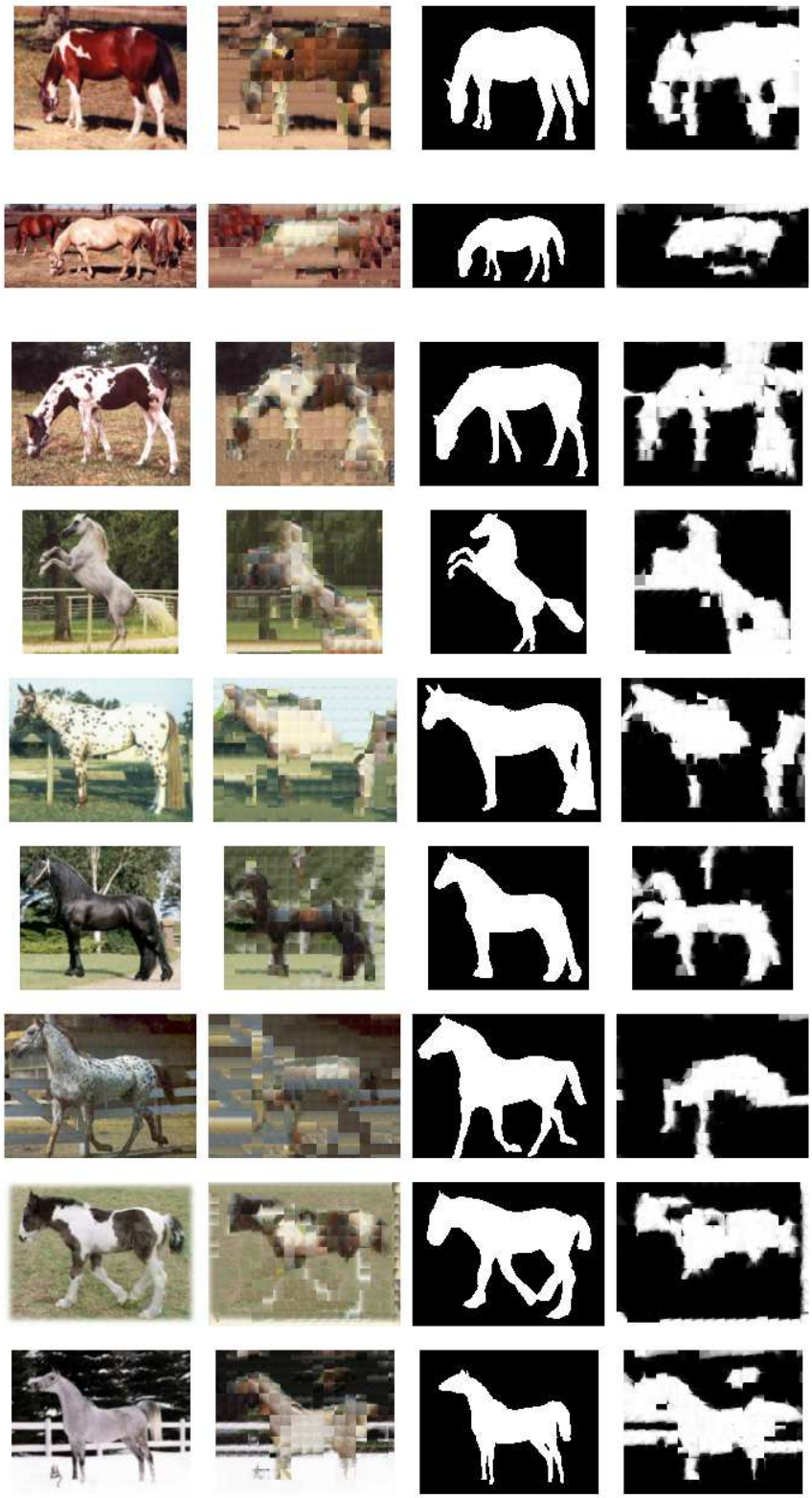


Figure 4.24: Final segmentation results, ranked 37-45. Columns as in fig. 4.18.

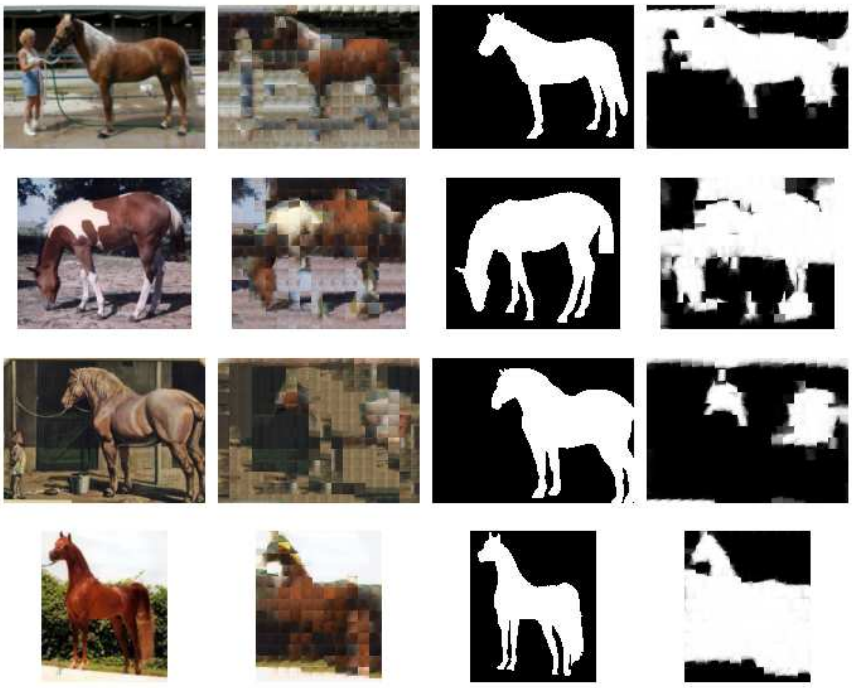


Figure 4.25: Final segmentation results, ranked 46-end. Columns as in fig. 4.18.

cover the image. The final stage uses belief propagation to search for a globally consistent figure-ground segmentation, which can be seen as piecing together a jigsaw puzzle where the pieces are our hypotheses.

We have evaluated our shape and appearance models as well as the whole segmentation algorithm and got fairly good results. Our shape model is learned from a modified k-means clustering algorithm called TI-clustering which gave the lowest error in our comparison under certain conditions. Our appearance models are a set of clustered patches for which the RGB colour space gave good results. We have shown some impressive segmentations despite highly textured foregrounds and backgrounds similar to foregrounds.

CHAPTER 5

CONCLUSIONS AND FUTURE DIRECTIONS

5.1 Conclusions

We have presented in this report our investigations into class-specific segmentation. After a brief summary of relevant literature, we examined in detail the work of Borenstein & Ullman ([9, 10]), and were unable to repeat their results convincingly. This motivated our new approach which used belief propagation to ensure global consistency from a set of local hypotheses of shape and appearance.

While we obtained fairly good results as presented, there are still several problems that need addressing in future work. Firstly it seems that our paradigm of dividing the image into a regular grid imposes several severe constraints:

- Shift-invariant matching must be used to ensure generality of our shape patch database. This is problematical purely for efficiency purposes: instead of generating the best matching appearance per shape, this must be done per shape and per shift, multiplying the computation by a factor of $(p-1)^2$.

- The patches cannot be superimposed. This presents a problem for example on the legs of horses which are often close together. Unless the shape patch database contains good examples of leg shapes close together, our algorithm cannot generate good shapes. Ideally, one should be able to fit arbitrary sized shapes together in arbitrarily overlapping positions, so that limbs could be modeled independently.

- The shape patches must all be the same scale. Clearly this is not a good restriction as we discovered in our evaluation: for small values of p (corresponding to small scales),

good leg shapes we obtained, but no head shapes; conversely, for large values of p corresponding to large scales, a good head shape was obtained but no leg shapes.

