Sprite Learning and Object Category Recognition using Invariant Features

Moray Allan

Doctor of Philosophy

Institute for Adaptive and Neural Computation

School of Informatics

University of Edinburgh

2007

Abstract

This thesis explores the use of invariant features for learning sprites from image sequences, and for recognising object categories in images.

A popular framework for the interpretation of image sequences is the layers or sprite model of e.g. Wang and Adelson (1994), Irani et al. (1994). Jojic and Frey (2001) provide a generative probabilistic model framework for this task, but their algorithm is slow as it needs to search over discretised transformations (e.g. translations, or affines) for each layer. We show that by using invariant features (e.g. Lowe's SIFT features) and clustering their motions we can reduce or eliminate the search and thus learn the sprites much faster. The algorithm is demonstrated on example image sequences.

We introduce the Generative Template of Features (GTF), a parts-based model for visual object category detection. The GTF consists of a number of parts, and for each part there is a corresponding spatial location distribution and a distribution over 'visual words' (clusters of invariant features). We evaluate the performance of the GTF model for object localisation as compared to other techniques, and show that such a relatively simple model can give state-of-the-art performance. We also discuss the connection of the GTF to Hough-transform-like methods for object localisation.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified. Some of the material in chapters 3 and 4 is adapted from Allan et al. (2005); section 3.5 describes the component of the sprite learning system contributed by Michalis Titsias. Some of the material in chapter 5 is adapted from Williams and Allan (2006).

(*Moray Allan*)

Table of Contents

Chapter 1

Introduction

One of the key problems in computer vision is object recognition: can we build a computer system which is able to identify the objects present in images? In this thesis we address the task of recognition of object categories (such as cars), using a generative model for invariant features extracted from images. We also address the task of sprite extraction, using similar invariant features extracted from image sequences to separate out the objects present in a scene based on their motion, then creating a ‘cardboard cut-out’ sprite model of each specific object’s appearance.

Objects can be dealt with either as specific objects, such as ‘the car you saw last Thursday’, or as object categories, such as ‘cars in general’. In some applications we want to detect exactly the same object we have seen before, such as the face of particular person who appeared in our training data (see for example Jones and Viola, 2003). In other applications we want to detect objects based on the class of objects they belong to, such as handwritten digits that can take different forms while still meaning the same thing (see for example Le Cun et al., 1990). For sprite extraction we will deal with specific objects present in a single scene, while for object recognition we will deal with categories of object. The appearance of individual objects varies as they are seen in different lighting conditions, positioned in different ways, and viewed from different angles. To learn an object’s motion through an image sequence we need to cope with this change in appearance, and also with occlusion caused by other objects coming between the object and the camera. There is further variation between members of the same category of objects: different examples of the same object category may have very different shapes, textures, and colours. To recognise object categories we need to deal with both types of appearance variation.

Specific objects or object categories can be modelled at a variety of levels. We can take an image file and work directly with the pixel information it contains (as in Le Cun et al., 1990; Jones and Viola, 2003), treating the image as an array of colour or intensity observations. Alternatively, instead of working directly with pixel data, we can preprocess the data and work with features extracted from the images (see for example Weber et al., 2000b; Fergus et al., 2003). Useful image features typically transform raw pixel data into a more efficient, higher-level representation. In the experiments below, region detectors first select image regions which are likely to be informative, then the SIFT descriptor (Lowe, 2004) is used to produce a compact 128-dimensional representation of the distribution of orientations in each local region. If the features we select are not appropriate to the task we are performing, we risk throwing away important information that was present at the pixel level, but for many vision tasks using features gives a huge reduction in the amount of computational work needed to train models. This can make it practical to approach tasks which would be impractical if we worked directly with raw pixels, as they would require more computational time than is available. In our sprite extraction system we use feature-level information to speed up object motion learning, before learning the sprites themselves at the pixel level.

We approach both object recognition and sprite extraction using probabilistic modelling techniques. In both cases we use a generative model of the entire image including the background, rather than a discriminative model which only tries to distinguish the objects of interest. For object recognition we take labelled example visual scenes as training data for the object recognition system, then use the resulting models to detect in novel test data the object categories which were represented in the training data. For sprite extraction we fit a generative sprite model to image sequences, jointly optimising the models for each object and for the background to best fit the data.

This thesis examines how we can use invariant features of images to learn sprites for objects from image sequences, and how we can use invariant features to learn to recognise and localise categories of object. In both cases we use feature detectors to identify regions of interest in images, use region descriptors to create compact descriptions of those regions, match features between images, then learn a probabilistic model of the objects. We define the tasks of sprite learning and object localisation more closely below:

1.1 Fast learning of sprites

Given multiple views of a scene, such as the frames of a video sequence, the task is to learn the appearance of each of the independently moving objects. Each object can be thought of as a flat 'cardboard cut-out' with its own shape and surface texture, that has its own motion independent of the other objects.

To achieve this in an efficient manner, we first use invariant features to separate out individual objects and to learn an approximation for the motion of each one. We learn the objects in a scene by looking for consistent motion groupings, assuming that features belonging to a single object will move together, and that separate objects will have features that move independently. Once the objects have been learnt in this way, we use a probabilistic model which can reason about object occlusion to fill out the sparse feature models for the objects with pixel-level sprite and mask information. The objects in the scene are modelled as a number of ordered occluding layers, with a background layer behind them all.

The object motions learnt by the system would in themselves be useful in many applications. For example in a conference room situation we might want to model the movements of individual participants so that automatic cameras can keep them in view for remote participants. In other applications we also want to perform sprite extraction. For example for video editing we might need to extract the pixels corresponding to a particular object frame by frame, to paste it into another scene. More generally we might want, for example, to extract a single representative image of each object moving through a scene.

1.2 Object localisation

Given a set of annotated training images, the task is to learn models for object classes which will allow us to recognise objects of those classes in novel images, and to specify their locations in the images.

We model the object classes using the Generative Template of Features (GTF), a probabilistic model for objects in cluttered scenes. The GTF accounts for both features seen on the foreground object and features seen in the background.

Like the layered object model for learning object motions in the first task, the GTF only reasons about objects in terms of invariant features, not pixel-level sprite information. The GTF uses high-level descriptors of local appearance grouped together into 'visual words': just as the words present in a document let us make predictions about the topics it deals with, the visual words present in an image let us make predictions about the object classes which appear in it.

The data sets we use in the experiments mostly consist of street scenes, which might suggest applications in, for example, automated vehicle control, with a computer system reacting to other vehicles and pedestrians in its surroundings. However, there are far wider applications for object recognition methods. Even for a computer in an ordinary home environment, human–computer interaction could be significantly improved by the system understanding the objects in its environment: for example, recognising when there are people in the room, or what objects the people are holding.

1.3 Outline

Chapter 2 gives an overview of recent work on image features, including the difference-of-Gaussians features used in the experiments in chapter 4, and the maximally stable extrema regions and Harris affine features used in the experiments in chapter 6. We introduce methods for matching image features to detect instances of specific objects, and for clustering features to recognise object classes.

Chapter 3 describes a method for fast learning of sprites from image sequences, using invariant features. After a summary of related work, we describe how we can match features between the images in a sequence, and how we can learn object motions from the matched features. We then give a generative model for the pixel-level sprite information in layered image sequences and explain how this can be combined with the feature information previously learnt.

Chapter 4 describes some experimental results using the system developed in chapter 3.

We analyse the results of the feature extraction and feature-matching stages described above, then demonstrate what the system can achieve on some example video sequences.

Chapter 5 describes a method for learning and detecting object classes, using invariant features. After a summary of related work in object classification and localisation, we give a generative model for features in images with cluttered backgrounds. Finally we show how the Generative Template of Features model can be related to pose-clustering methods.

Chapter 6 describes some experimental results using the system developed in chapter 5, using the data from the PASCAL 2005 Visual Object Classes challenge (Everingham et al., 2005). After analysing the 'visual words' learnt by our clustering approach, we give some implementation details about the Generative Template of Features system used subsequently. A first set of experiments is carried out using the training and validation data, examining how the detection performance varies with the choice of parameters. A second set of experiments looks in more detail at the performance on the challenge test data.

Chapter 7 gives a summary of the thesis, and discusses some possible directions for future work.

Chapter 2

Feature detection and extraction

2.1 Introduction

The work described in this thesis uses invariant features to provide a higher-level representation of the objects in a scene than is given by raw image pixel data. This chapter looks at recent work on image features, including the feature types which we use in the experiments in chapters 4 and 6 below.

In many computer vision tasks it is useful to be able to generalise across images. For example, it is often useful for methods to be able to deal with large sets of transformations of the objects in a scene and of the camera, without having to learn separately about every possible arrangement of camera and objects. By providing a higher-level representation than is given by raw image pixel data, image features make it easier to develop methods capable of this kind of generalisation. Moreover, most images contain a great deal of information that is irrelevant to any individual computer vision task. Image features can be used to focus computation on areas of an image which are likely to be informative, while avoiding unnecessary computation on, for example, textureless regions. In this way, using image features rather than raw image data can allow faster inference about the objects in a scene.

In chapters 3 and 4 we use invariant features to speed up learning of object sprites from image sequences. Higher-level features allow more efficient matching between images than low-level pixel information. In chapters 5 and 6 we use invariant features to learn and recognise object classes in images. Higher-level features allow efficient learning across different object scales and poses.

Section 2.2 begins a discussion of image features; sections 2.3 and 2.4 give more detail on the particular region-based feature methods we use. Section 2.4.1 describes how we can perform feature matching to detect instances of specific objects, while section 2.4.2 introduces the feature-clustering approach that can be useful for matching more general object categories.

2.2 Image features

To match objects in different images, we can either use ‘direct’ pixel-based methods or ‘indirect’ feature-based methods. Whereas a direct method works from raw image data, with each image represented as a high-dimensional array of pixels, an indirect method may use only selected regions of the image, where the local neighbourhood of each region is represented by a low-dimensional descriptor. Indirect approaches can be thought of as feature selection on the raw image data.

The system described in chapter 3 uses this kind of feature extraction to identify regions whose motions can be reliably tracked through an image sequence, allowing sprite learning for moving objects to be carried out many times faster than the direct approach without the use of features. The object class learning system described in chapter 5 does not try to learn an optimal set of image features directly from image data, but works indirectly with feature extraction methods that have been shown to perform well in other situations. This makes learning much more efficient, greatly reducing the amount of data needing to be processed.

The approach we follow first selects image regions to be used for matching between images, then calculates a descriptor for each of these regions. Instead of using this kind of local feature, it is also possible to calculate image features based on global image characteristics, such as the characteristics of the image colour histogram, the global distribution of edge directions in the image, or global shape characteristics (see for example Laaksonen et al., 2002). Another whole-image approach to creating features is to use a mathematical decomposition of the data set such as its principal components (see for example Turk and Pentland, 1991). This kind of global approach cannot easily deal with the presence of multiple objects in an image, or with occlusion.

If local features are being used, the processes of choosing regions-of-interest and calculating descriptors for them can be combined, using a single sequence of filters. Combining region selection and descriptor calculation in this way makes sense if features are being learnt for particular data by optimising filter parameters. For example, Fukushima (1980) trained a neural network with alternating convolutional and subsampling layers to recognise digits across changes in position, while Vaillant et al. (1994) used a similar system to localise faces in images. Here we use region selection and descriptor calculation methods with fixed parameters for efficiency, so the choice of region-of-interest detector method and descriptor method can be made independently.

In our experiments below we use feature detector and descriptor binaries made available by David Lowe for difference-of-Gaussians features,[1] and by the Oxford Visual Geometry Group for other feature types.[2]

2.3 Region-of-interest detectors

A region-of-interest detector processes an image to give a set of regions which can be used as a summary of the image. This section describes the different region-of-interest detectors used in chapters 4 and 6 below. These detector methods are intended to find regions which will be identifiable in other images where the objects or camera have undergone transformations compared to the initial view, and where the lighting may have changed.

We can divide typical image regions-of-interest into edges, corners, and blobs. Edges are boundaries between image areas, providing scene segmentation information. Generally only a small proportion of detected edges correspond to actual object boundaries, while most correspond to transitions on object surfaces. Edge features are not well-suited to the approach described in chapter 4 below, which relies on geometric matching of feature locations to a model, as similar edge features will be detected all along a particular edge. Corner detectors fix feature location, allowing easy geometric matching, by giving features where two edges intersect. Just as not all edge features lie on object boundaries, corner features may arise from object texture rather than shape. Blobs are simply image regions which are brighter or darker

[1]`http://www.cs.ubc.ca/spider/lowe/keypoints/`
[2]`http://www.robots.ox.ac.uk/~vgg/research/affine/`

than their surroundings. Because corners and blobs are geometrically fixed on an object, object parts often give rise to similar features even when they are seen in very different views. In practice many region-of-interest detectors find both corners and blobs.

If we run, for example, a corner detector on a raw image, there will typically be many extraneous detections due to object texture and image noise, while object texture and image noise may obscure some informative object features. If we know the scale of feature which we are interested in, we can smooth the image to avoid detections at smaller scales, by convolving it with a Gaussian kernel. In many cases we do not know the relevant scale for an image, or there may be behaviour of interest at multiple scales. We can make our feature detection scale-invariant by convolving the image with Gaussian kernels at a range of scales, and running feature detection at each scale. If we consider all possible scales, we obtain a three dimensional 'scale space' (see for example Lindeberg, 1994) parameterised by x, y, and scale; detector implementations typically approximate this scale space by considering, for example, each $\sqrt{2}$ increase in scale.

Mikolajczyk et al. (2005) provide a good survey of region detector types. Below we summarise three different region detection methods: the difference-of-Gaussians detector which we use in the experiments in chapter 4, and the Harris affine and MSER detectors which we use in chapter 6.

Each of these region detection methods tries to find the most informative regions of an image given particular criteria. An alternative approach is simply to sample image regions at random. Nowak et al. (2006) show that region sampling can perform well when very large numbers of regions are used; while most detection methods have a ceiling on how many regions they can provide per image – for example, the number of corners – sampling can provide an unlimited number of regions. However, if we want to speed up processing by using relatively small numbers of regions, non-random region selection performs better.

2.3.1 Harris affine

The Harris affine detector (Mikolajczyk and Schmid, 2002; Schaffalitzky and Zisserman, 2002) is an affine-invariant version of the Harris detector.

The ordinary Harris detector finds corner features, by computing the Hessian of the smoothed image at each pixel, and calculating eigenvalues. Where one eigenvalue is significant (above some threshold value), there is an edge; interest points are placed where both eigenvalues are significant. This procedure can be made more efficient by first convolving the image with an appropriate mask, then using an approximation rather than calculating the true eigenvalue decomposition.

The Harris affine detector first uses the Harris method to find interest points. A scale is then chosen for each interest point by examining the response of the local image region to convolution with Gaussian kernels of different scales. Finally the shape of the elliptical region around the interest point is optimised using the eigenvalues of the second moment matrix.

Figure 2.1 shows some example Harris affine region-of-interest detections.

2.3.2 Difference-of-Gaussians

The difference-of-Gaussians detector (Lowe, 1999, 2004) is a fast region-of-interest detector, which tries to detect features that are invariant to scale change, by searching for features which are stable across scales in scale space. It gives circular regions with a scale and orientation, rather than arbitrary ellipses.

Interest points are placed at scale-space extrema of the difference-of-Gaussians function convolved with the image. The required convolution can be efficiently computed as the difference between the smoothed image at two nearby scales. This difference computation can be thought of as an approximation to the scale-normalised Laplacian which is used in the Harris affine detector above.

Candidate interest points are filtered to reject edges and low-contrast extrema, then the orientation for each interest point is found by computing a 36-bin orientation histogram for its local neighbourhood in the Gaussian-smoothed image corresponding to the feature's scale. The bin with the maximum count determines the orientation. If there is another peak within 80% of this count, another feature is created.

Figure 2.2 shows some example difference-of-Gaussians region-of-interest detections.

2.3.3 MSER

The maximally stable extremal region (MSER) detector (Matas et al., 2004) is a blob detector, which searches for regions where all the contained pixels are brighter or darker than all the pixels on the boundary.

MSERs can be found efficiently. First, image pixels are sorted by intensity, then image pixels are marked in this order, maintaining a list of connected components. MSERs are created for minima in the relative change in area of a component as the intensity threshold changes.

The MSER detector we use here replaces each detected collection of pixel locations by an approximating ellipse for further computation. Figure 2.3 shows some example MSER region-of-interest detections.

2.4 Region-of-interest descriptors

A region descriptor processes an image and a set of local regions to give a description for each region. A good region descriptor is invariant to a large range of transformations of a region, so that regions can be matched across images where objects are seen in differing poses and from different viewpoints. Mikolajczyk and Schmid (2005) provide a good survey of region descriptor types. This section discusses the SIFT region descriptor used in chapters 4 and 6 below.

The scale-invariant feature transform (SIFT) descriptor (Lowe, 1999, 2004) is intended to be invariant across a wide range of affine distortion, viewpoint and illumination. It describes the distribution of orientations in local regions, normalised to provide illumination invariance.

The SIFT implementations we use here give a 128-dimensional descriptor, made up from a 4×4 grid of of orientation histograms each with 8 bins. A 4×4 grid is laid down on a copy of the specified image region that has been transformed to a circle and convolved with a Gaussian to filter for edges of appropriate scale. Each grid cell is subdivided into 4×4 smaller regions; the dominant orientation and its magnitude is calculated for each smaller region, and these are collated to generate an orientation histogram for each grid cell. The orientation measurements are weighted by a Gaussian according to their distance from the region centre, and are smoothed across adjacent grid cells. The final feature descriptor vector is normalised to unit length,

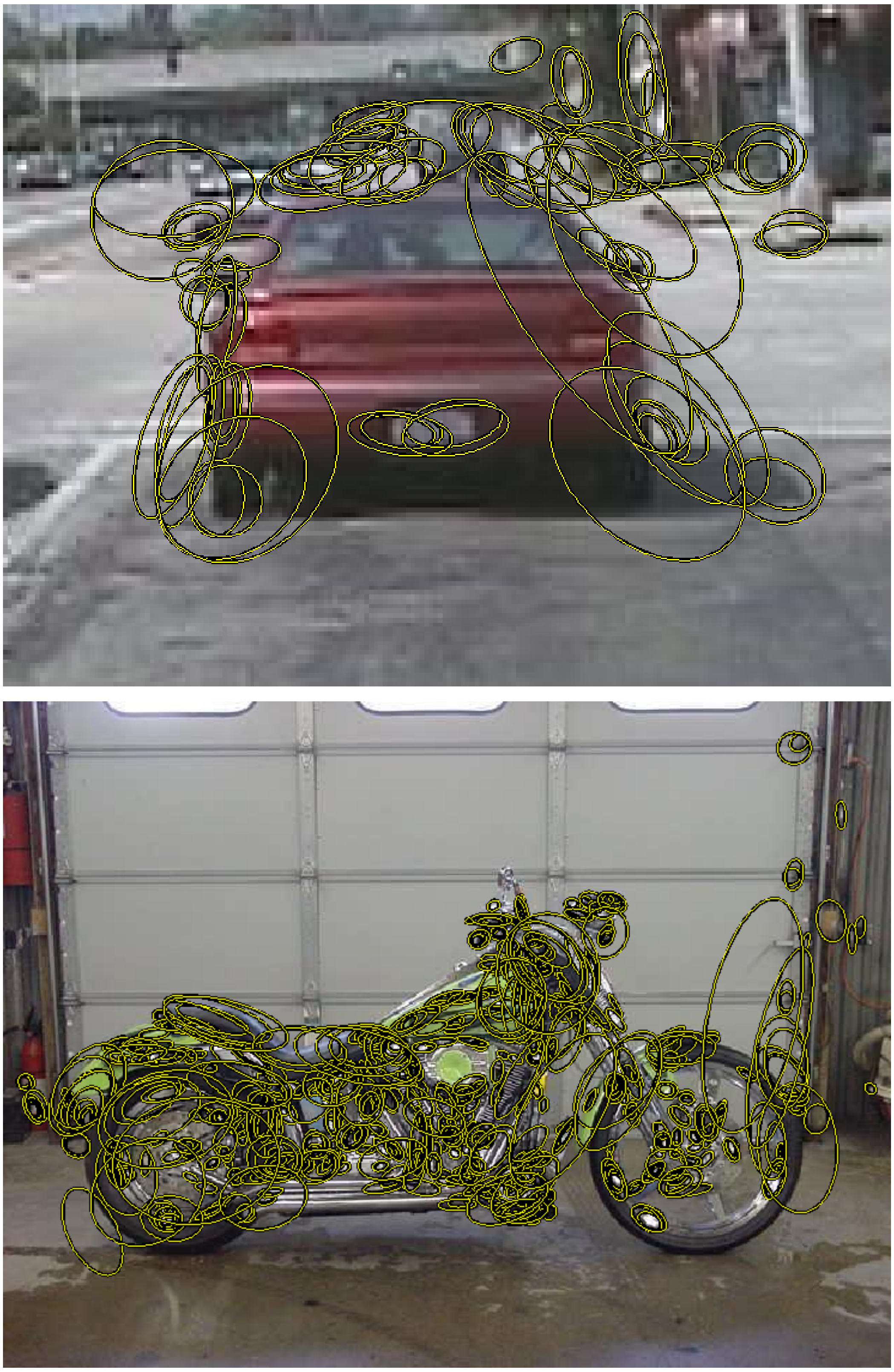


Figure 2.1: Regions of interest found by the Harris affine detector in two sample images.

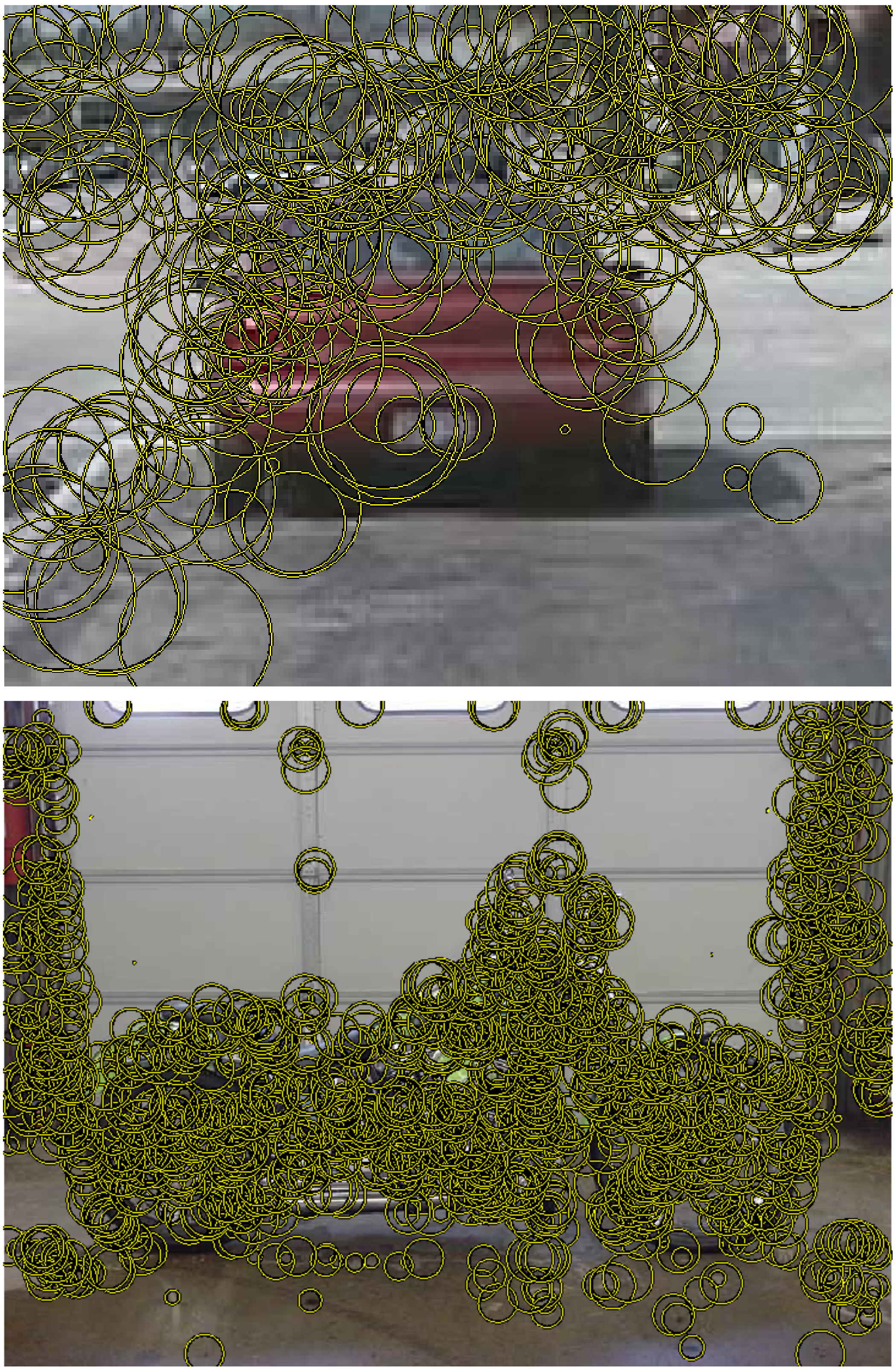


Figure 2.2: Regions of interest found by the difference-of-Gaussians detector in two sample images.

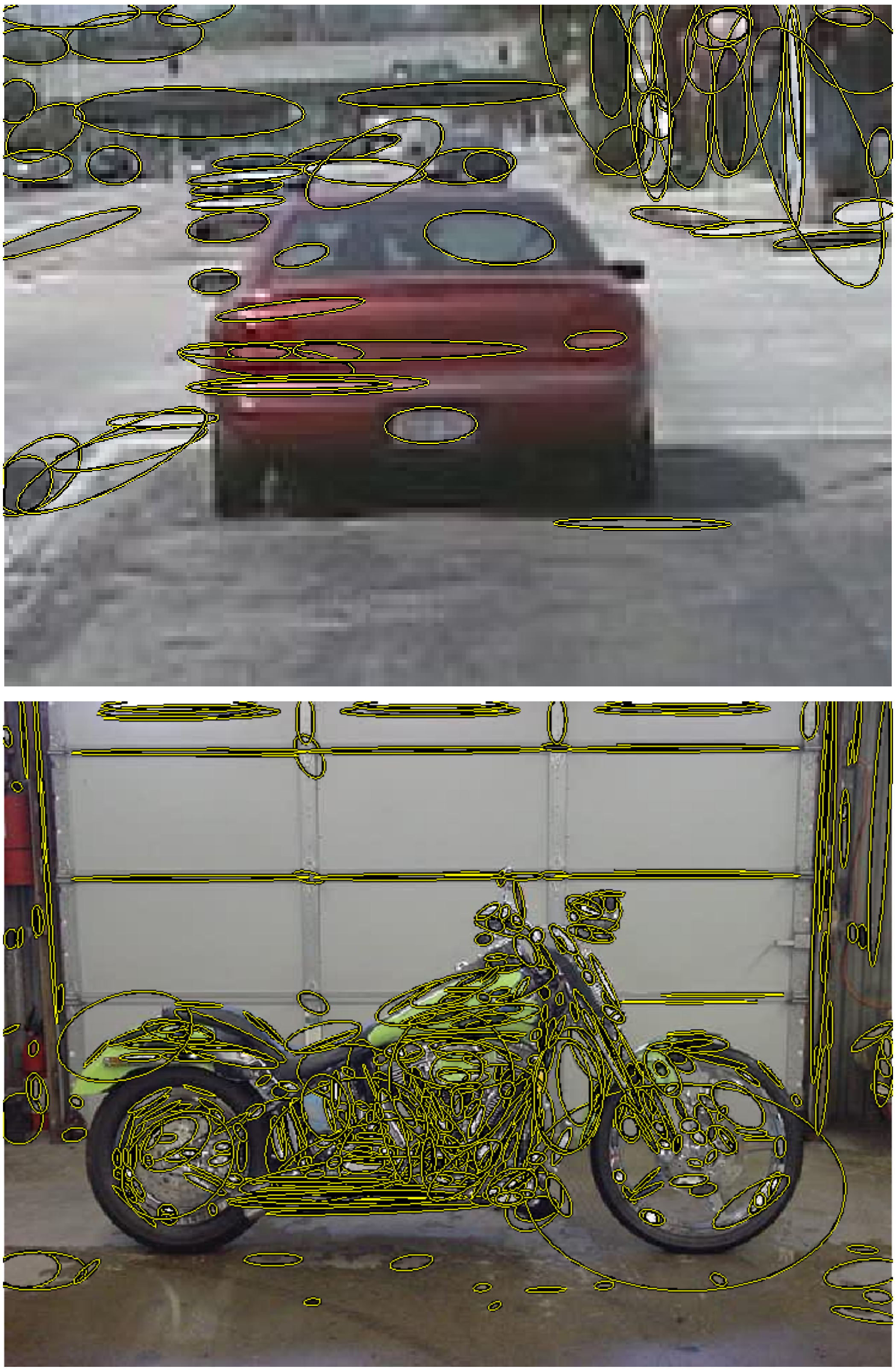


Figure 2.3: Regions of interest found by the MSER detector in two sample images.

with thresholding applied to prevent individual large gradients from swamping the rest of the distribution.

2.4.1 Matching features

The system described in chapter 3 identifies where a specific object appears in a set of images by looking for geometrically-consistent sets of matching features. We can match a new SIFT feature to an existing database by looking for the feature in the database with the minimum Euclidean distance to the new feature.

We use a procedure suggested by Lowe (2004) to reduce incorrect matches. The nearest and second-nearest neighbours in the database are found, based on Euclidean distance in the 128-dimensional feature space. The nearest neighbour is then accepted as a good match if the distance to it is less than 60% of the distance to the second-nearest neighbour. This means that matches are rejected as ambiguous if there are several features in the database which match equally well.

2.4.2 Clustering regions of interest

The system described in chapter 5 clusters SIFT features, vector-quantising the 128-dimensional descriptors to a set of 'visual words'. Object categories are then recognised based on the clusters that are active for the features in a new image. Clustering features to produce a vector quantised descriptor is a common technique in object recognition systems; we use the popular k-means method to learn appropriate clusters from training data.

It is common to expand the detected interest regions before clustering descriptors. Scaling up a detected elliptical region before running the region descriptor allows the descriptor to capture a larger region of object texture, while still anchoring the region on an interest point that should be detectable in other images. Chapter 6 gives some experimental results showing improved object category recognition performance with expanded regions (see Table 6.3).

Chapter 3

Fast learning of sprites using invariant features

3.1 Introduction

This chapter describes a method for fast learning of sprites from image sequences, using invariant features. The goal is to create a system which takes as input multiple views of a scene, and learns an appearance and mask for each of the objects in the scene, and also the background appearance. Chapter 4 describes experiments using the sprite learning system described here, and discusses the results obtained. Some of the material in this and the following chapter is adapted from Allan et al. (2005).

We model a scene as a flat background, with two-dimensional layers in front of it. Each layer, including the background, can move independently. This framework has been widely used in previous work, since for example Wang and Adelson (1994) and Irani et al. (1994). The layer appearance includes the object surface texture and colour, while the layer mask describes which regions of the overall layer are solid, occluding the background and other layers, and which are transparent, allowing the background and other layers to show through behind it. We allow each layer, including the background, to undergo affine transformations, including translation, rotation, and scaling.

The approach described in this chapter combines the use of invariant features (such as SIFT features, Lowe, 2004) with the Titsias and Williams sprite-learning model. First the layer motions are learnt approximately, using invariant features, then pixel-based expectation maximisation is used to learn the layer sprites and masks. Learning object motions during the expectation maximisation phase, without prior tracking, is slow, because we need to allow a large set of discretised transformations for each layer between consecutive images in the sequence. We use invariant features to quickly separate the object motions, and calculate transformations which warp each layer from each image back to a reference frame. A sprite for the object in question can then be quickly learnt using the set of images transformed back to the reference frame. Pixel-based optic flow methods have trouble with regions of low texture, where flow information is unreliable. These regions will also give few identifiable features, but by using motion information about image features in textured areas to seed a dense appearance model we can end up with a good model even for textureless regions.

Beyond the immediate task of sprite extraction, this kind of object learning is a useful precursor to many other tasks. Many human–computer interaction scenarios would be much simplified if the computer system could separate and recognise the objects in a scene. Separating scene images into independent objects allows object recognition to be run more accurately on coherent parts of a scene. Motion-based segmentation aids reasoning about what events are happening in a video scene (Chan et al., 2006). In an extended video, motion segmentation is useful for unsupervised learning about what objects occur frequently (Quack et al., 2006). Representing image sequences as combinations of transformed layers can also be viewed as a compression technique. A set of appearances and masks, together with a set of layer transformations, can provide a much more compact representation of an image sequence than storing all the original images directly (Sawhney and Ayer, 1996).

Section 3.2 below gives a summary of related work, both pixel-based 'direct' methods and feature-based methods. Section 3.3 describes how we match features between images in a sequence, and section 3.4 describes how we learn object motions from the matched features, using a sequence-based version of the RANSAC algorithm. Section 3.5 describes a generative model which can be used to explain the images in a sequence, and gives details of how to combine the feature-based and pixel-based methods.

3.2 Literature review

This section describes some work related to our method for learning objects from image sequences. We discuss ‘direct’ pixel-based methods (section 3.2.1) and feature-based methods (section 3.2.2) for object learning. Pixel-based methods have the advantage of providing an image segmentation and extracted sprites. Feature-based methods provide a higher-level representation of objects that can be used to identify the same object in new scenes: the techniques we describe below aim to find a set of local patches of an object’s surface which can be matched against images of the object seen from a wide range of viewpoints, so as to detect and locate the object in novel views.

We describe here work using video streams made up of time sequences of single image frames. In some applications additional data may be available: for example, Ginhoux and Gutmann (2001) describe an object tracking system which uses pairs of stereo images, while Beal et al. (2003) use audio as well as visual information to track people. Here we deal only with single view video data, treating it as the most basic case to be modelled.

3.2.1 Direct methods

Some object-learning methods work directly from image pixel information, using for example ‘optic flow’ to determine local directions of motion, then clustering the local motions to find overall object transformations.

Irani et al. (1994) describe a system to track objects in a video sequence, and extract their appearance. Their system finds the dominant motion in a sequence, segments the images to exclude the object corresponding to this motion, then repeats the process on the remaining parts of the images. To track each object’s motion they calculate a weighted average of recent frames, transformed to correct for the motion found so far, so that parts of the scene which are not undergoing the same motion blur out over time. At each frame they search over the parameter space of a selected transformation to find the motion which minimises the error between the pixels in that frame and the current internal object representation.

Irani et al. obtain reasonable results, but, for example, their technique of removing background and occluding objects by blurring across frames does not seem ideal: Wang and Adelson (1994) improve on this by modelling scene ‘layers’ which can move independently and occlude each other. Wang and Adelson use optic flow to create a velocity map for each pair of consecutive frames. From the velocity map they estimate affine transformations for local image regions, then use an adaptive *k*-means algorithm to cluster these transformations. Each cluster describes a larger image region with coherent motion. They iteratively calculate an image segmentation in this way and re-estimate motions within the segmented regions, then accumulate these regions across a sequence into moving ‘layers’. Layers are represented by sprites built up by performing an inverse transformation from each frame to a stable view. Their system successfully learns foreground and background layers for simple example video sequences.

Black and Anandan (1993, 1996) use a robust estimation framework to adapt a basic optic flow framework to deal with multiple motions. Their approach is designed to deal with discontinuities in optic flow, but unlike Wang and Adelson they do not divide the motions into layers. Instead they introduce robust statistics into their optic flow estimation, so that they can recover the dominant motion for a whole frame without distraction from a secondary motion. They then detect outliers and estimate the secondary motion from these regions. Black and Anandan claim that their approach is more general, since it can deal with piecewise-smooth flow fields, and that it is more efficient than previous layer methods. However, if we want to learn objects rather than just analyse motion, layers are a natural way to divide up the scene; Black and Anandan do not propose here a convincing alternative to layers for analysing video sequences at a higher level than the motions present between individual pairs of frames.

Weiss and Adelson (1996) suggest a mixture model for motion segmentation based on the optic flow between two images. Compared to previous approaches, they add a prior which enforces spatial consistency. They also discuss how to estimate the number of separate motions that should be modelled, and describe an expectation maximisation algorithm which automatically determines the number of motions.

Jojic and Frey (2001) learn ‘flexible sprites’ from input image sequences. Like Wang and Adelson’s, their system models images as generated from layered sprites, but their sprites are allowed to deform over time: for each sprite they learn a probabilistic appearance map and mask. Each image is generated by superimposing each layer on the previous layers and background, where the masks provide matte alpha channels allowing parts of the foreground layers to be transparent. Sprite motions are found by searching over a discretised transformation space. Jojic and Frey describe a probabilistic model to find the sprites given the image sequence and given the chosen number of layers and sprites. Since exact inference would be intractable, they use an approximate variational inference scheme to learn the object sprites, searching over translational object motions.

Williams and Titsias (2003, 2004) describe how object learning can be made more computationally efficient by using robust statistics and ‘greedy’ sequential learning of objects, learning one object at each stage. Unlike Jojic and Frey, their model allows them to use exact expectation maximisation to learn object models and masks. Their GLOMO algorithm removes pixels belonging to objects already learnt from the images, so that secondary motions will be learnt in their turn. Titsias and Williams (2004) further describe how the GLOMO algorithm can be sped up by approximate tracking of objects before their full structure is known. Beginning with approximate masks excluding any objects already learnt, they search over discrete sets of transformations to track an object to the next frame, then robustly update its mask and appearance model. Titsias and Williams show that using tracking greatly increases the speed with which their system can learn object mask and appearance models.

Even with these speed-ups, searching for object motions in this generative model framework is still slow. The discretised set of transformations searched over for each layer at each frame cannot be made too small or tracking will fail. Fast Fourier transform tricks give increased search efficiency for translational motion, but when more general classes of transformation are allowed the set of discrete transformations that must be considered grows quickly. For example, allowing affine transformations gives each layer at each frame six degrees of freedom, so even if only small motions are considered, the discrete set of transformations is large.

Below we will describe how we can combine the use of invariant features with the Titsias and Williams model to achieve approximate tracking before ‘direct’, pixel-based expectation maximisation learning of object sprites and masks.

Some more recent work has looked at how to generalise layer-based models to deal with changes in layer appearance over time. Winn and Blake (2005) add a dependency between the object appearance in one frame of a sequence and its appearance in the next frame. This allows better tracking of non-rigid objects such as people. Kannan et al. (2005) take this idea further by using a deformation field to explicitly model variation within an object through an image sequence. Kumar et al. (2004, 2005) represent articulated objects by multiple parts, and allow lighting variation and motion blur. Object motion is estimated using cross-correlation between an object fragment and an image, with a Markov Random Field controlling the relationship between the fragment locations. Jojic et al. (2006) also allow deformation in appearance and shape, due to non-rigid motion and illumination change. Hierarchical model selection is used, with switch variables allowing a choice between a set of models of varying expressiveness during expectation maximisation. This makes the overall model more robust against getting stuck in local minima than if the most expressive model, with the most parameters to learn, was used alone.

Another body of work, for example Wexler et al. (2002); Xiao and Shah (2005), has concentrated on improving the pixel-by-pixel quality of the learnt objects. For the best pixel-level results, the learnt mask must be a smooth alpha channel, with individual image pixels sometimes coming from multiple layers. For example, hair viewed from a distance does not produce a hard edge in the mask, since individual hairs are much thinner than an image pixel. Apostoloff and Fitzgibbon (2004) use ground-truth data for example images to learn priors on alpha distribution and spatio-temporal consistency, improving the quality of the layer models they can recover.