Additionally, our experiments have shown that the class-specificity of our shape patch database is fairly low: to a considerable degree, the shapes that we have learned are so local as to just restrict curvature, and the neighbourhood adjacency probabilities (ψ^{model}) are currently fairly weak, requiring the additional overlap likelihood (ψ^{overlap}) for good results.

5.2 Suggestions for Improvements

In light of these problems, we should like to improve the technique in the following areas:

- Shape Model:

  We wish to improve the shape model so that it can represent structure at multiple scales, and is possibly rotation and affine invariant. Clearly, for arbitrary objects, local shape does not occur at only one scale, and we must therefore extend our shape model to incorporate this.

  Additionally we should aim to learn a more global notion of shape, perhaps in a similar vein to the Constellation model of [23], but ideally without the exponential cost in the number of parts. In [21] the Pictorial Structures model is presented, but this requires a fixed known shape, as does [65]. We would like to learn a global model for shape automatically from our ground-truth segmentations. As far as we are aware this has not been done.

- Appearance Model:

  Currently our appearance model is a simple database of clustered patches. We should hope to apply some dimensionality reduction methods such as PCA to reduce computational cost. We should also like to incorporate notions of uncertainty and learn appearance likelihoods properly. One approach might be to use the Epitome images of [35] which learns a compact image representation that could effectively be used as an appearance database that copes well with irregular patch shapes and shift-invariance.

- Guided Search:

  Our current search technique generates hypotheses in a brute-force manner, trying every possible combination of shape and appearance, and is therefore very expensive. Perhaps a method to guide hypothesis generation based on image features

could be used to ameliorate this.

There has been recent success in applying interest point detectors to segmentation ([41]), so perhaps these could be used, at least for an initial estimate of location. Alternatively, perhaps a coarse-to-fine subdivision of the image would be useful: good hypotheses at a coarse scale guide the choice at finer scales.

In areas of uncertainty, the search should be guided by areas of more certainty, and knowledge of this should be made available as output in the form of an uncertainty map, telling us where the algorithm has weaker information in the image and is relying on its class prior.

- Using Bottom-Up Cues:

  We are not yet using bottom-up cues for segmentation. For example, edges could be combined with chamfer matching (see e.g. [70]) at a local scale, or colour likelihood models could be learned for appearance (e.g. a Gaussian Mixture Model as in [7]).

  Ideally we should like to be able to combine the top-down segmentation techniques of the previous chapter with bottom-up segmentation techniques such as [11]. Borenstein et al present such a fusion in [8]. However, in their formulation, the bottom-up segmentation must be over-complete and the top-down segmentation cannot over-rule this. One should instead aim for a fully probabilistic approach where no hard decisions are made in favour of either top-down or bottom-up cues.

- Inter-Class Discriminability:

  We have not focused particularly on ensuring inter-class discriminability. While not necessarily very important for segmentation, it becomes very important for image classification or object detection. We should hope to be able to extract from different classes different features of shape and appearance that are most discriminating, and use these to match only the correct class in an image. In one of our example results there is a dog running along side the horse which has been segmented as foreground due to its similar appearance to a horse. Clearly there will be a limit to what we can discriminate: are donkeys for example sufficiently different in appearance to discriminate from horses? At the same time, for segmentation purposes, it is not certain how important discriminability is: for example, most four-legged animals look fairly similar.

  For discriminability of the shape model, it would be constructive to create a training

database of non-class *shapes*. Our shape model has tended to learn generic shapes rather than the specific shapes that would help discriminate between classes.

- Combined Recognition and Segmentation:

  Following on from the previous point we should like not only to segment the object given knowledge of what is present in the scene, but also to estimate simultaneously the probability that the image contains a particular class. So ideally one would train the algorithm on several classes, and presented with a new image, it would automatically choose the correct class and segment it, or reject as an unknown class. Perhaps a hierarchical segmenter or classifier could be built, e.g. mammals $\rightarrow$ horses $\rightarrow$ breed.

  Starting points to investigate this would be [42, 73].

- Evaluation:

  We shall need to extend our evaluation to include several different classes of objects for discriminability comparisons. Especially interesting to investigate would be very similar classes such as for example zebras, horses and donkeys.

  Ideally the need for trimaps should be removed; perhaps an interactive refinement scheme as in [58] would work well. Alternatively, if the background context could be determined automatically (such as forest, beach, field, etc.) then appearance models for these could be learned off-line.

  Video sequences could be used for training and testing. By segmenting video sequences, we could effectively track objects. The training sets we have used so far are not only class dependent but view dependent (e.g. horses viewed from the side; faces viewed head-on), and so to work for general sequences we would probably need to remove this restriction. Also, for video sequences, the background is usually fairly constant (or at least slowly changing) and so the background appearance model could perhaps be learned only for the first frame, largely removing the need for the trimap.

  The output from a stereo algorithm (e.g. [17]) could be used to generate rough segmentations based on depth for training data. Or, conversely, good figure-ground segmentations of left and right image pairs would greatly improve the 3-D reconstruction process for single objects by ensuring a correct disparity contour resulted.

Dates				Focus
2004	September	—	December	Investigate new shape models
2005	January	—	March	Investigate new appearance models
	April	—	June	Consider new guided search techniques
	July	—	September	Improve class-specificity of the method
	October	—	December	Investigate combined recognition and segmentation
2006	January	—	March	Begin thorough evaluation
	April	—	September	Final results & thesis write-up

Table 5.1: Timetable for future work.

5.3 PhD Timetable

We suggest an approximate timetable, shown in table 5.1, for work to be undertaken over the next two years, dividing the work into blocks of roughly three months.

APPENDIX A

TRAINING AND VALIDATION DATA

A.1 Towards Improved Class-Specific Segmentation

To evaluate our method in Chapter 4, we used the training and validation data presented in fig. A.1, consisting of a set of 50 colour images of horses viewed side-on and at roughly the same scale. Each image has an associated ground-truth segmentation mask image that has been created by hand. The validation data, shown in fig. A.2, consist of a similar set of 50 colour images with their masks. The validation data also has associated trimaps as discussed in §4.2.1.

The horse images were obtained from Borenstein's website.[1] However we hand-segmented the ground-truth mask images, as those provided did not seem to align with the appearance images. The images were generated from an original set of images at higher resolution by down-sampling with a suitable Gaussian blur followed by bilinear filtering to roughly 160×120 pixels, to ensure a reasonable computational load and rough agreement in scale.

A.1.1 Colour Spaces

For our evaluation we investigated three colour spaces: RGB, YUV, and grey-scale. The standard computer image colour space is RGB, so to obtain the corresponding YUV representation the following transformation was applied:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.419 & 0.500 \\ 0.500 & -0.419 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{A.1}$$

[1] http://www.wisdom.weizmann.ac.il/~boren/segmented_horses/.

Figure A.1: The training data: colour images with ground-truth segmentation.

Figure A.2: The validation data: colour images with ground-truth segmentation and trimaps. For the trimaps, red indicates foreground and blue indicates background. Note in our technique we only actually use the blue regions to build background appearance models, and the red foreground regions are ignored.

A.2 Borenstein & Ullman: Fragment Based Approaches

To evaluate Borenstein & Ullman's fragment based algorithms in Chapter 3, we used the data presented in this section. For as fair an evaluation as possible, the training and test images were largely the same as above, obtained from Borenstein's website. The background image set was obtained from the Caltech image archive which is apparently the same as Borenstein & Ullman used.[2] For this evaluation, the high resolution images with their corresponding hand-segmentations were down-sampled to 80×60 pixels or thereabouts, preserving aspect ratio.[3] All images were treated as grey-scale images, i.e. colour information was thrown away, by taking the Y (luminance) channel of the YUV transformed images.

The 188 down-sampled horse images are shown in fig. A.3, with corresponding hand-segmentations for the first 40 images in fig. A.4. The first 200 down-sampled non-class images, from a total set of 451, are shown in fig. A.5. Note that the hand-segmentations for non-class images would all be black, i.e. no foreground regions.

[2] `http://www.vision.caltech.edu/html-files/archive.html`.

[3] This is a quarter the resolution used for our technique above. In personal correspondence, Borenstein suggested that his technique would not scale well to higher resolutions.