3.2.2 Feature-based methods

The object learning methods described above all work directly with pixel data from the images of a video sequence. In this section we will consider methods which work with some higher-level feature representation of the images. These object-learning methods extract features from the images in a sequence, match the features between images, then cluster feature motions to find overall object transformations.

Some older work used extremely simple feature detectors to allow tracking that would otherwise have been computationally intractable – for example, Gee and Cipolla (1995) used a dark-pixel finder to allow efficient head tracking by greatly narrowing the search space for the eyes, nostrils and lips. With increased computing power we can efficiently use more complex types of features. We might consider, for example, an alternative image representation in terms of 'wavelets' (see for example Graps, 1995). Jojic et al. (2003) learn 'epitomes' which represent images in terms of their texture and shape. The object learning methods we discuss below generally use feature detectors which have been manually tuned to represent images in terms of relatively small sets of features which are informative for particular tasks. For example, for object tracking we might want to exclude edge features which may migrate along edges, if we think that these will confuse our tracking algorithm.

Lowe (1999, 2001, 2004) uses the 'Scale Invariant Feature Transform' to create sets of features from images, which are designed to be invariant to image translation, scaling, and rotation. (Mikolajczyk and Schmid (2002) describe a method to find features which are also invariant to significant affine transformations, but for many tasks the additional computation this requires is not necessary.) Lowe uses these features for recognition of specific objects, by learning a set of features from an image of an object then detecting a sufficient subset of the same features in new images. Brown and Lowe (2002) describe how Hough transformation clustering followed by RANSAC (Fischler and Bolles, 1981) can be used to clean the feature matches, removing from consideration matches which are incompatible with a reconstruction of the object's geometry transformed from the view of the object that was originally learnt. Lowe's work here only deals with single views of objects, not video sequences, but his object recognition techniques suggest taking a similar approach to object tracking and learning for video data.

Fergus et al. (2003) and Fei-Fei et al. (2003) also learn invariant features of objects, and use the models they learn to detect objects in new views. Their system is designed to learn not individual objects but wider object classes, such as faces and cars. They use Kadir and Brady's feature detector (Kadir and Brady, 2001) to find features invariant to location and scale. To deal with object classes they need to estimate the variability of object appearance; this kind of estimation is in fact also desirable for individual objects, if we want to correctly recognise them (and reject similar objects) over a wide range of scenes. Views of an object within a video scene, or across multiple scenes, provide information about its variability similar to the information about the variability of an object class from a set of example images of members of that class.

Sivic et al. (2004) describe a system similar to the one we set out below, which uses invariant features to learn objects from video sequences, but they extract 3D sets of points from the data rather than 2D sprite models. They extract two types of features from their data, then process matches between frames in a number of stages to carry out short- and long-range track repair, and aggregate tracks to objects with coherent motion using RANSAC. It is likely that our results below could also be improved by track repair, at the cost of extra computation. They show that the sets of features they learn for an object can be used to detect that object in novel views. Rothganger et al. (2004) use affine-invariant image patches found by a Kanade-Lucas-Tomasi feature tracker to construct a similar model of the rigid three-dimensional parts of a scene. Their object models have local affine patches centred on the feature locations, but do not give overall appearance models. They use a RANSAC clustering algorithm to find motion tracks which are consistent with each other through a video sequence, segmenting the motion tracks into a number of three-dimensional models. To match these models with novel views, they again use a RANSAC algorithm to find matches which are geometrically consistent, generating an alignment consensus from matches with low error, then expanding the match set by including matches with higher error which fit with this alignment. Wills et al. (2003) also extract and match image features, then use RANSAC to cluster feature motions. However, their system only deals with pairs of images, avoiding the additional issues raised by longer image sequences.

A variety of approaches have used matrix factorisation to segment the motions in a scene once individual features have been tracked over time. Costeira and Kanade (1995) describe a multibody factorisation method to perform motion segmentation. Gruber and Weiss (2004) give an improved algorithm which can deal with noise and missing data. They point out that this approach can correctly segment, for example, cylinders rotating at different speeds on the same axis. (Weiss (1997) also includes a discussion of cases such as rotating cylinders which can confuse many motion segmentation algorithms.) Many other approaches which learn object structure from motion in video sequences would segment the front and back of a rotating cylinder as objects moving in opposite directions. Whether or not such oversegmentation is a serious problem depends on the overall model being used in a system – in many applications undersegmentation causes useful data to be lost, but oversegmentation can be corrected in post-processing.

Vidal and Sastry (2003) discuss the case of multiple moving objects seen in two perspective views, and suggest a nonlinear optimisation approach to learn the motion parameters for the objects, avoiding the need to iterate between separate feature segmentation and motion estimation steps. Vidal and Hartley (2004) treat motion segmentation as a subspace clustering problem, allowing missing data and partially-dependent motions; they suggest using power factorisation to project the feature motion data into a five-dimensional space, and generalized principal component analysis to cluster the projected subspaces.

3.3 Feature matching

If we have multiple views of a scene, as in the frames of a video sequence, the objects in the scene can be learnt by looking for regions of the scene which move together. This section describes how we can quickly obtain data for this kind of motion analysis, by extracting features from each frame, and matching between frames to give information about how different object regions move through the sequence.

We process video frames using the difference-of-Gaussians region detector and scale invariant feature transform (SIFT) region descriptor (Lowe, 2004), as described in chapter 2, to find image features designed to be invariant to image translation, scaling, and rotation[1]. Each feature is described by a 128-dimensional feature descriptor whose computation is based on gradient orientation histograms for a number of regions local to the keypoint. (We emphasize that these features are only one example of invariant features and that other features as proposed in e.g. Mikolajczyk and Schmid (2002) could also be used and may possibly give better performance.) By running over the frames in a video sequence, and comparing the image features detected in each frame with the features already found, we build up a 'dictionary' of distinct features, and a 'concordance' which records all the frames in which each feature was seen, and its location in each of these frames. Features from each new frame are compared with those already in the dictionary using Lowe's nearest neighbour matching technique, matching a feature to its nearest neighbour in SIFT feature space if the distance to the nearest neighbour is less

[1]We thank David Lowe for making his invariant feature detector code publically available at `http://www.cs.ubc.ca/spider/lowe/keypoints/`.

than 0.6 times the distance to the second nearest neighbour, as in Lowe (2004). We can reduce the computation needed in feature matching and subsequently in clustering by only adding to the feature dictionary features that have appeared in at least two consecutive frames, which are likely to be the most informative. In the rest of this section we assume that this feature extraction and matching has already been performed, and use 'feature' to refer to all the detected image features across a sequence matched to the same entry in the feature dictionary.

3.4 Motion estimation

This section describes how, given feature information about how regions move through a video sequence, we can combine information about the regions that move together to learn about overall object motions in the sequence.

Given two views of a scene, we can use the RANSAC algorithm (Fischler and Bolles, 1981) to find clusters of image features with consistent motion between the views. Candidate clusters are initialised with randomly-chosen seed features, and expanded by adding other features whose motion is consistent, then the cluster containing the most features is chosen as best. By repeating this process from seed features which have not yet been assigned to a cluster, we can, for example, find all the independently-moving clusters in two frames of video.

If we had full information on the location of all image features in every frame of a video sequence, we could run RANSAC over the frames together to find the independently-moving objects. However, in practice our feature information is sparse, with individual features usually only detected in a few frames of a sequence. One solution to this problem would be to run RANSAC between all adjacent pairs of frames in the sequence, since adjacent frames will typically have many features in common, and then to cluster the clusters from each pair of frames to find the objects. Alternatively, we could run RANSAC between individual frames and a key frame of interest. If we only consider object transformations between adjacent pairs of frames, we may not learn a sufficiently rigid group of object features to be useful for sprite learning; choosing a single key frame would avoid this problem, since rigid transformations can be enforced between the key frame and the other frames of the sequence where the object occurs, but this introduces the problem of choosing an appropriate key frame. Indeed, in some image sequences there may not be any single useful key frame, since there may not be any one frame where all the objects appear together. The approach we suggest instead uses a key frame per object, chosen during a sequence-based RANSAC.

The algorithm described below deals with sparse feature location data, and avoids the problems of reconstructing objects from contradictory clusters in different frames by maintaining a 2D geometric model for each object separate from its view in any individual frame. This geometric model contains all the features which belong to the object, at positions learnt from all the frames where they appear.

Algorithm 1 Object learning algorithm

```
repeat
  Find all the features not yet assigned to an object.
  for each of $n$ candidate objects do
    Initialise a new object with $k$ features from some key frame, which are not yet assigned
    to an object, creating a geometric model containing these features in the positions at
    which they appeared in the key frame.
    repeat
      Use least squares to find transformations which project the features in the object
      model from each frame to their positions in the object's geometric model.
      Project every feature from each frame where it appears to the object using these
      transformations, and calculate its variance in position.
      If the variance across all the frames was below a threshold, and lower than for any of
      the previous objects, add the feature to the object.
    until there is no change in the object, or the maximum number of iterations is reached.
  end for
  Choose the candidate object that accounts for most currently unassigned features.
until all features in the key frame are assigned to an object, or the maximum number of
objects is found.
```

3.4.1 Sequence RANSAC algorithm

The objects in a sequence are learnt one after the other by a sequence-based version of the RANSAC algorithm, summarised in Algorithm 1. This algorithm expands candidate objects by finding all the features in the sequence whose motion is compatible with that of randomly-chosen seed features, then instantiates the candidate which expands to include the greatest number of features. This procedure of creating candidate objects and picking the best is repeated until all the features have been assigned to objects.

We start by randomly selecting a seed frame from the sequence, and randomly choosing seed features from this frame. Since we want to find two-dimensional affine transformations between objects and frames, we need three seed features. An initial geometric model for the candidate object is created from the the three features' positions in the seed frame.

In each of the frames where these three features appear together, we can find an affine transformation between their positions in the frame and their positions in the geometric model. We have

$$\begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} \tag{3.1}$$

$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} x'_2 \\ y'_2 \\ 1 \end{pmatrix} \tag{3.2}$$

$$\begin{pmatrix} x_3 \\ y_3 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} x'_3 \\ y'_3 \\ 1 \end{pmatrix} \tag{3.3}$$

and want to find an affine such that for each feature

$$\begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{pmatrix} x'_i \\ y'_i \\ 1 \end{pmatrix}. \tag{3.4}$$

This gives a system of simultaneous equations with solution

$$a = \frac{(x'_2 - x'_3)(y_1 - y_2) - (x'_1 - x'_2)(y_2 - y_3)}{(x_2 - x_3)(y_1 - y_2) - (x_1 - x_2)(y_2 - y_3)} \tag{3.5}$$

$$b = \frac{(x'_1 - x'_2) - a(x_1 - x_2)}{y_1 - y_2} \tag{3.6}$$

$$c = x'_1 - ax_1 - by_1 \tag{3.7}$$

$$d = \frac{(y'_2 - y'_3)(y_1 - y_2) - (y'_1 - y'_2)(y_2 - y_3)}{(x_2 - x_3)(y_1 - y_2) - (x_1 - x_2)(y_2 - y_3)} \tag{3.8}$$

$$e = \frac{(y'_1 - y'_2) - d(x_1 - x_2)}{y_1 - y_2} \tag{3.9}$$

$$f = y'_1 - dx_1 - ey_1. \tag{3.10}$$

Solving for each frame gives a sequence of affine transformations which describes the motion of the object. The features we should aim to add to the object are those whose positions throughout the sequence are compatible with this motion.

If we already have an affine $\begin{pmatrix} a_k & b_k & c_k \\ d_k & e_k & f_k \\ 0 & 0 & 1 \end{pmatrix}$ mapping an object from each image to its geometric model, and a new feature is added to the object's geometric model at (p_i, q_i), the sum squared error in this feature's projected location is

$$E_i = \sum_{k=1}^{N} (a_k x_{ik} + b_k y_{ik} + c_k - p_i)^2 + (d_k x_{ik} + e_k y_{ik} + f_k - q_i)^2, \tag{3.11}$$

where k iterates over the N images in the sequence. We can decide the best location in the object's geometric model for the feature by minimising the sum squared error:

$$\frac{dE_i}{dp_i} = 2\sum_{k=1}^{N} p_i - a_k x_{ik} - b_k y_{ik} - c_k \tag{3.12}$$

$$\frac{dE_i}{dp_i} = 0 \quad \Rightarrow \quad \begin{array}{l} p_i^* = \frac{1}{N}\sum_{k=1}^{N} a_k x_{ik} + b_k y_{ik} + c_k, \\ q_i^* = \frac{1}{N}\sum_{k=1}^{N} d_k x_{ik} + e_k y_{ik} + f_k. \end{array} \tag{3.13}$$

It is also useful to consider the variance in the feature's projected location, $\frac{1}{N}E_i$, as a measure of how well the feature fits with the overall motion of the object.

In standard RANSAC all the features compatible with a proposed motion cluster can be found directly, but when we cluster features across an image sequence our seed features might only appear together in a small number of frames, and so only give us enough information to find compatible features in a few frames of the sequence. Therefore we need to iterate. At each iteration we look at the frames about which we have information so far, and add to the current object all the features which appear in them and are compatible with its motion there. Any features which we add to the object can give us information about more frames to use at the next iteration.

To judge if a given feature is compatible with the current candidate object we look at all the frames where it appears along with at least three features from the object. We find transformations that map features currently assigned to the object from their locations in the frames to the object with the least squared error, and then project the feature onto the object using each of these these transformations. If the feature matched the current object's motion precisely, it would always be projected to the same position on it. By setting different threshold values for the variance of the projected positions, we can extract tighter or more relaxed motion groupings. If the variance is below our set threshold, and is lower than the variance in projected position which the feature had against the objects we have already extracted, then we assign it to the current object for the next iteration.

A lower variance threshold will give a larger number of smaller motion groupings, and will lead to affine transformations that more closely represent the motions of the features in each grouping. A higher threshold will allow a single motion grouping to approximate a more complex set of feature motions. The optimal threshold value is dependent not only on the video data being used, but on how we want to use the resulting affine transformations. However, for many video sequences a high threshold value ($\sigma = 20$ pixels) will still give good results, since features from one object will fit extremely badly to the motion of other objects. The experiments we report below were run with $\sigma = 4$ pixels. In general we have found that if there appear to be k coherent moving objects in the sequence, these will correspond to to the k motion groupings that contain the largest number of features. Any other motion groupings detected typically correspond to local deviations from the affine model.

We continue iterating until the list of features belonging to the current object stops changing or we reach some maximum number of iterations. Once we have created a number of candidate objects, we pick the one which was able to explain the motion of the largest number of features. We then mark all its features as ruled out from being seeds for any future objects, so that we avoid repeatedly extracting the same motion grouping.

We can continue extracting objects in this way as long as there are at least three potential seed features remaining, but since we will tend to find objects in order by how many features' motions they account for, we may decide to stop once we have extracted some predetermined maximum number of objects.

The algorithm described above is quite similar to that presented in Rothganger et al. (2004), although note that since we are building 2D instead of 3D models the initialization of a model is much easier in our case.

3.5 Sprite learning

This section describes the layered generative model we use to explain the images in a sequence. We use object transformation information learnt by the feature-based sequence RANSAC algorithm described above to speed up object learning compared to using this pixel-based model alone.

In the simplest case we have two layers, a foreground object and static background. Later in this section we show how to extend the model to deal with multiple objects and with a moving background. The background layer has an appearance model $\mathbf{b}$, which is a vector of pixel appearances b_i. Here we treat b_i as an intensity value from a greyscale image, but it can also be a colour vector. The foreground layer has an appearance model $\mathbf{f}$ and a mask π. The mask allows the layer sprite to fit the shape of the object it models. The mask is also a vector with one element per pixel, where each element is the probability that that pixel is generated from the foreground object. The foreground layer may be bigger or smaller than the background.

At each frame in an image sequence the foreground object undergoes some transformation T_j. The sequence RANSAC algorithm above gives a sequence of affine transformations. Given the transformation T_j, an image $\mathbf{x}$ is drawn according to the model described in Williams and Titsias (2004). Each pixel in image j either comes from the background, or is occluded by the foreground object and comes from the transformed foreground $T_j\mathbf{f}$. The latent variable $\mathbf{s}$ is the binary mask of the foreground object in the image, a vector that is one for pixels generated from the foreground object, and zero for pixels generated from the background.

$$P(\mathbf{s}|T_j) = \prod_{i=1}^{P} (T_j\pi)_i^{s_i} (\mathbf{1} - T_j\pi)_i^{1-s_i}, \tag{3.14}$$

where $T_j\pi$ is the transformed mask prior for the foreground and P is the number of pixels in the image. If $(T_j\pi)_i \simeq 0$, pixel i has high probability of coming from the background, while if $(T_j\pi)_i \simeq 1$ it has high probability of coming from the foreground. Each pixel in an image $\mathbf{x}$ is then drawn from a Gaussian,

$$p(\mathbf{x}|T_j, \mathbf{s}) = \prod_{i=1}^{P} [N(x_i; (T_j\mathbf{f})_i, \sigma_f^2)]^{s_i} [N(x_i; b_i, \sigma_b^2)]^{1-s_i}. \tag{3.15}$$

The likelihood of an observed image $\mathbf{x}$ can be found by marginalising over transformations and masks. First marginalising over the latent variable $\mathbf{s}$, the probability of an image given transformation T_j is

$$p(\mathbf{x}|T_j) = \prod_{i=1}^{P} (T_j\pi)_i N(x_i; (T_j\mathbf{f})_i, \sigma_f^2) + (\mathbf{1} - T_j\pi)_i N(x_i; b_i, \sigma_b^2), \tag{3.16}$$

In our experiments we assume a uniform prior over the set of permitted transformations. We use expectation maximisation to learn $\mathbf{b}$, $\mathbf{f}$, π and σ.

It is straightforward to extend the model to deal with a moving background and multiple foreground layers. If the background is not static, $\mathbf{b}$ is bigger than the individual images. With two foreground layers, equation 3.16 becomes

$$p(\mathbf{x}|T_{j_1}, T_{j_2}, T_{j_b}) = \prod_{i=1}^{P} (T_{j_1}\pi_1)_i N(x_i; (T_{j_1}\mathbf{f}_1)_i, \sigma_1^2)$$
$$+ (\mathbf{1} - T_{j_1}\pi_1)_i (T_{j_2}\pi_2)_i N(x_i; (T_{j_2}\mathbf{f}_2)_i, \sigma_2^2) + (\mathbf{1} - T_{j_1}\pi_1)_i (\mathbf{1} - T_{j_2}\pi_2)_i N(x_i; (T_b\mathbf{b})_i, \sigma_b^2). \tag{3.17}$$

3.5.1 Learning the model

Learning the transformations for an image sequence using this model is intractable: the time complexity for searching over transformations for a sequence with L foreground layers and a background layer, each permitted a set of J transformations between images, is $O(J^{L+1})$. Jojic and Frey (2001) suggest an approximate method using variational expectation maximisation, while Williams and Titsias (2004) learnt objects sequentially in a ‘greedy’ manner. We use information from feature-based object learning to avoid this intractable search.

We use the transformations found by the feature-based algorithm described above to provide estimates for each layer's transformations. Effectively each frame of an image sequence is warped through the appropriate inverse transformation for an object to create a new image sequence where the object is stabilised, with the rest of the scene moving around it. We learn the remaining parameters in two stages. First we make an independent prediction for each layer's mask and appearance. It is possible to allow some local search here to try to improve on the transformations given by the feature-based method. Secondly we search across occlusion orders for the layers to find the layer ordering with the highest likelihood. The parameters for all the layers can then be optimised together using this occlusion ordering.

Local search on object transformations can be carried out by maximizing the variational lower bound of the log likelihood,

$$F = \sum_{n=1}^{N} \sum_{j_\ell \in N(j^n)} Q^n(j) \log \frac{P_j p(\mathbf{x}^n | T_j, T_{j_b^n})}{Q^n(j)} \leq \sum_{n=1}^{N} \log \sum_{j_\ell=1}^{J} P_j p(\mathbf{x}^n | T_j, T_{j_b^n}), \tag{3.18}$$

where $Q^n(j)$ is a multinomial distribution over transformations of the object in the image n, so that $Q^n(j) = 0$ for any $j^n \notin N(j^n)$, where $N()$ is the local neighbourhood of transformations, and P_j is a uniform prior over transformations.

Uniform components can be added to the foreground and background models to make them robust (Rowe and Blake, 1995; Williams and Titsias, 2004):

$$p_r(x_i; f_i) = \alpha_f N(x_i; f_i, \sigma_f^2) + (1 - \alpha_f) U(x_i); \;\; p_r(x_i; b_i) = \alpha_b N(x_i; b_i, \sigma_b^2) + (1 - \alpha_b) U(x_i). \tag{3.19}$$

This helps greedy learning where layers occlude each other, and also prevents problems if the affine transformation produced by the feature tracking is inaccurate on a particular frame.

The log likelihood F for each layer can be maximised using variational expectation maximisation, optimising the parameters π, $\mathbf{f}$, and σ. Each object's appearance model $\mathbf{f}$ is initialised randomly, σ to a single high value, and the mask π to 0.5.

3.6 Summary

This chapter described a method for fast learning of sprites from image sequences, using invariant features, and looked at related work. The next chapter looks at some experimental results obtained with the sprite learning system described here.

Chapter 4

Fast learning of sprites: implementation and results

4.1 Introduction

This chapter describes some experimental results on learning sprites for multiple moving objects from video sequences, using the methods described in chapter 3. Some of the material here was published in Allan et al. (2005).

Section 4.2 looks at extracting features from the frames of a video sequence, and matching them to build a database of features that occur in the sequence. Section 4.3 describes the results obtained when we use the sequence RANSAC algorithm as described in section 3.4.1 above to cluster features into objects based on their motion groupings, and examines the effect of the motion consistency threshold parameter. Section 4.4 uses the results from sequence RANSAC together with the pixel-based system of Titsias and Williams (2004), as described in section 3.5 above, to learn a sprite for each of the objects (including the background) in example video sequences.

4.2 Feature matching

Given a number of images making up a video sequence, rather than using the full pixel information from each image we can extract higher-level features that summarise the image contents.

We can then match these features between images, to build up a database of consistent features, with information on the frames of the sequence in which they occur and their locations in those images.

As described in chapter 2, there are many possible feature representations which we can use for image data. In the experiments described in this chapter we use a difference-of-Gaussians region detector and SIFT region descriptor (Lowe, 1999), giving features which are intended to be invariant under image translation, scaling and rotation. We could for example use affine-invariant features, such as those described by Mikolajczyk and Schmid (2002). However, our feature preprocessing over the sequence automatically discards features which do not match between frames, so it might in fact give as much benefit to use a broader, less transformation-invariant, feature set. In video sequences there are often long series of frames where objects' transformations are close to simple translations, and detecting more features in the images could potentially allow us to perform better tracking at such times.

We step through the video frames, building up a database of features. We can think of this database as a 'dictionary' of 'visual words'. At each frame we detect all difference-of-Gaussians features (Figure 4.1), and add to the database all features which do not match a feature already present there. We use Lowe's suggested criterion for matching: we accept a match as correct if the feature-space error between an image feature and the best candidate match in the feature database is less than 0.6 times the error for the second-best match. For each feature in the database, we keep a record of all the frames where it has a match. Once we have run through the whole sequence, the database contains the information we need to quickly find feature matches between given frames: we can simply run through the feature records and check for features which had matches in both the frames in question.

In Figure 4.1 there are some uninformative features detected near the left and right edges of the frames, where the *Groundhog Day* data has black lines which create artificial interest points. While the frames could be trimmed down to remove these black edges, for the examples here the frames have been left at their original size, since these uninformative features are automatically filtered out when we perform feature matching and clustering between frames. Figure 4.2 shows the features which match between three frames of the *Groundhog Day* sequence.

Table 4.1 shows the number of features extracted from each image in the 41-frame *Dekalog*

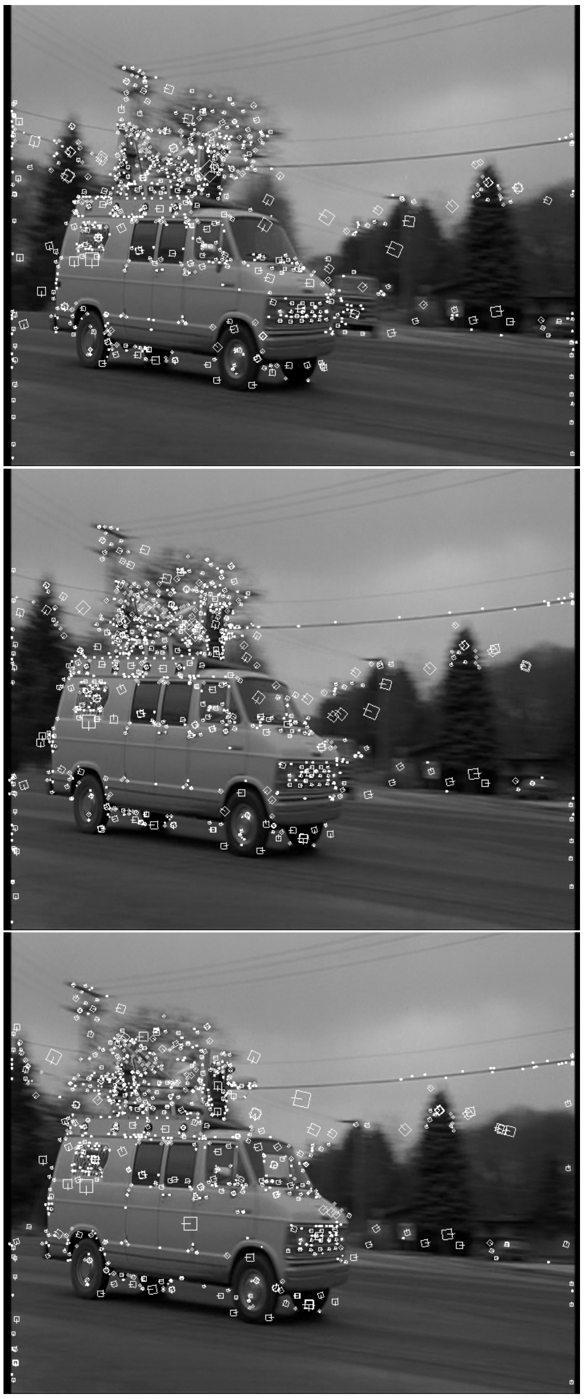


Figure 4.1: Features found in frames 46, 47, and 48 of *Groundhog Day* sequence, showing feature size and orientation.

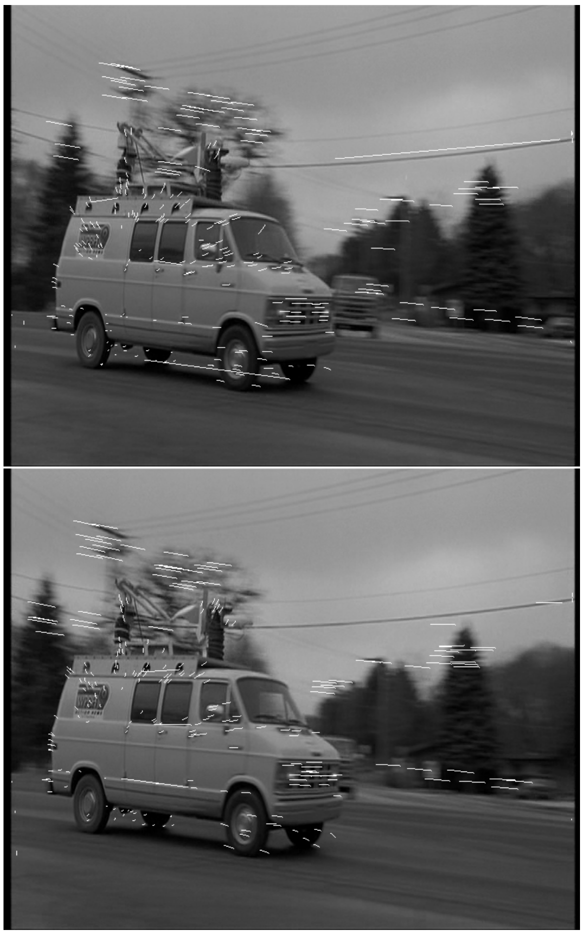


Figure 4.2: Features matched between frames 46 and 47, and frames 47 and 48 of *Groundhog Day* sequence: lines show the motion of a feature between consecutive frames.

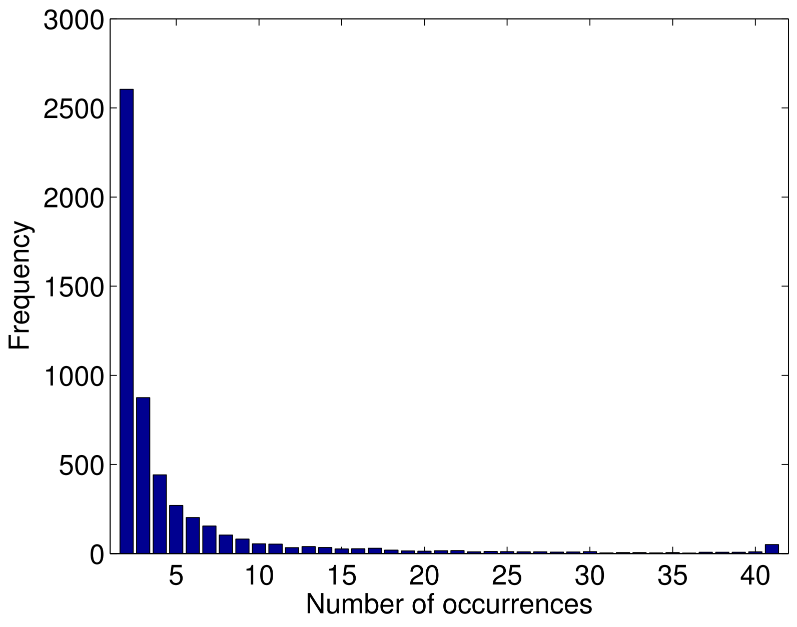


Figure 4.3: *Dekalog* sequence: number of features occuring in a given number of frames.

sequence, how many of these features do not match to any feature already in the database, and a running count of the feature database size. The number of features extracted per frame varies slightly through the sequence, but many fewer new features are added to the database after the first frame, since many extracted features match with features already present in the database. Table 4.2 shows the number of features in the final database which occur in a given number of frames. Figure 4.3 shows the same data as a graph. For comparison, Table 4.3 shows the number of features which occur in a given number of consecutive frames. Most ‘chains’ of occurences in consecutive frames are short, but our feature matching algorithm increases the data available for object learning by also using matches between more distant frames.

Figure 4.4 shows the frames where each feature is present in the *Dekalog* sequence, for a sample of 200 features. Most of these features only occur in a few frames of the sequence. There are a few long ‘chains’ of consecutive occurences, but even features that are present over a large part of the sequence tend to be absent in many of the frames within that part.

After features have been extracted from the last frame in the sequence, all features which have not matched to a feature found in any other frame are discarded, so the final dictionary size falls to less than it was after the first seven frames of the sequence without this check. Table

Frame	Features	New features	Total features kept
1	1576	1576	1576
2	1565	781	2357
3	1538	656	3013
4	1543	634	3647
5	1545	689	4336
6	1508	670	5006
7	1580	796	5802
8	1520	728	6530
9	1501	785	7315
10	1509	779	8094
11	1469	757	8851
12	1470	790	9641
13	1510	831	10472
14	1408	814	11286
15	1452	859	12145
16	1427	831	12976
17	1435	876	13852
18	1463	957	14809
19	1439	945	15754
20	1445	974	16728
21	1455	972	17700
22	1441	1004	18704
23	1450	978	19682
24	1365	901	20583
25	1428	992	21575
26	1380	956	22531
27	1314	916	23447
28	1349	959	24406
29	1321	928	25334
30	1321	949	26283
31	1329	967	27250
32	1313	930	28180
33	1276	912	29092
34	1302	948	30040
35	1283	924	30964
36	1332	961	31925
37	1304	947	32872
38	1357	981	33853
39	1390	1010	34863
40	1442	1063	35926
41	1462	1100	5312

Table 4.1: Feature matching statistics for *Dekalog* sequence.

Number of occurrences	Frequency
(1)	(31714)
2	2604
3	875
4	441
5	270
6	202
7	155
8	104
9	81
10	55
11	53
12	33
13	39
14	34
15	26
16	27
17	30
18	19
19	15
20	13
21	16
22	17
23	10
24	12
25	11
26	10
27	10
28	8
29	9
30	11
31	4
32	6
33	6
34	4
35	6
36	3
37	7
38	7
39	7
40	10
41	50

Table 4.2: *Dekalog* sequence: number of features occuring in a given number of frames.

Consecutive occurences	Frequency
(1)	(7440)
2	2497
3	726
4	359
5	213
6	139
7	104
8	74
9	56
10	36
11	28
12	30
13	22
14	22
15	18
16	9
17	21
18	9
19	5
20	9
21	10
22	6
23	5
24	3
25	5
26	6
27	6
28	8
29	8
30	1
31	2
32	1
33	1
34	1
35	4
36	1
37	1
38	1
39	1
40	1
41	50

Table 4.3: *Dekalog* sequence: number of features occuring in a given number of consecutive frames.

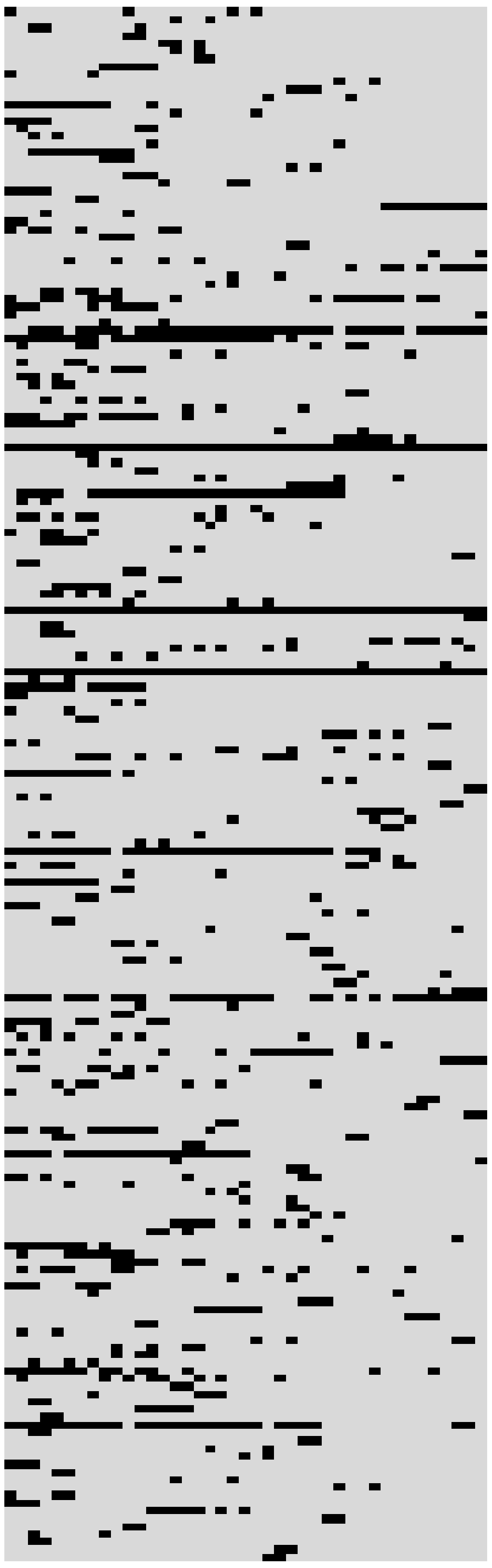


Figure 4.4: *Dekalog* sequence: feature presence/absence in each frame for a sample of 200 features. Each row represents a single feature in the database; each column represents a single frame in the sequence.

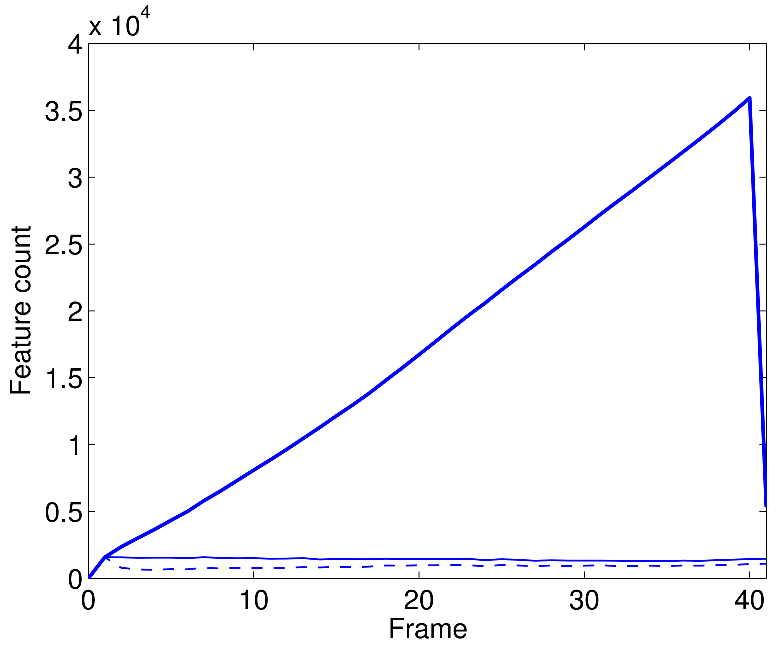


Figure 4.5: Feature counts for each frame in the *Dekalog* sequence: features per frame (thin line), new features (dashed line), total features kept (thick line).

4.1 and Figure 4.5 show how the database size grows roughly linearly after the first frame of the sequence, then drops again after matching has been performed between features from all frames and unmatched features can be discarded.

Processing of long sequences can be sped up, and memory requirements reduced, by discarding features that are only seen once more quickly. For example, features which have not been seen again 10 frames after they first occured are unlikely to be informative. For the short sequences given as examples here, this speed-up is unnecessary, so we match and store features even if they only occur in two distant frames of the sequence.

4.3 Clustering features into objects

After the feature 'dictionary' is built, the sequence RANSAC algorithm described in section 3.4.1 above is used to cluster features into objects according to independent motion groupings. Figure 4.6 shows the frames in which a sample of 200 features from the *Dekalog* sequence are present (as in Figure 4.4), coloured according to the objects to which they are assigned. Figure

4.7 shows some example frames of the *Dekalog* sequence, with the features again coloured according to the objects to which they are assigned.

The great majority of the features belong to the near background (green) and far background (red); out of the two foreground objects, more features are found for the rear car (blue) than the one which occludes it (yellow), since the car nearest the camera is quite blurry so gives fewer good regions of interest.

4.3.1 Effect of motion consistency threshold

As described in section 3.4.1, the sequence RANSAC algorithm has a parameter which controls how similar features' motions are required to be before they are assigned to the same grouping. Table 4.4 shows the number of objects found in the *Dekalog* sequence for different settings of this parameter value: for example, when $\sigma = 4$ pixels, as used in the results shown so far in this chapter and in the other experiments below, 7 objects are found. Figures 4.8 and 4.9 show how the feature classifications change as the motion consistency threshold is varied, overlaying all the feature motions in the sequence on a single frame and colouring the resulting tracks according to the objects to which they are assigned. Large values of σ assign all the features to a single object; as σ is reduced, more objects are found to explain increasingly fine differences in motion.