Figure A.3: Class image set.

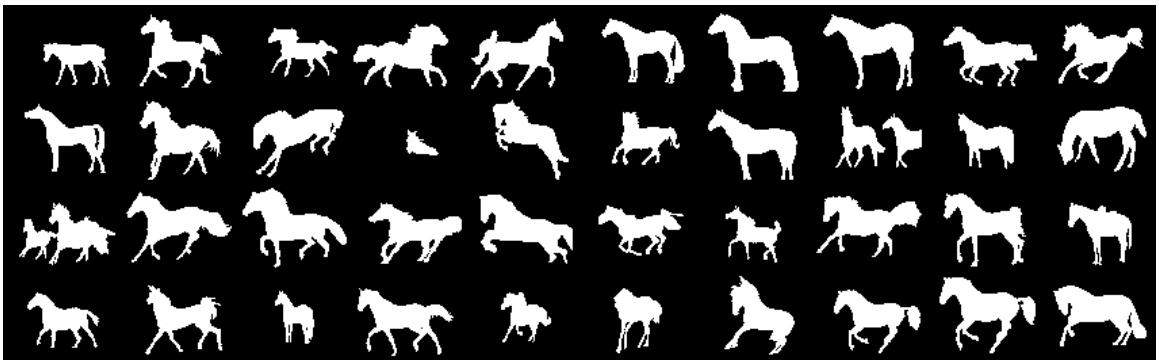


Figure A.4: Hand-segmented ground-truth for class images.

Figure A.5: Non-class image set.

APPENDIX B

THE SECOND MOMENT MATRIX

We present here definitions of scale-space, Affine Gaussian scale-space, and the Second Moment Matrix, all of which are used in [45, 46, 3, 53], which we describe in §2.1.6. We present more detail than in the literature, standardising the notation somewhat and filling in derivations and proofs that are not present. At the end of this appendix we also provide a proof that these iterative schemes are not guaranteed to converge.

B.1 Scale-Space

The scale-space of an image is the image represented at different resolutions. This is in general created by convolving the image I, with Gaussian filters G, at several scales s:

$$L(\mathbf{x}, s) = \mathcal{G}(s) * I(\mathbf{x}) \tag{B.1}$$

where $\mathbf{x} = (x, y)$ is the coordinate frame of the image. When building a scale space, to maintain a uniform information change between successive levels, the scale factor should be distributed geometrically: $s_n = k^n s_0$. Features such as corners and edges can then be detected at different resolutions by applying appropriate derivative-based functions at different scales.

The amplitude of spatial derivatives in general decreases with scale, since they are averaged over a larger area. To compensate for this and thereby maintain scale invariance, derivative-based functions should be normalised with respect to the scale of observation. Using the following notation for derivatives of an arbitrary function $\bullet$:

$$\bullet_{i_1,\ldots,i_m} = \frac{\partial^m \bullet}{\partial i_1 \ldots \partial i_m} \tag{B.2}$$

they define the scale normalised derivative D of order m as:

$$D_{i_1,\ldots,i_m}(\mathbf{x}, s) = s^m L_{i_1,\ldots,i_m}(\mathbf{x}, s) = s^m \mathcal{G}_{i_1,\ldots,i_m}(s) * I(\mathbf{x}) \tag{B.3}$$

These normalised derivatives behave well under scaling of intensity patterns. Consider two images I and I' imaged at different scales such that $I(\mathbf{x}) = I'(\mathbf{x}')$ where $\mathbf{x}' = t\mathbf{x}$. Image derivatives are then related by:

$$s^m \mathcal{G}_{i_1,\ldots,i_m}(s) * I(\mathbf{x}) = (ts)^m \mathcal{G}_{i_1,\ldots,i_m}(ts) * I'(\mathbf{x}') \tag{B.4}$$

Thus we obtain:

$$D_{i_1,\ldots,i_m}(\mathbf{x}, s) = D'_{i_1,\ldots,i_m}(\mathbf{x}', ts) \tag{B.5}$$

and hence the scale normalised derivative is invariant with respect to scale.

B.2 Affine Gaussian Scale-Space

A natural generalisation of the isotropic scale-space defined above is to allow the Gaussian filter to have a full (symmetric positive semi-definite) covariance matrix. We can now look at the image at anisotropic resolutions by convolving it with 2-D Gaussians:

$$\mathcal{G}(\mathbf{x}; \Sigma) = \frac{1}{2\pi\sqrt{\det \Sigma}} \exp(-\frac{1}{2}\mathbf{x}^T \Sigma^{-1} \mathbf{x}) \tag{B.6}$$

If $\Sigma = tI$ for some scalar t and the identity matrix I, then this corresponds to the isotropic Gaussian as above.

Given an intensity image $I : \mathbb{R}^2 \to \mathbb{R}$, the affine Gaussian scale-space representation of I is then

$$L(\mathbf{x}; \Sigma) = \mathcal{G}(\mathbf{x}; \Sigma) * I(\mathbf{x}) \tag{B.7}$$

B.2.1 Linear Transformation Properties

Theorem B.1 The affine Gaussian scale-space is closed under linear (e.g. affine) transformations of the coordinates. Let I_L and I_R be two images such that at point $\mathbf{u}_0$ in I_L and point $\mathbf{v}_0$ in I_R, the local neighbourhoods are related by a linear transformation $\mathbf{v} = A\mathbf{u}$:

$$I_L(\mathbf{u}) = I_R(A\mathbf{u}) = I_R(\mathbf{v}) \tag{B.8}$$

Writing the affine Gaussian scale-space representations of the two images as

$$L(\mathbf{u};\Sigma_L) = \mathcal{G}(\mathbf{u};\Sigma_L) * I_L(\mathbf{u}) \tag{B.9}$$

$$R(\mathbf{v};\Sigma_R) = \mathcal{G}(\mathbf{v};\Sigma_R) * I_R(\mathbf{v}) \tag{B.10}$$

then L and R are related by

$$L(\mathbf{u};\Sigma_L) = R(\mathbf{v};\Sigma_R) \tag{B.11}$$

where

$$\Sigma_R = A\Sigma_L A^T \tag{B.12}$$

Hence for any matrix Σ_L there exists a matrix Σ_R such that the affine Gaussian scale-space representations of I_L and I_R are equal.

Proof *Expanding the convolution of L at point* $\mathbf{u}_0$*:*

$$\begin{aligned} L(\mathbf{u}_0;\Sigma_L) &= \int_{\mathbf{u}} \mathcal{G}(\mathbf{u}_0-\mathbf{u};\Sigma_L) I_L(\mathbf{u}) \mathrm{d}\mathbf{u} && \text{(B.13)} \\ &= \int_{\mathbf{u}} \frac{1}{\det A} \mathcal{G}(A^{-1}(A\mathbf{u}_0 - A\mathbf{u});\Sigma_L) I_R(A\mathbf{u}) \mathrm{d}(A\mathbf{u}) && \text{(B.14)} \end{aligned}$$

since $\mathbf{v} = A\mathbf{u}$ *implies* $\mathrm{d}\mathbf{v} = \det A \mathrm{d}\mathbf{u}$*. Now, under a linear transformation, the anisotropic Gaussian kernel transforms as*

$$\begin{aligned} \mathcal{G}(A^{-1}\mathbf{z},\Sigma_L) &= \frac{1}{2\pi\sqrt{\det\Sigma_L}} e^{\frac{1}{2}(A^{-1}\mathbf{z})^T\Sigma_L^{-1}(A^{-1}\mathbf{z})} && \text{(B.15)} \\ &= \frac{\sqrt{\det AA^T}}{2\pi\sqrt{\det A\Sigma_L A^T}} e^{\frac{1}{2}\mathbf{z}^T(A\Sigma_L A^T)^{-1}\mathbf{z}} && \text{(B.16)} \end{aligned}$$

giving us

$$\mathcal{G}(A^{-1}\mathbf{z},\Sigma_L) = \det A \mathcal{G}(\mathbf{z}, A\Sigma_L A^T) \tag{B.17}$$

Inserting eq. B.17 into eq. B.14 with $\mathbf{z} = A\mathbf{u}_0 - A\mathbf{u}$ *and* $\mathbf{v}_0 = A\mathbf{u}_0$*:*

$$\begin{aligned} L(\mathbf{u}_0;\Sigma_L) &= \int_{\mathbf{u}} \frac{\det A}{\det A} \mathcal{G}(A\mathbf{u}_0 - A\mathbf{u}; A\Sigma_L A^T) I_R(A\mathbf{u}) d(A\mathbf{u}) && \text{(B.18)} \\ &= \int_{\mathbf{v}} \mathcal{G}(\mathbf{v}_0 - \mathbf{v}; A\Sigma_L A^T) I_R(\mathbf{v}) d\mathbf{v} && \text{(B.19)} \\ &= R(\mathbf{v}_0; A\Sigma_L A^T) && \text{(B.20)} \end{aligned}$$

□

B.3 Second Moment Matrix in Affine Gaussian Scale-Space

Extending the isotropic second moment matrix defined in §2.1.5 as $C(\mathbf{x}, s_{\mathrm{i}}, s_{\mathrm{d}})$, we define the second moment matrix based on *anisotropic* smoothing for I_L and I_R as respectively:

$$\mu_L(\mathbf{u}; \Sigma_{L,\mathrm{i}}, \Sigma_{L,\mathrm{d}}) = \mathcal{G}(\mathbf{u}; \Sigma_{L,\mathrm{i}}) * ((\nabla L(\mathbf{u}; \Sigma_{L,\mathrm{d}}))(\nabla L(\mathbf{u}; \Sigma_{L,\mathrm{d}}))^T) \tag{B.21}$$

$$\mu_R(\mathbf{v}; \Sigma_{R,\mathrm{i}}, \Sigma_{R,\mathrm{d}}) = \mathcal{G}(\mathbf{v}; \Sigma_{R,\mathrm{i}}) * ((\nabla R(\mathbf{v}; \Sigma_{R,\mathrm{d}}))(\nabla R(\mathbf{v}; \Sigma_{R,d}))^T) \tag{B.22}$$

with $\Sigma_{\{L,R\},\mathrm{i}}$ and $\Sigma_{\{L,R\},\mathrm{d}}$ the integration scale and differentiation scale covariance matrices respectively.

B.3.1 Transformation under Linear Transformations

Lemma B.2 With the same linear transformation between the two images as before, we have:

$$\mu_L(\mathbf{u}; \Sigma_{L,\mathrm{i}}, \Sigma_{L,d}) = A^T \mu_R(A\mathbf{u}; A\Sigma_{L,\mathrm{i}}A^T, A\Sigma_{L,\mathrm{d}}A^T)A \tag{B.23}$$

Proof *First we show that*

$$\nabla L(\mathbf{u}; \Sigma_L) = A^T \nabla R(A\mathbf{u}; A\Sigma_L A^T) \tag{B.24}$$

by writing the derivative of the Gaussian as

$$\nabla \mathcal{G}(\mathbf{z}, \Sigma_Z) = -\Sigma_Z^{-1}\mathbf{z}\frac{1}{2\pi\sqrt{\det \Sigma_Z}}e^{-\frac{1}{2}\mathbf{z}^T\Sigma_Z^{-1}\mathbf{z}} = -\Sigma_Z^{-1}\mathbf{z}\mathcal{G}(\mathbf{z}, \Sigma_Z) \tag{B.25}$$

which means

$$\begin{aligned}
\nabla L(\mathbf{u}; \Sigma_L) &= (\nabla \mathcal{G}(\mathbf{u}, \Sigma_{L,d})) * I_L(\mathbf{u}) \\
&= ((-\Sigma_L^{-1}\mathbf{u})\mathcal{G}(\mathbf{u}, \Sigma_L)) * I_L(\mathbf{u}) \\
&= (-\Sigma_L^{-1}\mathbf{u})L(\mathbf{u}; \Sigma_L) \\
&= (-\Sigma_L^{-1}\mathbf{u})R(A\mathbf{u}; A\Sigma_L A^T) \\
&= ((-\Sigma_L^{-1}\mathbf{u})\mathcal{G}(A\mathbf{u}, A\Sigma_L A^T)) * I_R(A\mathbf{u}) \\
&= ((-A^T A^{-T}\Sigma_L^{-1}\mathbf{u})\mathcal{G}(A\mathbf{u}, A\Sigma_L A^T)) * I_R(A\mathbf{u}) \\
&= (-A^T(A\Sigma_L A^T)^{-1}A\mathbf{u})\mathcal{G}(A\mathbf{u}, A\Sigma_L A^T)) * I_R(A\mathbf{u}) \\
&= (A^T \nabla \mathcal{G}(A\mathbf{u}, A\Sigma_{L,d}A^T)) * I_R(A\mathbf{u}) \\
&= A^T \nabla R(A\mathbf{u}; A\Sigma_L A^T)
\end{aligned}$$

as required.

The next step of the proof is to expand the outer convolution of eq. B.21:

$$\int_{\mathbf{u}} \mathcal{G}(\mathbf{u}_0 - \mathbf{u}; \Sigma_{L,\mathrm{i}}) A^T ((\nabla R(A\mathbf{u}; A\Sigma_{L,\mathrm{d}} A^T))(\nabla R(A\mathbf{u}; A\Sigma_{L,\mathrm{d}} A^T))^T) A \mathrm{d}\mathbf{u} \tag{B.26}$$

which, following similar steps to the proof of theorem B.1, can be rewritten as

$$A^T \int_{\mathbf{v}} \mathcal{G}(\mathbf{v}_0 - \mathbf{v}; A\Sigma_{L,\mathrm{i}} A^T)((\nabla R(\mathbf{v}; A\Sigma_{L,\mathrm{d}} A^T))(\nabla R(\mathbf{v}; A\Sigma_{L,\mathrm{d}} A^T))^T) \mathrm{d}\mathbf{v} A \tag{B.27}$$

□

B.3.2 Invariance Property of Fixed Points

Let us now assume that we can adapt the shape of the scale matrices such that at point $\mathbf{u}_0$ in image I_L we can find a *fixed point* M_L of the second moment matrix:

$$\mu_L(\mathbf{u}; \Sigma_{L,\mathrm{i}}, \Sigma_{L,\mathrm{d}}) = M_L \tag{B.28}$$

where the scale matrices are scalar multiples of M_L^{-1}:

$$\Sigma_{L,\mathrm{i}} = s_{\mathrm{i}} M_L^{-1} \tag{B.29}$$

$$\Sigma_{L,\mathrm{d}} = s_{\mathrm{d}} M_L^{-1} \tag{B.30}$$

Theorem B.3 If we know as before that the local neighbourhood of $\mathbf{u}_0$ in image I_L is related to the local neighbourhood of $\mathbf{v}_0$ in image I_R by a linear transformation A:

$$I_L(\mathbf{u}) = I_R(A\mathbf{u}) = I_R(\mathbf{v}) \tag{B.31}$$

then there must exist a corresponding fixed point at $\mathbf{v}_0$ in image I_R:

$$\mu_R(\mathbf{v}; \Sigma_{R,\mathrm{i}}, \Sigma_{R,\mathrm{d}}) = M_R \tag{B.32}$$

$$\Sigma_{R,\mathrm{i}} = s_{\mathrm{i}} M_R^{-1} \tag{B.33}$$

$$\Sigma_{R,\mathrm{d}} = s_{\mathrm{d}} M_R^{-1} \tag{B.34}$$

Proof *Using the transformation property (eq. B.23) we get*

$$\mu_L(\mathbf{u}; \Sigma_{L,\mathrm{i}}, \Sigma_{L,\mathrm{d}}) = A^T \mu_R(\mathbf{v}; A\Sigma_{L,\mathrm{i}}A^T, A\Sigma_{L,\mathrm{d}}A^T)A \tag{B.35}$$

and so

$$M_R = \mu_R(\mathbf{v}; \Sigma_{R,\mathrm{i}}, \Sigma_{R,\mathrm{d}}) = A^{-T} M_L A^{-1} \tag{B.36}$$

where

$$\Sigma_{R,\mathrm{i}} = A\Sigma_{L,\mathrm{i}}A^T = s_\mathrm{i}(AM_L^{-1}A^T) = s_\mathrm{i}(A^{-T}M_L A^{-1})^{-1} = s_\mathrm{i}M_R^{-1} \tag{B.37}$$

$$\Sigma_{R,\mathrm{d}} = A\Sigma_{L,\mathrm{d}}A^T = s_\mathrm{d}(AM_L^{-1}A^T) = s_\mathrm{d}(A^{-T}M_L A^{-1})^{-1} = s_\mathrm{d}M_R^{-1} \tag{B.38}$$

which specifies the fixed point as required.