Table 4.5 shows the number of features assigned to each object in each frame of the *Dekalog* sequence with $\sigma = 4$ pixels. Objects 1 and 7 correspond to the distant and near background (red and green); objects 2 and 3 correspond to the two moving cars (blue and yellow). The other motion groupings arise because the affine model we use in the sequence RANSAC algorithm is not sufficient to explain all the feature motions seen in the sequence. For example, parts of the trees in the background wave in the wind, and all the features of each car, if taken together, display non-affine motion in relation to the camera position. If we simply wanted to group features, it might be better to use a less tight motion consistency threshold which gave fewer motion groupings, but for sprite extraction we would like each affine motion grouping to be as tightly defined as possible.

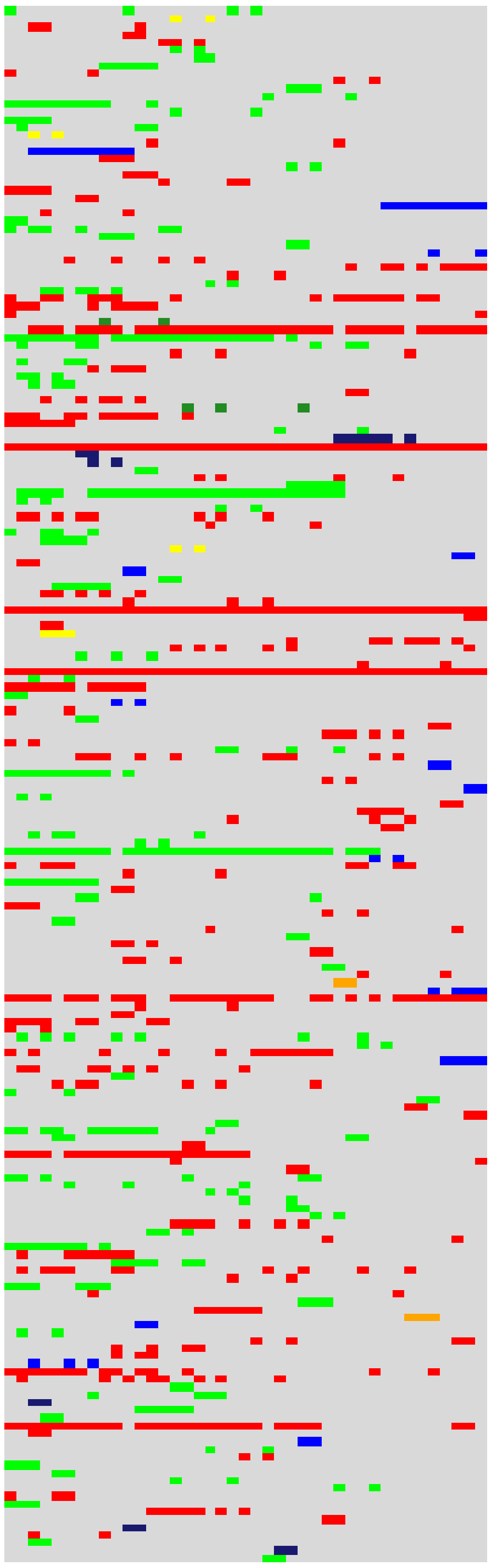


Figure 4.6: *Dekalog* sequence: feature classifications for a sample of 200 features.

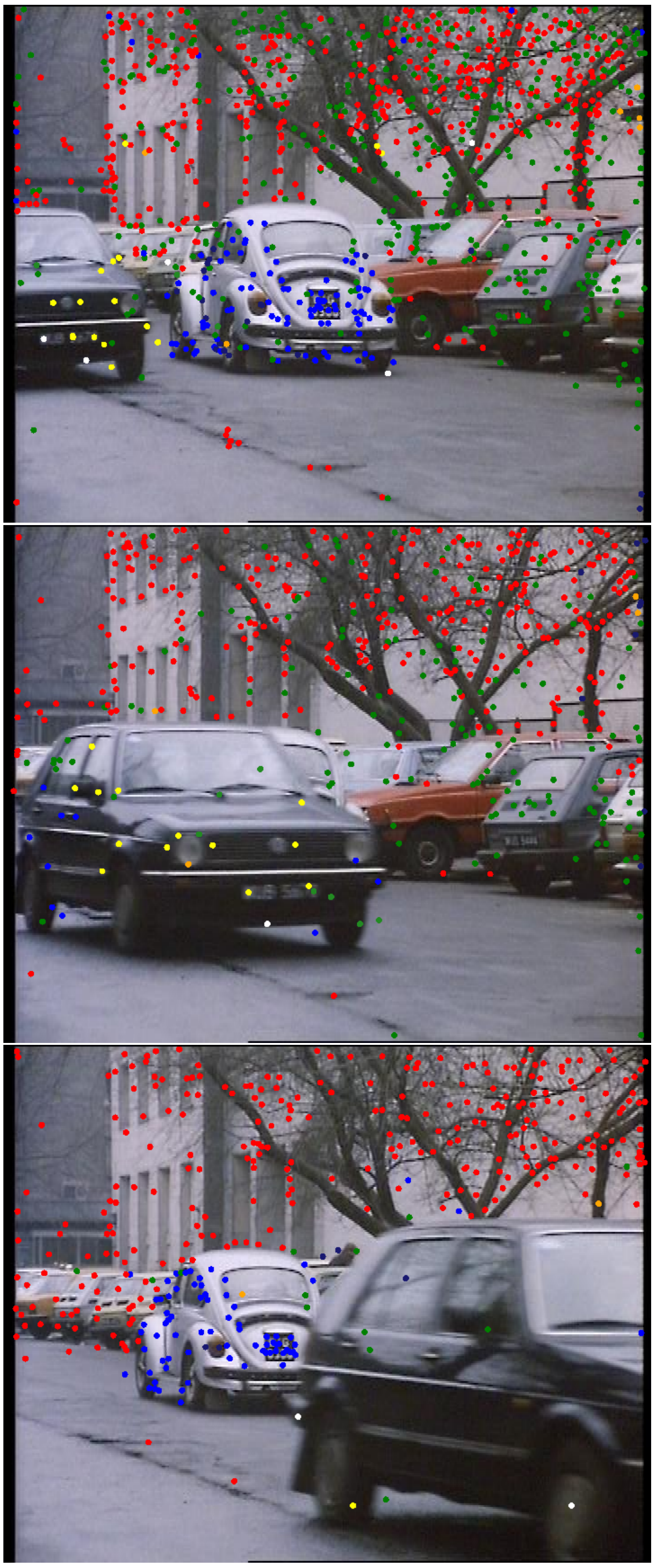


Figure 4.7: *Dekalog* sequence, frames 1, 20 and 39 with feature classifications.

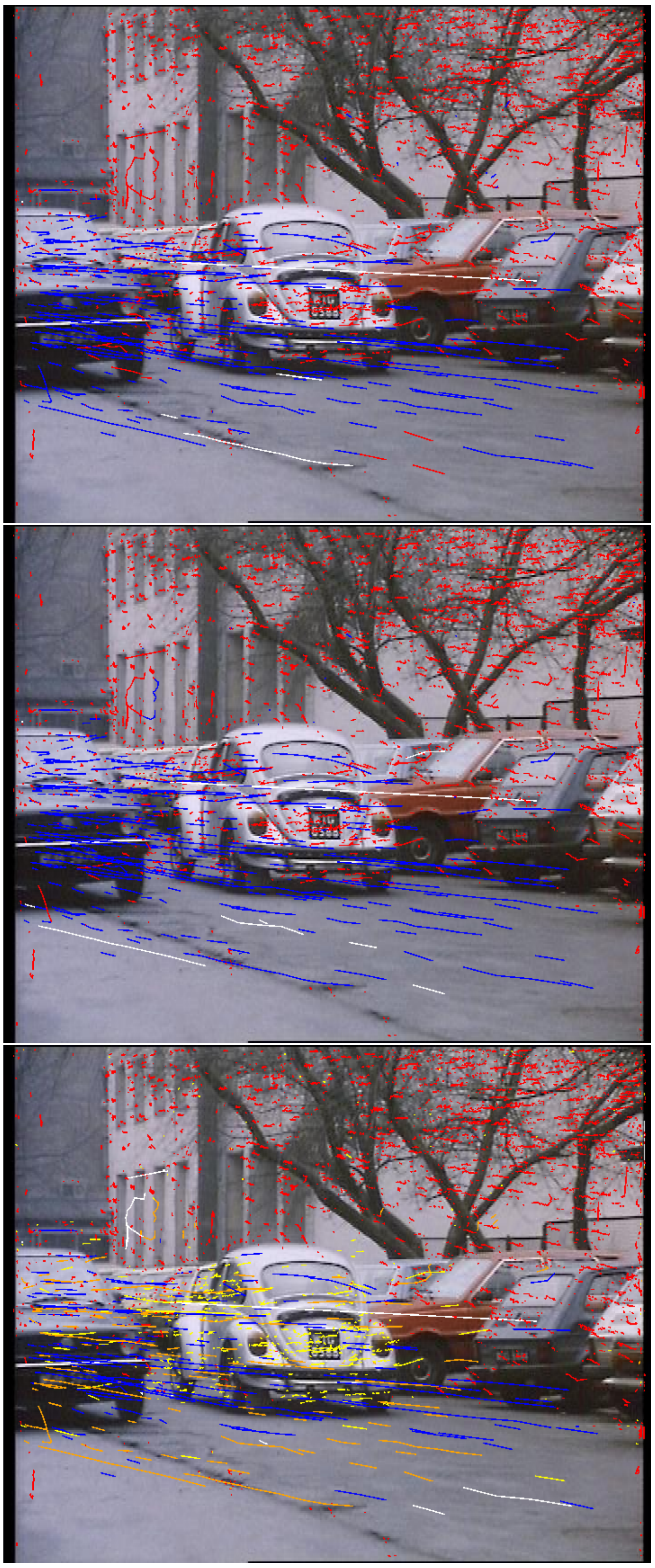


Figure 4.8: *Dekalog* sequence: classification of features as the motion consistency threshold is varied: top to bottom, 32, 16, 8.

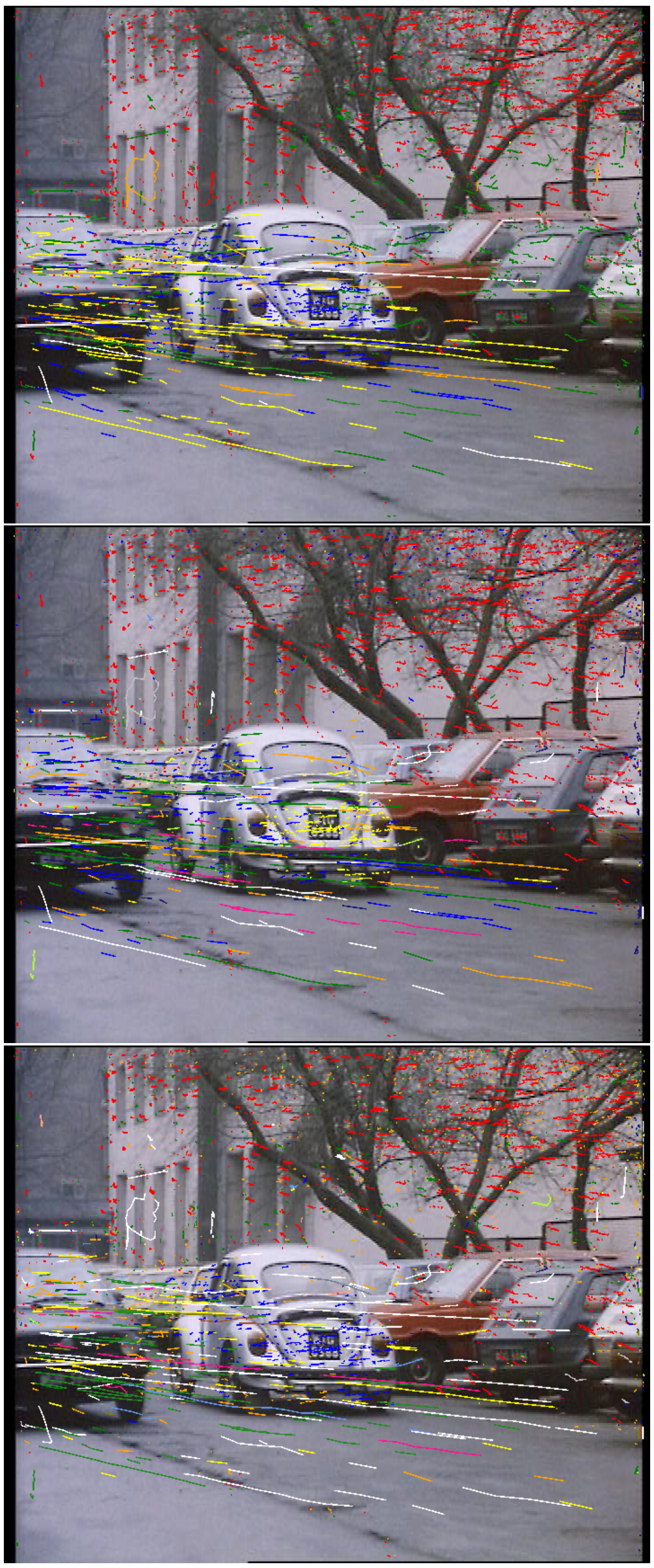


Figure 4.9: *Dekalog* sequence: classification of features as the motion consistency threshold is varied: top to bottom, 4, 2, 1.

Threshold	Objects found
1	15
2	10
3	11
4	7
5	8
6	3
7	4
8	4
9	4
10	3
11	3
12	2
13	3
14	3
15	3
16	2
32	2
64	1

Table 4.4: *Dekalog* sequence: number of objects found as the motion consistency threshold is varied.

4.4 Layered generative sprite model

So far this chapter has considered learning an object's motion and the features which belong to the object. For many applications it is useful to extract a sprite model of the object. The features matched in a given frame already define an implicit mask for an object, and we could use this information to seed an image segmentation algorithm, but to extract reliable sprites for objects we need to combine sprite information across frames. This section describes results using the method described in section 3.5 to learn a sprite for each object in a video sequence, linking our feature-based object learning system to the pixel-based system of Titsias and Williams (2004).

We use the sequence RANSAC algorithm to find motion clusters, then learn sprites corresponding to the motions found. The sprite extraction algorithm receives as input the affine transformations for each object and learns the mask and appearance for each object. Our MATLAB implementation, which is by no means has been optimised for speed, uses the MATLAB routine `imtransform` to carry out affine transformations of images.

Below we describe the results obtained on two different image sequences. The affines

Frame	Unclassified	Obj. 1	Obj. 2	Obj. 3	Obj. 4	Obj. 5	Obj. 6	Obj. 7
1	5	523	129	17	6	23	4	469
2	3	522	128	15	6	31	6	474
3	4	537	114	15	8	28	7	452
4	1	524	115	15	5	27	6	409
5	3	529	106	16	4	23	7	370
6	3	520	93	11	3	26	6	327
7	2	529	98	6	5	22	7	284
8	2	525	87	6	4	25	8	258
9	2	540	81	6	3	22	9	218
10	1	520	88	7	8	19	7	206
11	1	523	81	9	9	13	5	203
12	1	443	62	10	9	15	6	243
13	1	427	57	10	8	13	7	249
14	0	389	43	14	10	13	8	240
15	0	398	34	18	9	12	6	216
16	1	351	28	17	9	11	9	249
17	0	357	12	16	9	17	10	197
18	1	398	8	25	7	17	9	152
19	1	387	4	11	6	9	6	155
20	1	332	10	13	4	10	8	166
21	0	375	11	8	3	4	7	144
22	0	316	21	14	5	7	4	153
23	0	346	23	14	8	6	5	147
24	0	322	24	3	6	5	4	150
25	1	329	15	8	2	6	4	156
26	1	329	9	16	3	5	3	130
27	1	318	12	18	3	3	1	105
28	1	319	10	17	3	4	1	96
29	0	329	4	21	5	11	1	79
30	0	334	3	22	5	16	1	54
31	1	337	2	20	5	16	0	53
32	0	346	10	20	3	17	0	45
33	1	332	25	10	4	12	1	31
34	0	357	36	7	6	10	0	13
35	1	334	44	3	8	5	0	15
36	1	339	47	3	6	1	0	18
37	1	316	62	2	5	4	0	24
38	2	323	64	1	4	4	0	23
39	2	311	81	1	2	5	0	17
40	1	306	83	0	1	3	0	18
41	0	269	71	0	0	2	0	17

Table 4.5: *Dekalog* sequence: number of features in each frame allocated to each object, at motion consistency threshold $\sigma = 4$ pixels.

Frame	Features	New features	Total features kept
1	1302	1302	1302
2	1263	538	1840
3	1268	438	2278
4	1207	378	2656
5	1185	407	3063
6	1127	422	3485
7	998	386	3871
8	990	407	4278
9	979	505	4783
10	923	449	5232
11	1022	495	5727
12	1278	657	6384
13	1439	712	7096
14	1012	495	7591
15	1107	498	8089
16	1003	534	8623
17	1131	611	9234
18	1050	551	9785
19	960	469	2815

Table 4.6: Feature matching statistics for books sequence.

from sequence RANSAC were used directly, without local search. Video files illustrating these results can be found at `http://www.tardis.ed.ac.uk/~moray/features/`.

4.4.1 Books sequence

The 'books' sequence is a 19 frame stop-motion film of two books moving independently against a background of three further books (see Figure 4.10). Each frame is of size 640×480. The foreground books undergo translation and rotation, and one of the books occludes the other for several frames. The camera moves slightly between frames, so the overall transformations between frames involve further translation, rotation, and scaling.

Extracting features from the images and matching the extracted features across the sequence to compile a feature dictionary of distinct features took around 128 seconds. The resulting feature dictionary contained 2815 features. Table 4.6 and Figure 4.11 show the number of features extracted from each image, how many of these features do not match to any feature already in the database, and a running count of the feature database size. Table 4.7 and Figure 4.12 show the number of features which occur in a given number of frames.

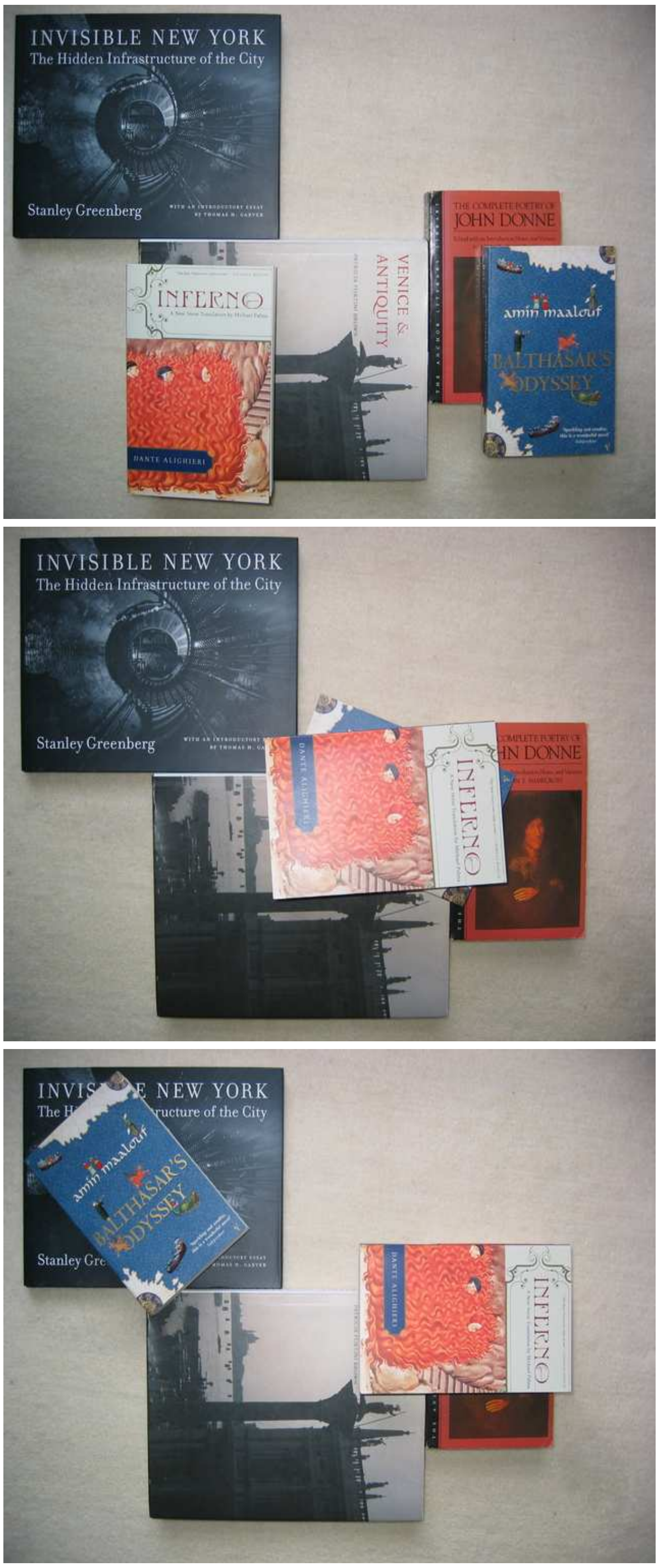


Figure 4.10: Some example frames from the books sequence.