□

This shows that under arbitrary invertible affine transformations, these fixed point conditions are preserved. This allows us to find a transformation from each image to a frame such that the intensities within this frame can be matched up to rotation, as follows.

We define the transformed images by

$$I_{L'}(\mathbf{u}') = I_{L'}(M_L^{\frac{1}{2}}\mathbf{u}) = I_L(\mathbf{u}) \tag{B.39}$$

$$I_{R'}(\mathbf{v}') = I_{R'}(M_R^{\frac{1}{2}}\mathbf{u}) = I_R(\mathbf{u}) \tag{B.40}$$

where the square root is defined such that $M^{\frac{1}{2}}M^{\frac{T}{2}} = M$.[1] Again, using the transformation property (eq. B.23) we can write

$$\begin{aligned}\mu_L(\mathbf{u}; s_\mathrm{i}M_L^{-1}, s_\mathrm{d}M_L^{-1}) &= M_L^{\frac{T}{2}}\mu_{L'}(M_L^{\frac{1}{2}}\mathbf{u}; M_L^{\frac{1}{2}}s_\mathrm{i}M_L^{-1}M_L^{\frac{T}{2}}, M_L^{\frac{1}{2}}s_\mathrm{d}M_L^{-1}M_L^{\frac{T}{2}})M_L^{\frac{1}{2}} \\ &= M_L^{\frac{T}{2}}\mu_{L'}(\mathbf{u}'; s_\mathrm{i}I, s_\mathrm{d}I)M_L^{\frac{1}{2}}\end{aligned}$$

Substituting (eq. B.28) and rearranging gives

$$\mu_{L'}(\mathbf{u}'; s_\mathrm{i}I, s_\mathrm{d}I) = I \tag{B.41}$$

[1] Note this is different from [3]: their paper seems to be mistaken or at least not clearly specified given the proofs in this document.

$$\begin{array}{ccc} I_L(\mathbf{u}) & \xrightarrow{\mathbf{v}=A\mathbf{u}} & I_R(\mathbf{v}) \\ {\scriptstyle \mathbf{u}'=M_L^{\frac{1}{2}}\mathbf{u}}\downarrow & & \downarrow{\scriptstyle \mathbf{v}'=M_R^{\frac{1}{2}}\mathbf{v}} \\ I_{L'}(\mathbf{u}') & \xrightarrow{\mathbf{v}'=A_R\mathbf{u}'} & I_{R'}(\mathbf{v}') \end{array}$$

Figure B.1: Diagram illustrating commutativity of transformations.

A similar derivation for $\mu_{R'}$ shows that this too equals the identity. So we have found transformations for both images such that the second moment matrix becomes the identity. We can therefore write (from eq. B.36):

$$M_L = A^T M_R A \tag{B.42}$$

which is verified for

$$A = M_R^{-\frac{1}{2}} A_R M_L^{\frac{1}{2}} \tag{B.43}$$

for some arbitrary rotation, A_R. This implies that the affine transformation between the original images can be decomposed into a transformation into a isotropic frame, a rotation, and a transformation out of the isotropic frame. It is now easy to show that in the isotropic frame the two images are equal up to this rotation:

$$\mathbf{v} = A\mathbf{u} \tag{B.44}$$

$$\mathbf{u}' = M_L^{\frac{1}{2}}\mathbf{u} \tag{B.45}$$

$$\mathbf{v}' = M_R^{\frac{1}{2}}\mathbf{v} \tag{B.46}$$

$$\mathbf{v}' = M_R^{\frac{1}{2}} A M_L^{-\frac{1}{2}}\mathbf{u}' \tag{B.47}$$

$$\mathbf{v}' = A_R\mathbf{u}' \tag{B.48}$$

as illustrated in fig. B.1. By definition therefore, since the transformation commutes with the imaging process, the fixed points in the two images are affine invariants.

This scheme of normalisation to an isotropic frame offers a useful technique: *if* we can find fixed points in the two images we will be able to compare intensity patterns or compute descriptors of features in the normalised frame such that they match up to rotation (which can be compensated for easily) in a fashion that is invariant to affine geometric changes. Also, at matched features, we will get an estimate of the local affine transformation which has been used cleverly in e.g. [61, 59].

B.4 Investigation of Convergence

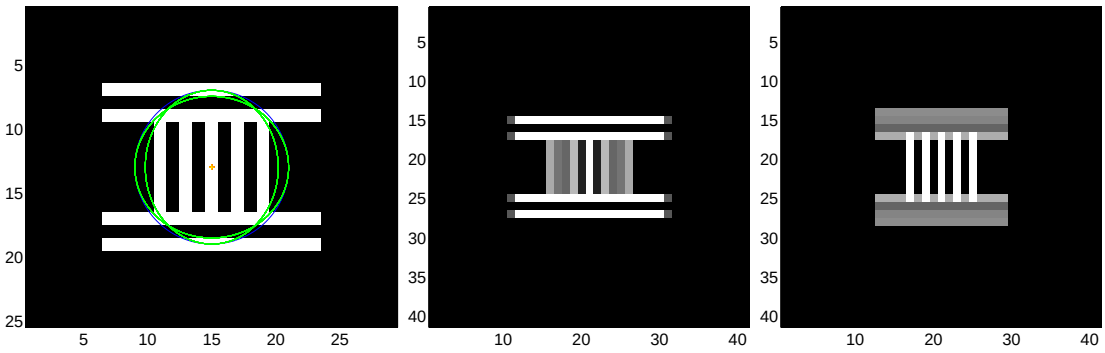
Blake & Marinos present in [6] an analysis of the convergence of an iterative adaptive algorithm based on the second moment matrix of gradient orientations for obtaining shape information from texture. It would be good to be able to prove a similar convergence for the Harris-Affine interest point detector algorithm of [53], described in §2.1.6, but this is not possible as counter-examples below will demonstrate. The reason for this is that by adapting the shape of the local neighbourhood, the algorithm is also changing the area of the image that is being considered since only a windowed region of the transformed coordinate frame is used. This can lead to the situation where the neighbourhood shape oscillates and never converges.

We present two counter-examples to convergence. These both assume that the scale of integration and of differentiation are held fixed. One example works if the spatial location is allowed to vary. The iteration seed point location and scale are chosen by hand in each case. We have not tried to find an example where the seed point is located with the Harris-Laplace detector ([52]) since we are simply trying to illustrate the principle of non-convergence on artificial images. Whether these situations arise in real images and when the scales are allowed to adapt is another matter.

The principle in both examples is to find an image and a feature such that the algorithm ends up oscillating between two neighbourhood shapes and never converges. This will happen where from one iteration to the next, the extra region of image introduced by the shape-adaptation has a high gradient in the tangential direction to the previous adaptation.

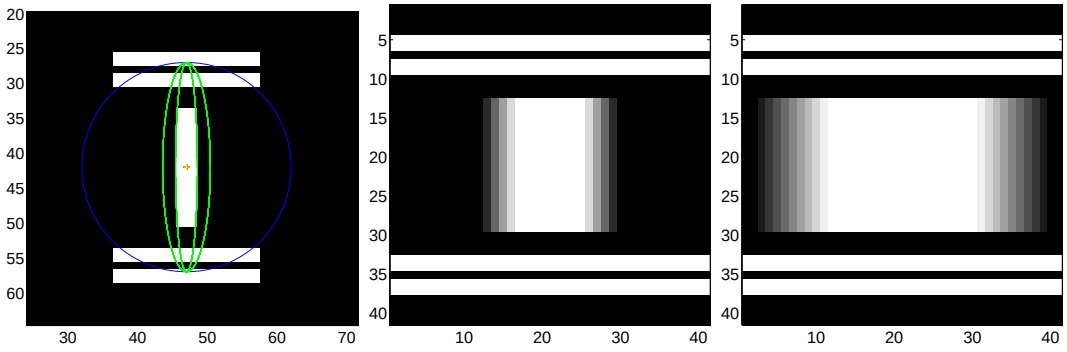
The first example is shown in fig. B.2. The test image is shown in (a) with the initial seed window overlaid in blue. After a certain time T the algorithm reaches an oscillating state illustrated in (b) and (c) such that $I_N^{[\tau+2]} = I_N^{[\tau]}$. The green ellipses in (a) show these two states. The second example is shown similarly in fig. B.3. In this case, if the localisation of the point is allowed to change as described above then the iteration still does not converge, though instead of oscillating between two states it cycles between many more since there are more parameters changing at each iteration.

The question remains whether the algorithm could be altered to guarantee convergence. Presumably the only way to prevent the existence of counter-examples such as these is if the algorithm uses all available information at each step (i.e. the whole image in some form), but this would ruin its local applicability which is the source of its power. It seems that in



(a) (b) (c)

Figure B.2: Counter-example image 1. (a) Image with initial (blue) and oscillating (green) neighbourhoods. (b) Normalised window $I_N^{[\tau]}$. (c) Normalised window $I_N^{[\tau+1]}$.



(a) (b) (c)

Figure B.3: Counter-example image 2. (a) Image with initial (blue) and oscillating (green) neighbourhoods. (b) Normalised window $I_N^{[\tau]}$. (c) Normalised window $I_N^{[\tau+1]}$.

practice though, having no guarantee of convergence does not pose a large problem.

APPENDIX C

SSD AND NCC

To generate matching scores between patches, we have investigated using both sum-of-squared differences (SSD) and normalised cross correlation (NCC). In this appendix we give the definitions for both of these, their relationship, and how to calculate them for midground patches where the foreground region should be matched independently of the background region.

We will deal with two patches $\mathbf{x}$ and $\mathbf{y}$ which are treated as vectors of dimension N with elements x_i and y_i respectively. Note that for *colour* patches, we simply concatenate the colour dimensions into one long vector. We also pre-process appearance patches by dividing each pixel by 255 (the brightest intensity value) to ensure the range of values for SSD falls between zero and one; note that this has no effect for NCC as it is normalised for variance.

C.1 SSD

The straightforward definition for sum-of-squared differences is as follows:

$$\text{SSD}'(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} \|x_i - y_i\|^2 \tag{C.1}$$

We choose to normalise this definition by N the dimension of the vector, so that changing the size of patches being compared does not scale the SSD score. Consequently from now on, SSD will be defined as:

$$\text{SSD}(\mathbf{x}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} \|x_i - y_i\|^2 \tag{C.2}$$

This measure has a minimum score of 0 representing a perfect match, and a maximum of 1 representing a black patch being matched against a white patch.

We can rewrite this equation by considering this to be an *expectation* of the squared difference with regards to a uniform distribution $\mathbf{u} = \left(\frac{1}{N} \dots \frac{1}{N}\right)^T$:

$$\begin{aligned} \text{SSD}_{\mathbf{u}}(\mathbf{x}, \mathbf{y}) &= \sum_{i=1}^{N} u_i \|x_i - y_i\|^2 && \text{(C.3)} \\ &= \mathcal{E}_{\mathbf{u}}(\|x - y\|^2) && \text{(C.4)} \end{aligned}$$

where the expectation $\mathcal{E}_{\mathbf{u}}(a) = \sum_{i=1}^{N} u_i a_i$.

But we can replace the uniform distribution by an arbitrary distribution $\mathbf{p}$ to represent a shape patch $\mathbf{s}$ (i.e. a mask patch where 1 represents foreground and 0 represents background) that has been normalised:

$$p_i = \frac{s_i}{\sum_{i=1}^{N} s_i} \tag{C.5}$$

Using this we get the effective normalised SSD score for a shape patch:

$$\text{SSD}_{\mathbf{p}}(\mathbf{x}, \mathbf{y}) = \mathcal{E}_{\mathbf{p}}(\|x - y\|^2) \tag{C.6}$$

This weighted SSD score will have a value as if the shape was completely foreground and the score had been taken over the whole patch, and so can be used comparably with non-weighted SSD scores.

Efficient computation

Note that for efficient computation, the definitions can be expanded as:

$$\begin{aligned} \text{SSD}_{\mathbf{p}}(\mathbf{x}, \mathbf{y}) &= \mathcal{E}_{\mathbf{p}}(\|x - y\|^2) && \text{(C.7)} \\ &= \mathcal{E}_{\mathbf{p}}(x^2) + \mathcal{E}_{\mathbf{p}}(y^2) - 2\mathcal{E}_{\mathbf{p}}(xy) && \text{(C.8)} \end{aligned}$$

The $\mathcal{E}_{\mathbf{p}}(x^2)$ and $\mathcal{E}_{\mathbf{p}}(y^2)$ could be pre-computed for each patch, but this is only sensible if the distribution $\mathbf{p}$ is fixed and known.

C.2 NCC

The normalised cross correlation gives a matching score between -1 and $+1$, by first centering (subtracting the mean) and whitening (dividing by the variance) the data before per-

forming a correlation. The standard notation for this is

$$\text{NCC}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}} \tag{C.9}$$

where $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ is the mean of $\mathbf{x}$ and similarly $\bar{y}$ is the mean of $\mathbf{y}$. Note that this score is *normalised* to the range -1 to $+1$. The value $+1$ means perfect correlation, -1 means perfect anti-correlation, and 0 means zero correlation.

As before, we can rewrite this in terms of expectations with respect to a uniform distribution $\mathbf{u}$. First note that the two terms on the bottom are simply the variances of $\mathbf{x}$ and $\mathbf{y}$ scaled up:

$$\sum_{i=1}^{N}(x_i - \bar{x})^2 = N\mathcal{V}_{\mathbf{u}}(x) \tag{C.10}$$

where the variance $\mathcal{V}_{\mathbf{u}}(x) = \mathcal{E}_{\mathbf{u}}(x^2) - \mathcal{E}_{\mathbf{u}}(x)^2$. We can rewrite the NCC therefore in terms of these variances and expand as follows:

$$\begin{aligned}
\text{NCC}_{\mathbf{u}}(\mathbf{x}, \mathbf{y}) &= \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{N\sqrt{\mathcal{V}_{\mathbf{u}}(x)\mathcal{V}_{\mathbf{u}}(y)}} && \text{(C.11)}\\
&= \frac{\mathcal{E}_{\mathbf{u}}((x - \mathcal{E}_{\mathbf{u}}(x))(y - \mathcal{E}_{\mathbf{u}}(y)))}{\mathcal{V}_{\mathbf{u}}(x)^{\frac{1}{2}}\mathcal{V}_{\mathbf{u}}(y)^{\frac{1}{2}}} && \text{(C.12)}\\
&= \frac{\mathcal{E}_{\mathbf{u}}(xy) - \mathcal{E}_{\mathbf{u}}(x)\mathcal{E}_{\mathbf{u}}(y)}{\mathcal{V}_{\mathbf{u}}(x)^{\frac{1}{2}}\mathcal{V}_{\mathbf{u}}(y)^{\frac{1}{2}}} && \text{(C.13)}
\end{aligned}$$

It is trivial now to replace the uniform distribution $\mathbf{u}$ with the distribution $\mathbf{p}$ representing the shape patch to get the normalised NCC score:

$$\text{NCC}_{\mathbf{p}}(\mathbf{x}, \mathbf{y}) = \frac{\mathcal{E}_{\mathbf{p}}(xy) - \mathcal{E}_{\mathbf{p}}(x)\mathcal{E}_{\mathbf{p}}(y)}{\mathcal{V}_{\mathbf{p}}(x)^{\frac{1}{2}}\mathcal{V}_{\mathbf{p}}(y)^{\frac{1}{2}}} \tag{C.14}$$

As with the weighted SSD, this weighted NCC score will have a value as if the score had been taken over a whole patch, and so can be used comparably with non-weighted NCC scores.