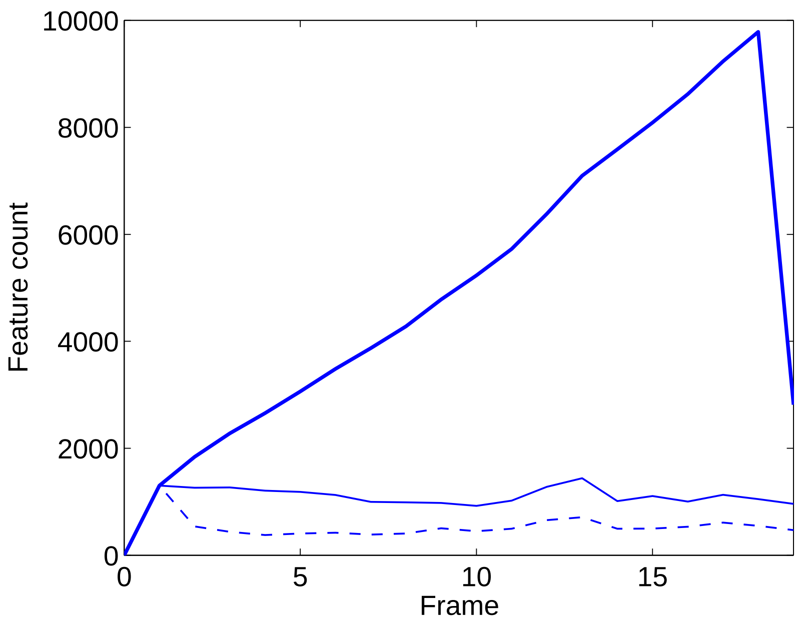


Figure 4.11: Feature counts for each frame in the books sequence: features per frame (thin line), new features (dashed line), total features kept (thick line).

Number of occurrences	Frequency
(1)	(7439)
2	1077
3	429
4	265
5	194
6	158
7	114
8	93
9	69
10	65
11	51
12	47
13	64
14	23
15	28
16	32
17	15
18	12
19	32

Table 4.7: Books sequence: number of features occuring in a given number of frames.

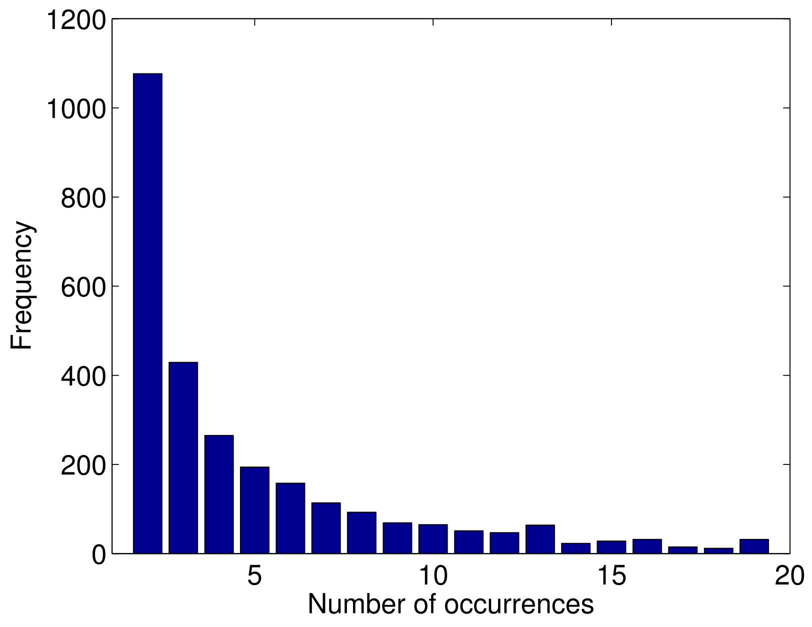


Figure 4.12: Books sequence: number of features occuring in a given number of frames.

Sequence RANSAC terminated after 4 seconds, when it finished assigning features to three objects. In all the experiments here we produced 20 candidates at each stage of sequence RANSAC. One of the objects corresponded to the background, a second to the right-hand moving book, and a third to the left-hand moving book.

Figures 4.13, 4.14 and 4.15 show some example frames transformed onto the geometric models for the background and the two books, stabilising the image sequence with respect to each object.

The EM algorithm took 51 minutes to learn the background and the two books in the image sequence, using 25 iterations of EM and no local search. A method that needs to search explicitly over J transformations would take J times longer. For the books sequence a suitable discretization scheme would need to consider at least translations and rotations. Unless additional information can be brought to bear we would need to consider 640×480 translations and rotations at perhaps 2° intervals (although FFT tricks could be used for the translations).

Figure 4.16 shows the element-wise products of the masks (thresholded at 0.5) with the books' appearance models, and the appearance model for the background. We can see that the objects in this sequence have been extracted well. The shadow of the second book, prominent

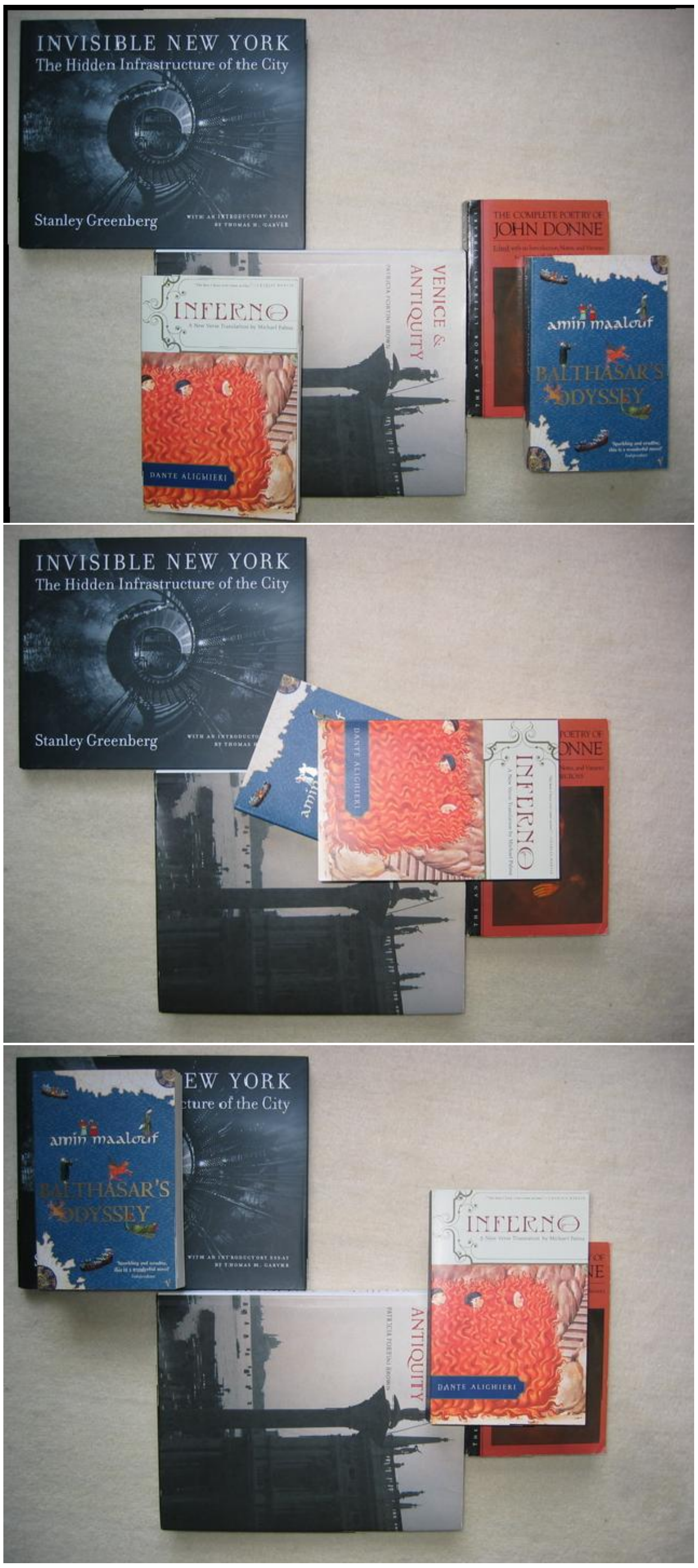


Figure 4.13: Example frames from the books sequence, stabilised with respect to the background.

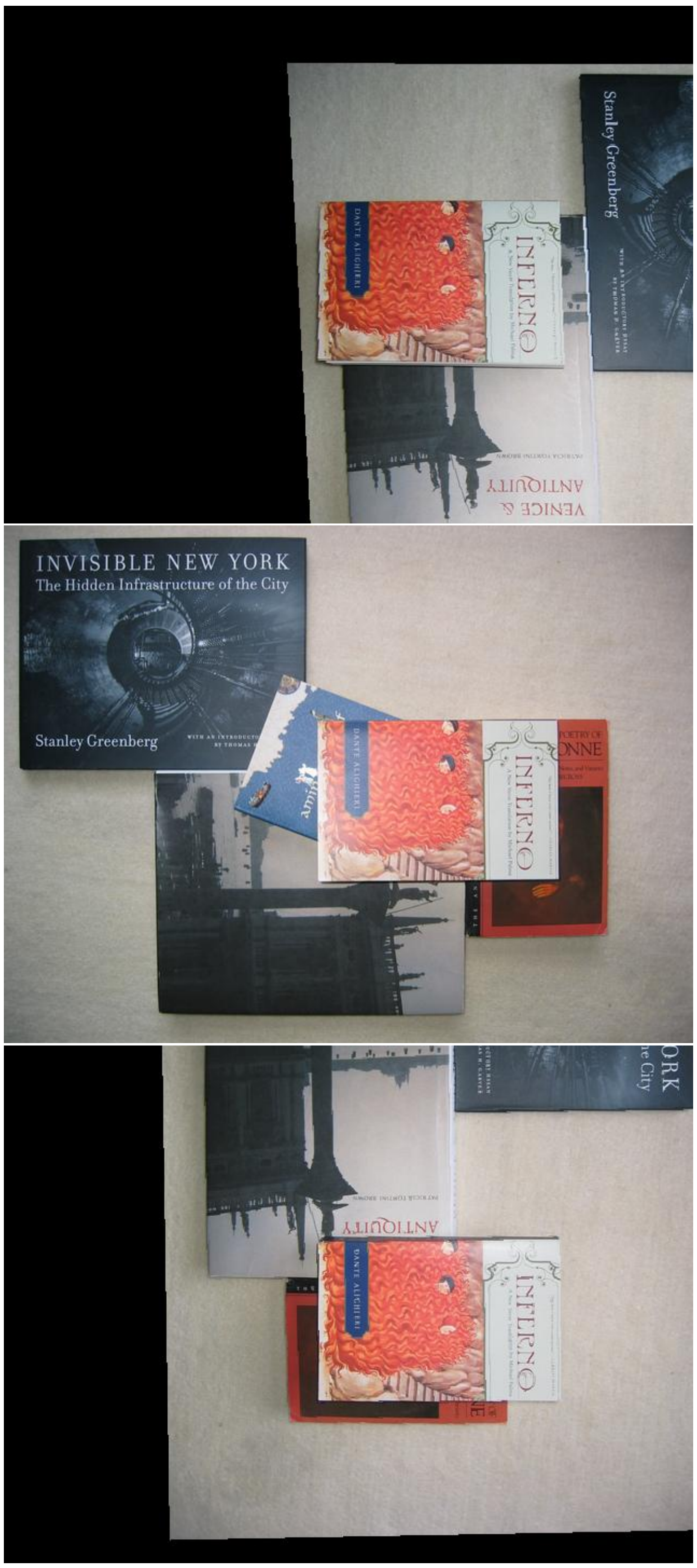


Figure 4.14: Example frames from the books sequence, stabilised with respect to one of the moving books.

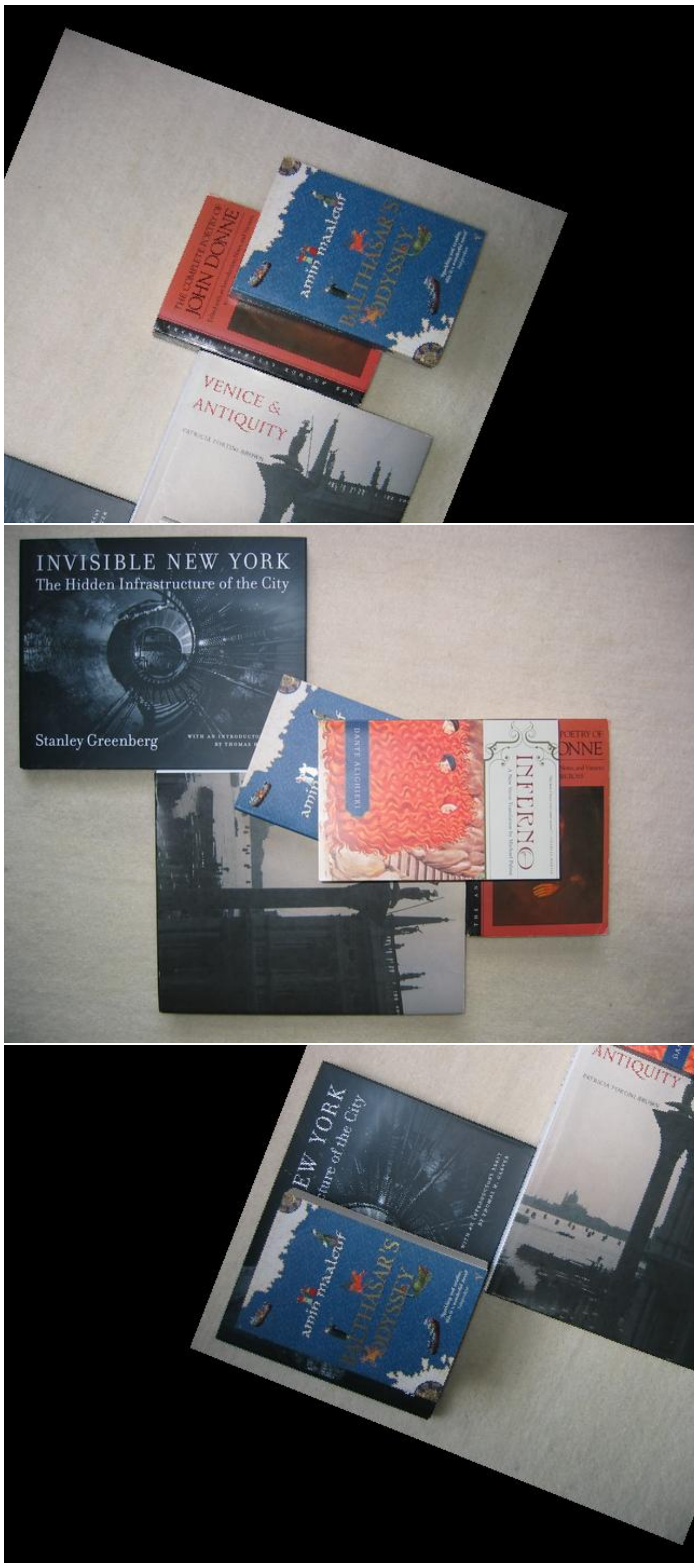
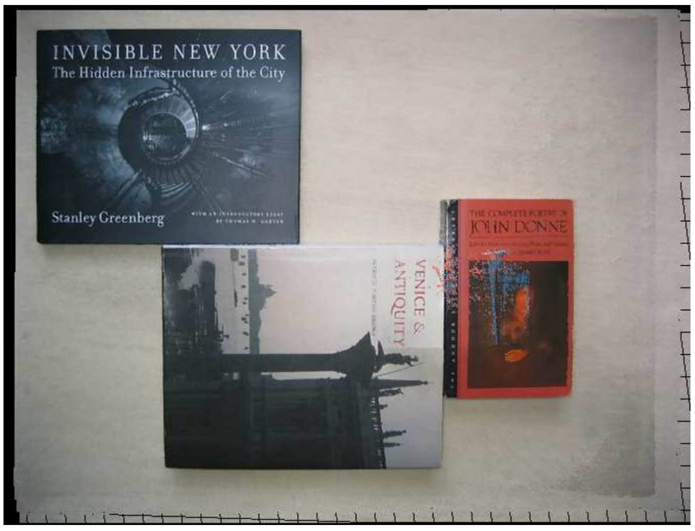
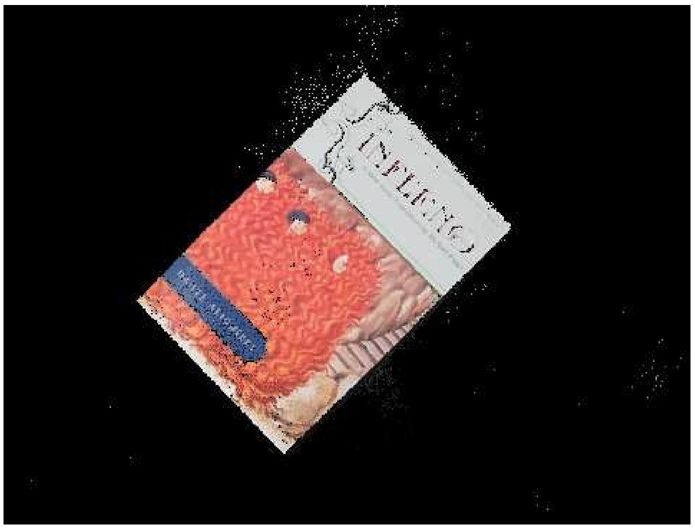


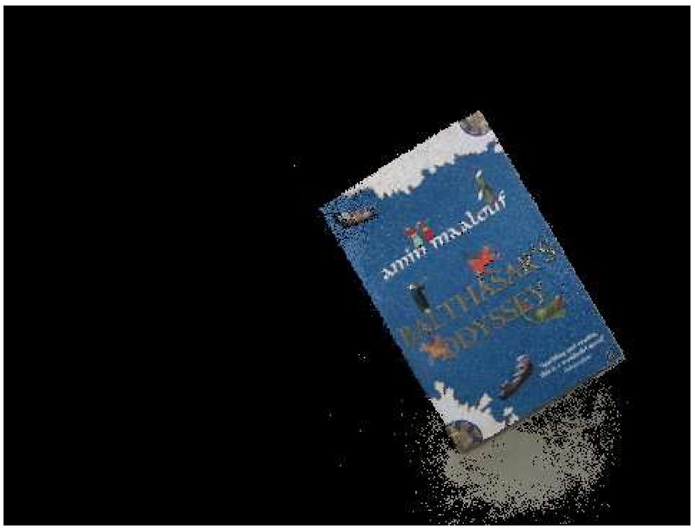
Figure 4.15: Example frames from the books sequence, stabilised with respect to one of the moving books.