Efficient computation

The above weighted NCC score can be computed efficiently by decomposing the variances into expectations. Then only one pass through the data is necessary, accumulating $\mathcal{E}_{\mathbf{p}}(x)$, $\mathcal{E}_{\mathbf{p}}(x^2)$, $\mathcal{E}_{\mathbf{p}}(y)$, $\mathcal{E}_{\mathbf{p}}(y^2)$, and $\mathcal{E}_{\mathbf{p}}(xy)$.

C.3 Relationship Between SSD and NCC

If we take our original data $\mathbf{x}$ and $\mathbf{y}$ and centre and whiten it, we get new data:

$$x'_i = \frac{x_i - \mathcal{E}_{\mathbf{p}}(x)}{\mathcal{V}_{\mathbf{p}}(x)^{\frac{1}{2}}} \tag{C.15}$$

$$y'_i = \frac{y_i - \mathcal{E}_{\mathbf{p}}(y)}{\mathcal{V}_{\mathbf{p}}(y)^{\frac{1}{2}}} \tag{C.16}$$

Since $\mathcal{E}_{\mathbf{p}}(x') = 0$, $\mathcal{V}_{\mathbf{p}}(x') = 1$, and similarly for $\mathbf{y}'$, we now have:

$$\text{NCC}_{\mathbf{p}}(\mathbf{x}', \mathbf{y}') = \mathcal{E}_{\mathbf{p}}(xy) \tag{C.17}$$

$$= -\frac{1}{2}\text{SSD}_{\mathbf{p}}(\mathbf{x}', \mathbf{y}') + \frac{1}{2}\left[\mathcal{E}_{\mathbf{p}}(x'^2) + \mathcal{E}_{\mathbf{p}}(y'^2)\right] \tag{C.18}$$

C.4 Mixing Scores for Midground Patches

When generating midground hypotheses, the maximum likelihood descriptions of foreground and background regions are found independently based on the shape hypothesis being considered. We denote the shape patch as $\mathbf{s} = (s_1, \ldots, s_N)^T$, for which the fraction of foreground area is effectively the mean value $\bar{s} = \frac{1}{N}\sum_{i=1}^{N} s_i$, and the fraction of background area is simply $1 - \bar{s}$.

Given the maximum likelihood scores $v^{\diamondsuit}$ for the foreground region and $v^{\blacklozenge}$ for the background region, the overall score is simply the weighted sum:

$$v = \bar{s}v^{\diamondsuit} + (1 - \bar{s})v^{\blacklozenge} \tag{C.19}$$

APPENDIX D

MISCELLANEOUS

D.1 Detection Probabilities and Mutual Information

Borenstein & Ullman do not present the full mathematical detail needed to implement their papers ([9, 10]). For completeness therefore, we fill in the details in this section.

Using the definitions from §3.2, we can count the number of detections of fragment g_i in image sets $\mathcal{C}$ and $\mathcal{B}$:

$$N_{\mathcal{C}}(g_i) = \sum_{j=1}^{M_{\mathcal{C}}} \zeta(g_i, c_j) \tag{D.1}$$

$$N_{\mathcal{B}}(g_i) = \sum_{j=1}^{M_{\mathcal{B}}} \zeta(g_i, b_j) \tag{D.2}$$

Now, consider the random variable $X_{g_i} \in \{1, 0\}$ representing the detection or non-detection of fragment g_i, and the random variable $Y \in \{\mathcal{C}, \mathcal{B}\}$ representing the choice of image set, class or non-class. Below we simplify the notation $p(Y = \mathcal{C})$ to $p(\mathcal{C})$.

It is simple to estimate the class-conditional probabilities for the detection of fragment g_i by looking at detection frequencies:

$$p(X_{g_i} = 1|\mathcal{C}) = \frac{N_{\mathcal{C}}(g_i) + 1}{M_{\mathcal{C}} + 1} \tag{D.3}$$

$$p(X_{g_i} = 1|\mathcal{B}) = \frac{N_{\mathcal{B}}(g_i) + 1}{M_{\mathcal{B}} + 1} \tag{D.4}$$

where the ones have been added to ensure non-zero estimates. We can estimate the class probabilities as:

$$p(\mathcal{C}) = \frac{M_{\mathcal{C}}}{M_{\mathcal{C}} + M_{\mathcal{B}}} \tag{D.5}$$

$$p(\mathcal{B}) = \frac{M_{\mathcal{B}}}{M_{\mathcal{C}} + M_{\mathcal{B}}} \tag{D.6}$$

where clearly $p(\mathcal{B}) = 1 - p(\mathcal{C})$. From these we can derive detection probabilities:

$$p(X_{g_i} = 1) = \sum_{Y \in \{\mathcal{C},\mathcal{B}\}} p(X_{g_i} = 1|Y)p(Y) \tag{D.7}$$

$$p(X_{g_i} = 0) = \sum_{Y \in \{\mathcal{C},\mathcal{B}\}} p(X_{g_i} = 0|Y)p(Y) \tag{D.8}$$

We can write the likelihood ratio of the fragment being detected in class images versus non-class images as

$$L_{g_i} = \frac{p(X_{g_i} = 1|\mathcal{C})}{p(X_{g_i} = 1|\mathcal{B})} \tag{D.9}$$

and the mutual information between the variables X_{g_i} and Y:

$$I(X_{g_i}; Y) = H(X_{g_i}) - H(X_{g_i}|Y) \tag{D.10}$$

where the entropies can be calculated as

$$H(X_{g_i}|Y) = \sum_{Y \in \{\mathcal{C},\mathcal{B}\}} p(Y)H(X_{g_i}|Y) \tag{D.11}$$

$$H(X_{g_i}) = -\sum_{X_{g_i} \in \{0,1\}} p(X_{g_i}) \log_2 p(X_{g_i}) \tag{D.12}$$

$$H(X_{g_i}|Y = \mathcal{C}) = -\sum_{X_{g_i} \in \{0,1\}} p(X_{g_i}|\mathcal{C}) \log_2 p(X_{g_i}|\mathcal{C}) \tag{D.13}$$

$$H(X_{g_i}|Y = \mathcal{B}) = -\sum_{X_{g_i} \in \{0,1\}} p(X_{g_i}|\mathcal{B}) \log_2 p(X_{g_i}|\mathcal{B}) \tag{D.14}$$

$$\tag{D.15}$$

BIBLIOGRAPHY

[1] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *ECCV02*, page IV: 113 ff., 2002.

[2] R.C. Agrawal, R.K. Shevgaonkar, and S.C. Sahasrabudhe. A fresh look at the hough transform. *PRL*, 17(10):1065–1068, September 1996.

[3] A. Baumberg. Reliable feature matching across widely separated views. In *Computer Vision and Pattern Recognition, CVPR*, pages 774–781, 2000.

[4] S. Beucher. Watersheds of functions and picture segmentation. In *ICASSP82*, pages 1928–1931, 1982.

[5] C.M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1996.

[6] A. Blake and C. Marinos. Shape from texture: Estimation, isotropy and moments. *AI*, 45(3):323–380, October 1990.

[7] A. Blake, C. Rother, M. Brown, P. Perez, and P.H.S. Torr. Interactive image segmentation using an adaptive GMMRF model. In *ECCV04*, pages Vol I: 428–441, 2004.

[8] E. Borenstein, E. Sharon, and S. Ullman. Combining top-down and bottom-up segmentations. In *POCV04*, 2004.

[9] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *ECCV02*, page II: 109 ff., 2002.

[10] E. Borenstein and S. Ullman. Learning to segment. In *ECCV04*, 2004.

[11] Y.Y. Boykov and M.P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *ICCV01*, pages I: 105–112, 2001.