INVISIBLE NEW YORK
The Hidden Infrastructure of the City
Stanley Greenberg
VENICE & ANTIQUITY
THE COMPLETE POETRY OF JOHN DONNE

(a)



(b)



(c)

Figure 4.16: Panel (a) displays the learnt background. Panels (b) and (c) show the element-wise product of the thresholded mask and appearance model for the two books.

at the start of the sequence, has been included in the mask for that book.

4.4.2 *White* sequence

Our second experiment uses a 35-frame, 720×576, sequence from the film *White* (see Figure 4.17). A jeep drives from the distance in the centre of the frame along a road passing to the camera's left. The camera pans to follow the jeep, viewing more of the background scene over the course of the shot than is visible in each frame.

Feature extraction and matching took 99 minutes, giving a feature dictionary of 17475 features. We compared each feature from an individual frame to entries in a flat list of features already in the dictionary; the time for matching could be greatly reduced by using a more complex data structure that would allow quicker access to the features most likely to match, as described in Lowe (2004). Table 4.8 and Figure 4.18 show the number of features extracted from each image, how many of these features do not match to any feature already in the database, and a running count of the feature database size. Table 4.9 and Figure 4.19 show the number of features which occur in a given number of frames.

Sequence RANSAC found objects corresponding to the background and the jeep in 87 seconds. Figures 4.20 and 4.21 show some example frames transformed onto the geometric models for the background and the jeep. Note that these images have only been truncated to the frame boundaries to give a consistent display; the model does not limit the objects' size. Sprite extraction took 31 minutes, using 30 iterations of EM.

The sprite models learnt for the jeep (with the mask thresholded at 0.5) and for the background are shown in Figure 4.22. Since the camera pans to view a background scene larger than an individual frame, the background sprite we extract is a wide panoramic image. Part of the centre of the background sprite is blank, since this area is occluded by the jeep throughout the sequence, but the sprite is otherwise good. The jeep and its attached shadow have also been learnt successfully.

4.4.3 Discussion

The results above show that the proposed algorithm for sprite learning is effective, successfully learning appearance models and masks for the foreground objects and background of the

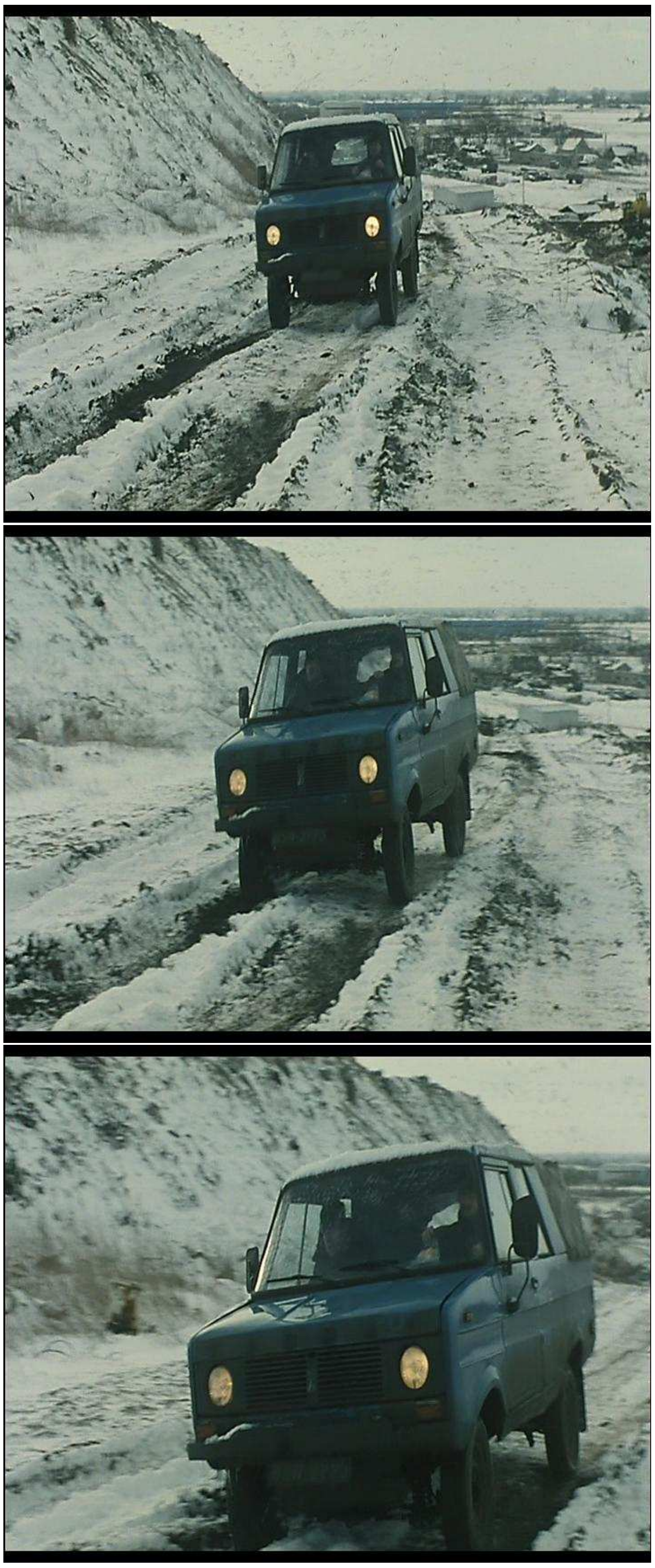


Figure 4.17: Some example frames from the *White* sequence.

Frame	Features	New features	Total features kept
1	7030	7030	7030
2	6796	3863	10893
3	6774	3632	14525
4	6792	3696	18221
5	6787	3798	22019
6	6826	4081	26100
7	6841	4266	30366
8	6814	4283	34649
9	6721	4585	39234
10	6742	4833	44067
11	6705	4798	48865
12	6232	4599	53464
13	6213	4572	58036
14	5825	4324	62360
15	5583	4128	66488
16	5260	3928	70416
17	5196	3942	74358
18	5128	3819	78177
19	4578	3512	81689
20	4390	3372	85061
21	4212	3196	88257
22	4066	3087	91344
23	3971	2993	94337
24	3998	3030	97367
25	3761	2977	100344
26	3637	2845	103189
27	3503	2768	105957
28	3305	2623	108580
29	2981	2382	110962
30	2754	2239	113201
31	2626	2054	115255
32	2579	2068	117323
33	2472	1933	119256
34	2569	1986	121242
35	2509	1963	17475

Table 4.8: Feature matching statistics for *White* sequence.

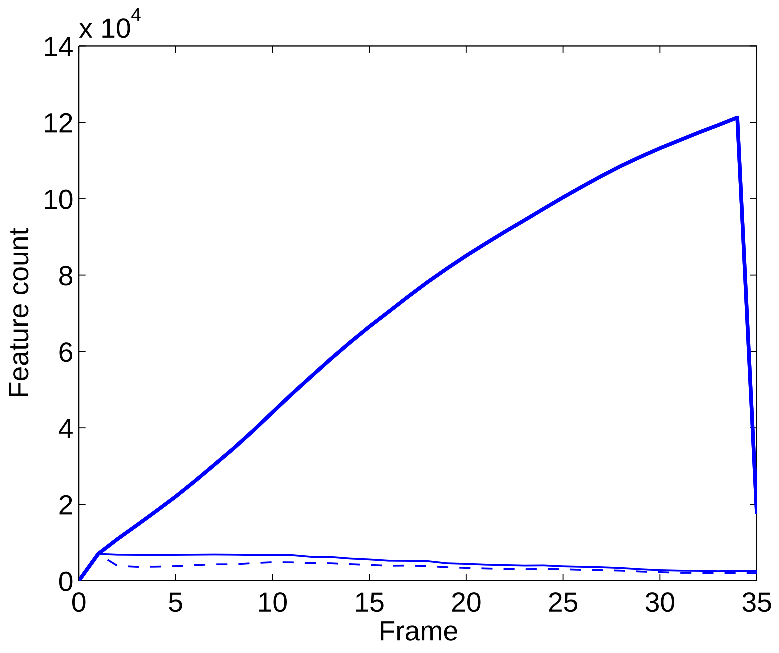


Figure 4.18: Feature counts for each frame in the *White* sequence: features per frame (thin line), new features (dashed line), total features kept (thick line).

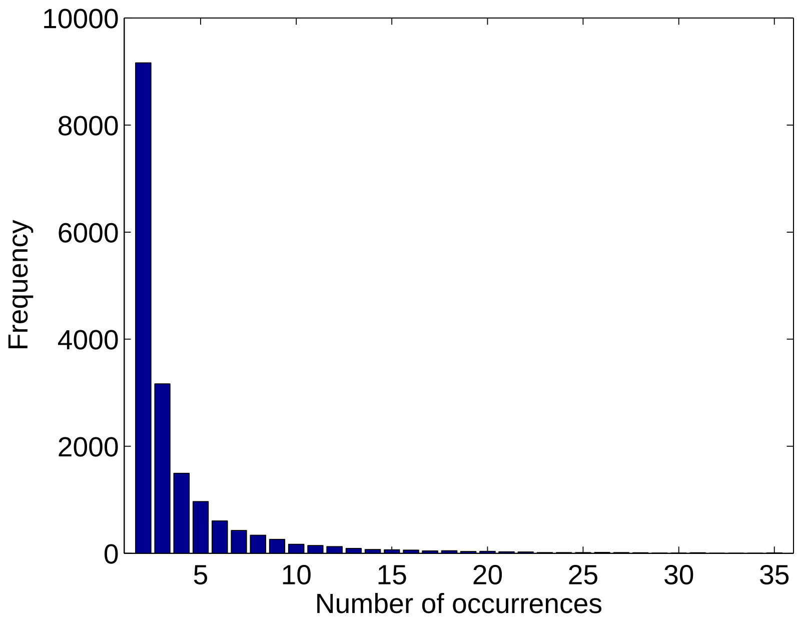


Figure 4.19: *White* sequence: number of features occuring in a given number of frames.

Number of occurrences	Frequency
(1)	(105730)
2	9164
3	3165
4	1494
5	966
6	605
7	426
8	337
9	261
10	167
11	146
12	124
13	91
14	69
15	65
16	59
17	45
18	48
19	32
20	35
21	26
22	24
23	12
24	14
25	12
26	16
27	14
28	9
29	4
30	5
31	10
32	2
33	2
34	2
35	6

Table 4.9: *White* sequence: number of features occuring in a given number of frames.

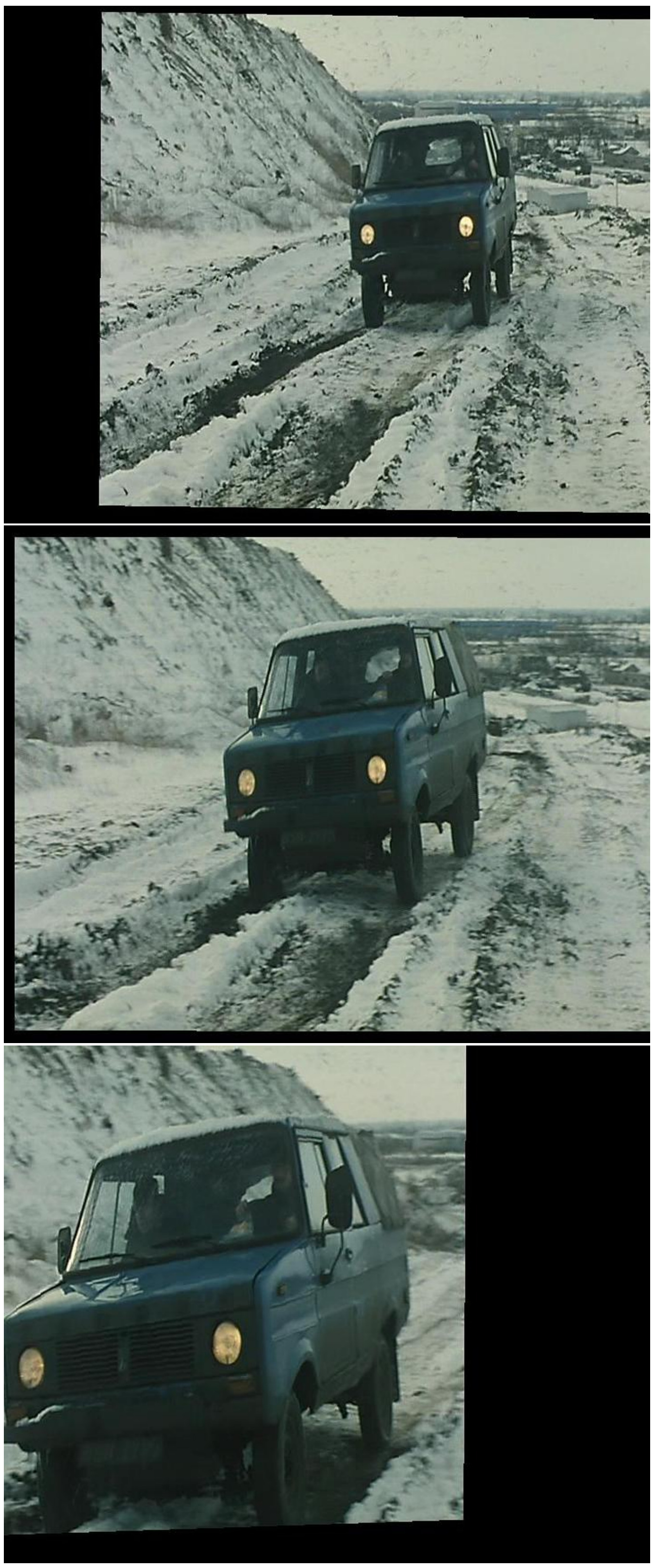


Figure 4.20: Example frames from the *White* sequence, stabilised with respect to the background.

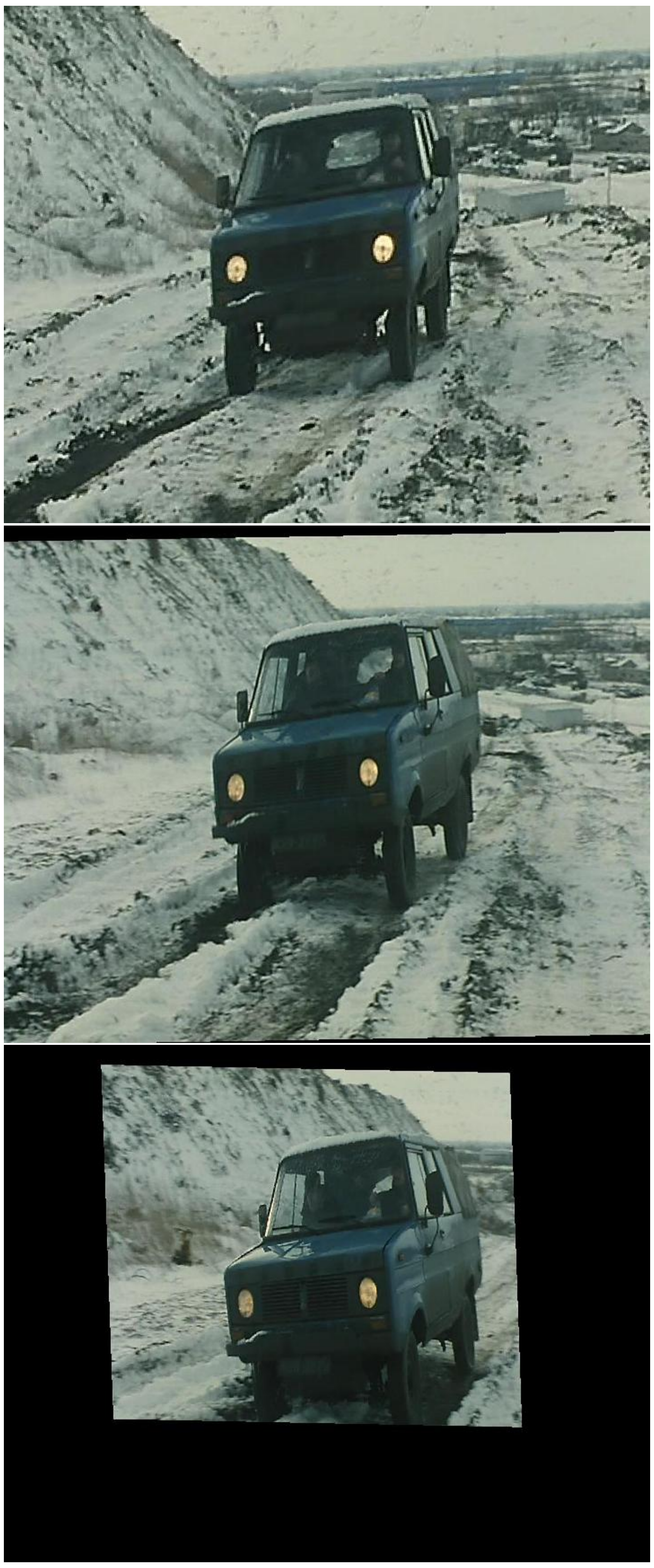
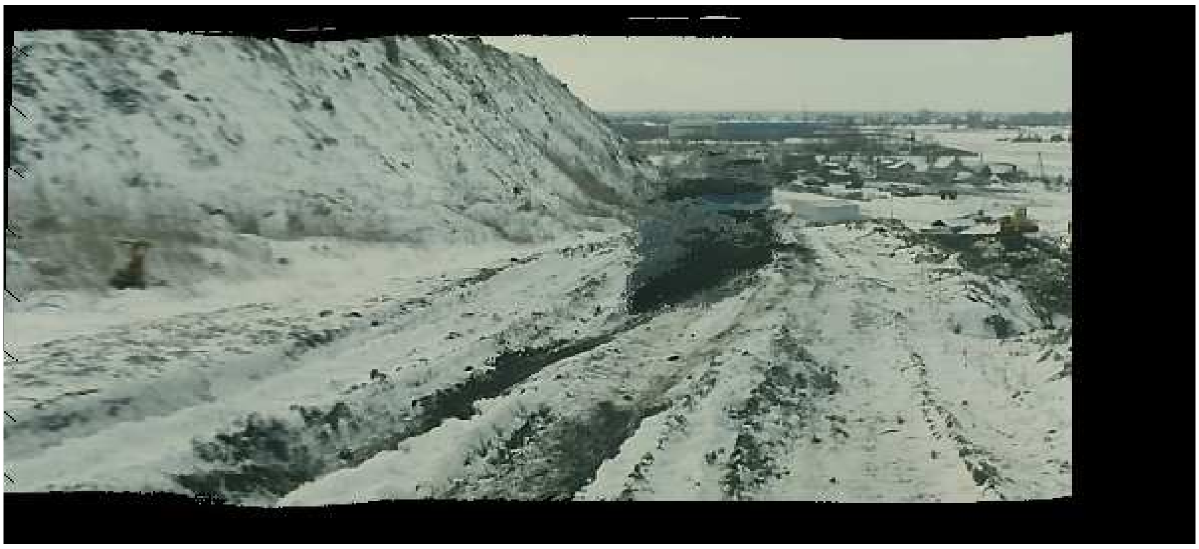
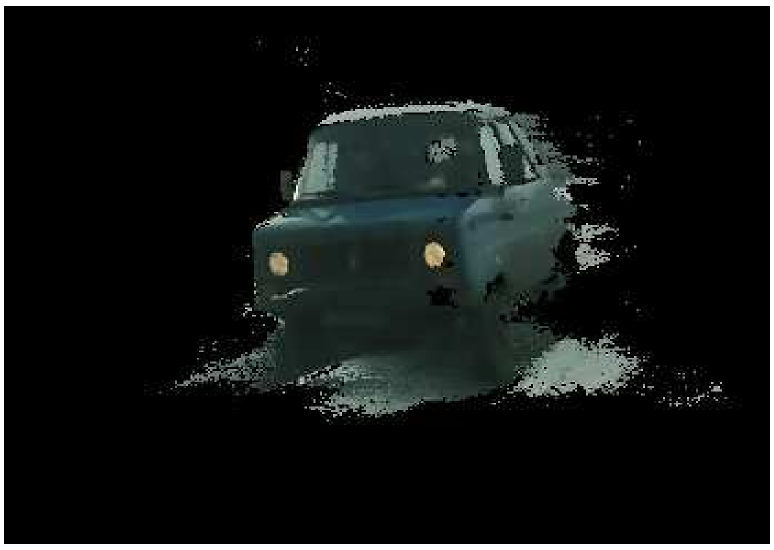


Figure 4.21: Example frames from the *White* sequence, stabilised with respect to the jeep.