[12] Y.Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *ICCV (1)*, pages 377–384, 1999.

[13] M.C. Burl and P. Perona. Recognition of planar object classes. In *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, page 223. IEEE Computer Society, 1996.

[14] M.C. Burl, M. Weber, and P. Perona. A probabilistic approach to object recognition using local photometry and global geometry. In *Proceedings of the 5th European Conference on Computer Vision-Volume II*, pages 628–641. Springer-Verlag, 1998.

[15] J. Canny. A computational approach to edge detection. *PAMI*, 8(6):679–698, November 1986.

[16] Y.Y. Chuang, B. Curless, D.H. Salesin, and R. Szeliski. A bayesian approach to digital matting. In *CVPR01*, pages II:264–271, 2001.

[17] A. Criminisi, J. Shotton, A. Blake, and P.H.S. Torr. Gaze manipulation for one-to-one teleconferencing. In *ICCV03*, pages 191–198, 2003.

[18] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. In *JRSS B*, pages 39:1–38, 1976.

[19] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

[20] L. Fei-Fei, R. Fergus, and P. Perona. A Bayesian approach to unsupervised one-shot learning of object categories. In *Proceedings of the 9th International Conference on Computer Vision, Nice, France*, pages 1134–1141, October 2003.

[21] P.F. Felzenszwalb and D.P. Huttenlocher. Object recognition with pictorial structures. In *MIT AI-TR*, 2001.

[22] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient belief propagation for early vision. In *Proceedings of IEEE CVPR*, 2004.

[23] 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*, June 2003.

[24] R. Fergus, P. Perona, and A. Zisserman. A visual category filter for Google images. In *Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic*. Springer, May 2004.

[25] V. Ferrari, T. Tuytelaars, and L.J. Van Gool. Simultaneous object recognition and segmentation by image exploration. In *ECCV04*, pages Vol I: 40–54, 2004.

[26] A.W. Fitzgibbon and A. Zisserman. On affine invariant clustering and automatic cast listing in movies. In *ECCV02*, page III: 304 ff., 2002.

[27] W. Förstner. A framework for low level feature extraction. In *Proceedings of the third European conference on Computer Vision (Vol. II)*, pages 383–394. Springer-Verlag New York, Inc., 1994.

[28] W.T. Freeman. *Steerable Filters and Analysis of Image Structure*. PhD thesis, MIT, 1992.

[29] W.T. Freeman, E.C. Pasztor, and O.T. Carmichael. Learning low-level vision. *IJCV*, 40(1):25–47, October 2000.

[30] B.J. Frey and N. Jojic. Transformation-invariant clustering using the em algorithm. *PAMI*, 25(1):1–17, January 2003.

[31] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *PAMI*, 6(6):721–741, November 1984.

[32] R.L. Gregory. *Eye and Brain: The Psychology of Seeing*. Oxford University Press, 1997.

[33] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th ALVEY vision conference*, pages 147–151, 1988.

[34] E.T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[35] N. Jojic, B.J. Frey, and A. Kannan. Epitomic analysis of appearance and shape. In *ICCV03*, pages 34–41, 2003.

[36] M.I. Jordan, Z. Ghahramani, T. Jaakkola, and L.K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[37] T. Kadir, D. Boukerroui, and M. Brady. An analysis of the scale saliency algorithm. Technical report, University of Oxford, 2003.

[38] T. Kadir and M. Brady. Saliency, scale and image description. *IJCV*, 45(2):83–105, November 2001.

[39] T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. In *ECCV04*, pages Vol I: 228–241, 2004.

[40] I. Laptev and T. Lindeberg. Space-time interest points. In *ICCV03*, pages 432–439, 2003.

[41] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *ECCV04*, 2004.

[42] B. Leibe and B. Schiele. Interleaved object categorization and segmentation. In *BMVC03*, 2003.

[43] B. Leibe and B. Schiele. Scale invariant object categorization using a scale-adaptive mean-shift search. In *DAGM04*, 2004.

[44] T. Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer, December 1993.

[45] 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 *Proc 3rd European Conference on Computer Vision, ECCV 1994*, pages 389–400, 1994.

[46] T. Lindeberg and J. Gårding. Shape-adapted smoothing in estimation of 3-d shape cues from affine deformations of local 2-d brightness structure. *Image and Vision Computing, 15(6):415–434, June 1997*, 1997.

[47] D.G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the International Conference on Computer Vision ICCV, Corfu*, pages 1150–1157, 1999.

[48] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, November 2004.

[49] D.J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[50] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *IJCV*, 43(1):7–27, June 2001.

[51] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *BMVC02*, page 3D and Video, 2002.

[52] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *ICCV*, pages 525–531, 2001.

[53] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *ECCV*, pages 128–142, 2002.

[54] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *CVPR03*, pages II: 257–263, 2003.

[55] K. Mikolajczyk, A. Zisserman, and C. Schmid. Shape recognition with edge-based features. In *Proceedings of the British Machine Vision Conference*, 2003.

[56] K.P. Murphy, Y. Weiss, and M.I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *UAI*, pages 467–475, 1999.

[57] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. San Mateo: Morgan Kaufmann, 1988.

[58] C. Rother, V. Kolmogorov, and A. Blake. Grabcut - interactive foreground extraction using iterated graph cuts. *Proc. ACM Siggraph*, 2004.

[59] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3d object modeling and recognition using affine-invariant patches and multi-view spatial constraints. *CVPR03*, pages II: 272–277, 2003.

[60] M.A. Ruzon and C. Tomasi. Alpha estimation in natural images. In *CVPR00*, pages I: 18–25, 2000.

[61] F. Schaffalitzky and A. Zisserman. Automated scene matching in movies, 2002.

[62] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or 'how do i organize my holiday snaps?'. In *ECCV02*, page I: 414 ff., 2002.

[63] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1-3):7–42, April 2002.

[64] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *PAMI*, 19(5):530–535, May 1997.

[65] H. Schneiderman and T. Kanade. Object detection using the statistics of parts. *IJCV*, 56(3):151–177, February 2004.

[66] J. Shi and J. Malik. Normalized cuts and image segmentation. In *CVPR97*, pages 731–737, 1997.

[67] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV03*, pages 1470–1477, 2003.

[68] A. Thayananthan, R. Navaratnam, P. H. S. Torr, and R. Cipolla. Likelihood models for template matching using pdf projection theorem. In *Proc. British Machine Vision Conference*, Lonon, UK, September 2004.

[69] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla. Learning a kinematic prior for tree-based filtering. In *Proc. British Machine Vision Conference*, volume 2, pages 589–598, Norwich, UK, September 2003.

[70] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume I, pages 127–133, Madison, USA, June 2003.

[71] A. Torralba, K.P. Murphy, W.T. Freeman, and M.A. Rubin. Context-based vision system for place and object recognition. In *ICCV03*, pages 273–280, 2003.

[72] K. Toyama and A. Blake. Probabilistic tracking with exemplars in a metric space. *IJCV*, 48(1):9–19, June 2002.

[73] Z. Tu, X. Chen, A.L. Yuille, and S.C. Zhu. Image parsing: unifying segmentation, detection, and recognition. In *ICCV03*, pages 18–25, 2003.

[74] T. Tuytelaars and L.J. Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *BMVC00*, 2000.

[75] S. Ullman, E. Sali, and M. Vidal-Naquet. A fragment-based approach to object representation and classification. In *VF01*, page 85 ff., 2001.

[76] L.J. Van Gool, T. Moons, and D. Ungureanu. Affine/photometric invariants for planar intensity patterns. In *ECCV96*, pages I:642–651, 1996.

[77] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *ECCV (1)*, pages 18–32, 2000.

[78] Y. Weiss and W.T. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, 2001.

[79] J.S. Yedidia, W.T Freeman, and Y. Weiss. Generalized belief propagation. In *Technical Report 2000-26, MERL, Mitsubishi Electric Research Labs.*, 2000.

[80] J.S. Yedidia, W.T. Freeman, and Y. Weiss. *Understanding belief propagation and its generalizations*. Morgan Kaufmann Publishers Inc., 2003.