(a)



(b)

Figure 4.22: Panel (a) displays the learnt background. Panel (b) shows the element-wise product of the thresholded mask and appearance model for the jeep.

example image sequences. The sequence RANSAC algorithm, using invariant features to discover motion groupings, makes the system much faster than methods that need to use extensive search to learn object motions.

Since it does not depend on matching objects between consecutive frames in an image sequence, the sequence RANSAC algorithm is robust to object occlusion, and works equally well even if we do not know the correct temporal ordering for a set of image frames, or if there are images taken simultaneously by multiple cameras from different viewpoints. However, for sequences where few features are detected on objects of interest, or where multiple objects present a similar appearance, motion learning performance could be improved by modelling frame-to-frame object motions across time, for example placing a Kalman filter on each object's sequence of affines.

The sprite model used above assumes that each object is rigid, and does not allow any significant change in two-dimensional appearance for a single object. It would be possible to allow more variation to deal with a wider range of objects. For example, Cootes et al. (1998) described a deformable sprite model to deal with a wider range of internal variation than is allowed by rigid sprites, demonstrating that it successfully modelled faces with different expressions and changes in pose. The strong geometric consistency requirement enforced by our sprite model means that an object with internal degrees of freedom might be incorrectly split up into multiple layers.

Another source of variation in object appearance is changes in viewpoint. The model above only learns a single appearance and mask for each object, but over a sequence, the appearance of even a rigid object can change markedly, as the object moves relative to a fixed camera, or as the camera moves. For example in the beginning of a sequence we may see a front view of a car, while at the end we see a side view. The layered generative model can be modified to allow the view of an object to vary significantly. We can achieve this by introducing a set of mask and appearance pairs, each one associated with a different viewpoint of the object, as proposed for example in Frey and Jojic (2003). Note also that this should not greatly increase the time taken, as the number of viewpoints needed is typically small. While there may be a continuous change in viewpoint through a sequence, we can cluster the observed viewpoints to give a small set of appearance models and masks.

The single parameter of the sequence RANSAC algorithm controls the tightness of the motion groupings extracted. In many image sequences, the main motion groupings extracted are insensitive to the parameter's value within a wide range: there is only a small variation in the position of features relative to the object to which they belong, while across a sequence there are extremely large differences between separate objects' motions. The sequence RANSAC algorithm could be made more robust for difficult image sequences by avoiding the need to set a fixed error threshold for learning object motions. Instead it might be possible to automatically determine an error threshold for each motion grouping based on the the distribution of features' motion variances relative to the motion model for the grouping under consideration.

Finally, we note that the extracted sprites could be improved by applying additional constraints. For example, an MRF prior over masks could be used to encourage coherent edges and discourage holes in the mask due to noise (following for example Blake et al., 2004), and object-specific colour models could enable the system to deal better with variation in object appearance under changing lighting conditions while still distinguishing objects with similar appearances but different colours.

Chapter 5

Object localisation using the Generative Template of Features

5.1 Introduction

This chapter describes a method for learning and recognising object classes, using invariant features. The goal is to create a system which can find instances of learnt object classes within new images, identifying the class of the object and giving its bounding box. Chapter 6 describes experiments using the object learning system described here, and discusses the results obtained. Some of the material in this chapter is adapted from Williams and Allan (2006).

Over the last few years there has been a surge of interest in the problem of object recognition for object classes. Current methods are better at recognising specific objects, such as a particular model of car, than at recognising classes of objects, such as the wider category of cars. Recognising classes of objects raises the same questions as recognising specific objects, but in addition requires generalisation from the training examples rather than simply accurate matching to training data.

We can distinguish between two related problems for object class recognition: classification, meaning detecting whether an object of the given class is present in an image, and localisation, meaning determining the position of an object in an image. We focus on using the Generative Template of Features (GTF) model for object localisation. We assume that each input image has been preprocessed by running a region-of-interest detector and matching the results to some set of ‘visual words’. The model, described fully in section 5.3 below, consists of a number of parts, with each part having a spatial location distribution and a distribution over visual words. Object localisation can be carried out by scanning the template over the image at a variety of positions and scales.

While we will use the GTF to model objects as sparse collections of image features, we can also view it as in a continuum with sprite models. At one end of the spectrum, dense sprite models explain images pixel by pixel. Moving along the spectrum we can view the pixel information as made up of a dense collage of local patches. These image patches can be compared by, for example, their normalised greyscale correlation, but we can also move towards more abstraction and use higher-level region descriptions with increased invariance. Finally, as in our experiments in chapter 6 below, we can choose to use a sparse representation, modelling only certain image regions that we expect to be especially informative.

Section 5.2 below gives a summary of related work on object class recognition. Section 5.3 describes the GTF model. Sections 5.4 and 5.5 looks at the relationship between the GTF and scanning window and pose-clustering methods respectively.

5.2 Literature review

This section discusses a variety of approaches to object class recognition using invariant features. We first discuss ‘bag of words’ approaches for image classification, in section 5.2.1. Then we look at three groups of methods for classification and localisation: scanning window methods in section 5.2.2, pose-clustering methods in section 5.2.3, and correspondence-based methods in section 5.2.4.

5.2.1 Classification without localisation

A variety of approaches have used image statistics without any explicit use of geometry to perform image classification. For example, Chapelle et al. (1999) use colour histograms to represent images, and perform classification using a support vector machine. The non-geometric approach that is most relevant to the model we describe below is the ‘bag of words’ approach that performs classification based on features extracted from local image regions.

The bag-of-words approach is named from its use in text classification. Rather than trying to parse the structure of a document, we can treat it as a set of unordered words, with a frequency count for each word. We can then compare documents based on their respective histograms of word frequency counts. Similarly, we can quantise local image features into 'visual word' groups, then compare images in a bag-of-words way based on the histograms of visual words that appear in them.

Dorko and Schmid (2005) use a bag-of-words approach with feature selection to choose the most informative parts. They use Harris-Laplace and Kadir-and-Brady region-of-interest detectors, and compute a SIFT descriptor for each region, then they cluster features from in-class images to learn a Gaussian mixture model for the class features. The trained Gaussian mixture model can be used as a classifier by imposing a separation boundary for each component. They reduce the size of their class models by ranking the Gaussian mixture model components based on their performance as individual classifiers, and choosing the best n components for the final model. They set a threshold number of the selected components that must give a positive classification for an overall image to be judged to belong to a given class.

Csurka et al. (2004) use a similar approach, with a Harris affine region-of-interest detector and SIFT descriptors. They use k-means rather than a Gaussian mixture model to create image feature clusters, and calculate a histogram of the number of activations of each cluster rather than observing only the presence or absence of features belonging to each cluster. They compare Naive Bayes and a support vector machine as overall classifiers, finding that the support vector machine gives better performance. Deselaers et al. (2005) also take a similar approach, using a discriminative log-linear classifier, which also outperforms baseline models including Naive Bayes.

Sivic et al. (2005) use shape adapted and maximally stable regions with SIFT descriptors, clustered using k-means clustering, as features for probabilistic Latent Semantic Analysis and Latent Dirichlet Allocation. They show that these models can discover image classes as 'topics' in unlabelled image collections, based on feature co-occurence statistics. Wang et al. (2006) use a more complex dependent hierarchical Dirichlet process, which models dependencies between image patch occurences, to learn a tree-structured taxonomy for image classes in an unsupervised manner. They achieve better classification performance from semi-supervised training with a small number of labelled images and visual words. They use Kadir-and-Brady regions of interest, SIFT descriptors, and k-means clustering.

These non-geometric classification methods can be extended to give estimates of object locations by, for example, identifying the image features which contribute to the classification decision and drawing a bounding box around them. To achieve good localisation performance, however, it is sensible to include some degree of geometric information in the model itself, like the methods we discuss in the following sections.

5.2.2 Scanning window methods

Scanning window methods run an object detector at different possible object scales and locations across an image, considering each object bounding box hypothesis and searching for maxima in the detector output. For example, Le Cun et al. (1990) used a scanning window approach with a neural network-based handwritten-digit detector trained using backpropagation. More recently scanning window approaches were used for example with a face detector using local image patches trained by boosting (Viola and Jones, 2004), and a car detector using clustered interest regions (Agarwal et al., 2004).

Torralba et al. (2004) follow the Viola and Jones boosting approach, but shares features across multiple classes. They show that this allows a given performance level to be achieved using fewer features than are required if independently-trained classifiers are used, while the features selected are more generic than those selected by independent training.

Shotton et al. (2006) use a conditional random field to learn an object classifier, using a mixture of feature types incorporating information about shape, texture, colour, location, and edges. They follow the approach of Torralba et al. (2004) to select discriminative features shared between classes. They use the conditional random field to give an approximate pixel-level labelling of the classes present in test images.

5.2.3 Pose-clustering methods

Pose-clustering methods (see Forsyth and Ponce (2003), §18.3) allow individual image features to vote on object instantiation parameters, then look for maxima in the summed voting space. For example, straight lines can be recognised by allowing noisy line segment features to vote for a range of line orientations passing through the feature's location, then looking for vote responses above some predetermined threshold (Hough, 1962). This approach can be generalised to detect arbitrary shapes (Ballard, 1981). Section 5.5 below looks at the relationship between the GTF and pose-clustering methods.

Using informative features like visual words can allow us to make votes based on quite specific predictions from individual features. For example, a single SIFT match allows us to predict an object's location, scale and orientation. Lowe (2001) matches images to models for individual objects using a Hough transform approach. Each SIFT feature in a test image is matched with its nearest neighbour in the database, and then votes for a model view along with an approximate location, scale and orientation for the hypothesised object. After voting, the winning hypothesis is verified by checking for overall geometry consistency between the image and the model, as judged by small residuals on a best-fit similarity transformation.

Leibe et al. (2004) take a similar approach to the case of object category recognition. They extract image patches at Harris interest points, cluster them using normalised greyscale correlation, and then vote based on the offsets within the object bounding boxes at which a given patch cluster was seen in the training data. Leibe and Schiele (2004) extend their method to deal with objects of varying scale. Here they use visual words created by agglomerative clustering of image patches extracted with radiuses proportional to the scales of the Harris-Laplace or difference-of-Gaussians interest regions in question. Fritz et al. (2005) add a support vector machine to verify whether the winning hypothesis is genuinely an instance of the class in question.

5.2.4 Correspondence-based methods

Correspondence-based methods (Grimson, 1990) match image features to features in a model. For example, the 'constellation model' (Burl et al., 1998) uses a number of object parts which are found in characteristic positions relative to each other, matching each part to a region in each image. For example, for faces we might think of having the eyes, nose, and mouth as parts. The constellation model learns the joint probability density on part locations.

Weber et al. (2000b,a) use the constellation model approach to model object classes. They first extract fixed size patches around interest points and cluster them using k-means to create candidate object parts, discarding small clusters. They then iteratively test the validation data classification performance of different combinations of small numbers of parts, starting from a random choice of parts and at each iteration testing the replacement of one part by a randomly selected one to greedily search for the most useful set of parts. Fergus et al. (2003) enhance the model to take into account variation in part appearance, and learn part appearances during expectation maximisation along with part location. They use the Kadir and Brady region of interest detector, giving circular regions of varying sizes. They also allow objects to vary in scale.

The constellation model can be slow to train, and at test time potentially requires a search over all possible correspondences between image features and object parts. As this is a combinatoric search it is exponentially slow unless aggressive pruning of the search tree can be achieved.

Helmer and Lowe (2004) improve the learning efficiency of the constellation model, making it practical to use up to 12 parts, rather than the 6 or 7 used by Fergus et al. (2003). They initialise their model with the feature which has the highest density around it in SIFT appearance space, then greedily add parts one-by-one to increase the likelihood of the data. To improve efficiency they suggest only considering adding parts from a sampled set of features, sampling features which are near to matched features or which come from images which have good matches to the existing model. They store the best match in each image for the parts currently in the model, so that when they consider adding a part to the model they only need to search for its own best matches and then calculate the change in the likelihood.

Fergus et al. (2005b) propose replacing the constellation model with a star model which has a lower complexity for learning and recognition, also making it practical to use up to 12 parts. Instead of learning the parts' relative locations as a joint Gaussian density, the parts' locations are learnt relative to a single landmark part, such that their positions are conditionally independent given the landmark. As well as being faster in learning and recognition, their star model generally gives improved performance over the constellation model. Felzenszwalb and Huttenlocher (2000) also proposed restricting the constellation model to a tree structure, to allow a more efficient algorithm to be used to find the best match for an image.

Crandall et al. (2005) test a more general class of structures for parts, k-fans. When $k = 1$

the k-fan gives a star graph, while when k is one less than the number of parts the k-fan has dependencies between all pairs of parts, like the constellation model. Learning with manually-chosen parts, they find that increasing spatial dependencies improve performance, but that for some classes the degree of improvement seen beyond a 1-fan model, or even a 0-fan model with independent part locations, is small, while the low-k model is much quicker in learning and recognition. Crandall and Huttenlocher (2006) show that the object parts and their relationships can be learnt in an unsupervised manner, without hand-labelling part locations in the training images.

5.3 Generative Template of Features

This section describes the Generative Template of Features model. We first discuss the main points of the model, then give some information on modelling choices we have made in modelling the background features (section 5.3.1), object scale (section 5.3.2), and mixing proportion (section 5.3.3). The partial deriviatives used to optimise object hypotheses by gradient ascent are given in appendix A.

We assume that a region-of-interest detector has been run on each image, and that a local image descriptor like Lowe's SIFT descriptor (Lowe, 2004) has been computed for each region of interest. We cluster the descriptors obtained from a set of training images to create a dictionary of 'visual words'. Thus for image m with N_m interest points we have pairs $(\mathbf{x}_{mi}, w_{mi})$, $i = 1, \ldots N_m$, where $\mathbf{x}_{mi}$ denotes the position of feature i in image m, and w_{mi} denotes the visual word to which it matches. X_m and W_m are matrices of all the feature positions and matching visual words for image m.

Consider an object which has pose variables θ. Here θ could denote for example the (x,y) position of the object in the image, position plus scale and rotation, or it could be more complex and include information on an object's internal degrees of freedom. Under the model defined in Sudderth et al. (2005) we have

$$p(\theta, X_m, W_m) = p(\theta) \prod_{i=1}^{N_m} p(\mathbf{x}_{mi}, w_{mi} | \theta). \tag{5.1}$$

Each feature $(\mathbf{x}_{mi}, w_{mi})$ is generated conditionally independently given θ.

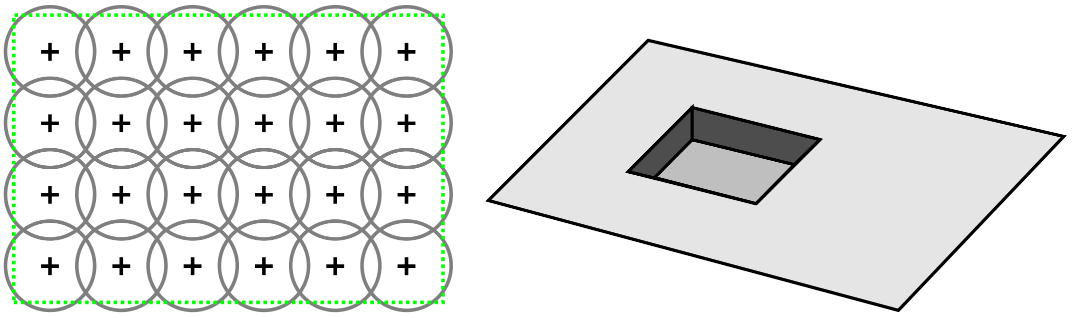


Figure 5.1: (a) GTF foreground model with grid of Gaussians; (b) background model.

Since image features may be generated either from background clutter or from an object of interest, we propose a mixture model

$$p(\mathbf{x}_{mi}, w_{mi}|\theta) = (1-\alpha)p_b(\mathbf{x}_{mi}, w_{mi}) + \alpha p_f(\mathbf{x}_{mi}, w_{mi}|\theta), \tag{5.2}$$

where p_b denotes the background model, p_f the foreground model for the object, and α is the probability of choosing to generate from the foreground. The background model may, for example, generate features anywhere in the image and with a broad distribution of visual word types, with $p_b(\mathbf{x}, w) = p_b(\mathbf{x})p_b(w)$. We use a more complex background model, which assigns lower background probability to the foreground area (Figure 5.1(b)), described in section 5.3.1 below.

The term $p_f(\mathbf{x}_{mi}, w_{mi}|\theta)$ can be further decomposed using the notion of parts. If z_{mi} is an indicator variable for each of the P possible parts in the object, then we have

$$p_f(\mathbf{x}_{mi}, w_{mi}|\theta) = \sum_{z_{mi}=1}^{P} p_f(\mathbf{x}_{mi}|z_{mi}, \theta)p(w_{mi}|z_{mi})p(z_{mi}). \tag{5.3}$$

This equation means that the location of feature mi, $\mathbf{x}_{mi}$, depends on the part from which it is generated and the object's instantiation parameters, while the visual word w_{mi} is generated from a multinomial distribution that depends on only the identity of the part that the visual word is generated from. Note that it is not absolutely necessary to cluster the descriptors into a discrete set of visual words; one could define $p(w|z)$ over real-valued descriptors, using for example a mixture of Gaussians, and possibly also make $p(w|z, \theta)$ vary with the object pose θ.

Each $p_f(\mathbf{x}_{mi}|z_{mi}, \theta)$ could be, for example, a Gaussian distribution. The foreground model that we use consists of a spatial grid of Gaussians (one for each part), each with an associated multinomial $p(w|z)$ in a canonical reference frame. Figure 5.1(a) shows a 6×4 GTF. In our implementation this model is transformed under θ by translation and separate x and y scalings to define $p_f(\mathbf{x}|z,\theta)$. A similar generative model was defined in Revow et al. (1996) for black (ink) pixels, without the visual words component.

For any set of images of a particular class of objects which has been normalised to a common reference frame (by translation, scaling, etc.), we expect to see regions which have propensities to generate particular visual words. For example, if we normalise a set of side views of cars, there will be regions towards the bottom left and right of the views which tend to generate visual words associated with wheels. While we define a spatial grid of object parts, it would also be possible to adapt part locations during GTF training.

Support for the hypothesis that there is one object of a given type O_j in the scene, plus background features, can be evaluated by computing

$$p(X_m, W_m|O_j) = \int p(\theta, X_m, W_m|O_j) d\theta \tag{5.4}$$

for each object model, including a pure background model O_0 to account for the case where there is no object of a known type present. In general the integral in equation 5.4 is not analytically tractable but if θ is low dimensional it can be approximated, for example by using numerical quadrature based on a grid of points in θ-space, or by using Laplace's approximation at a mode of $p(\theta|X_m, W_m, O_j)$. Localization of an object with respect to the θ parameters such as position and scale can be carried out with the GTF by scanning the template over the image at a dense grid of θ settings, and detecting maxima of $p(\theta|X_m, W_m, Oj)$ in this space, or alternatively by using a coarse grid in θ space with hill-climbing search.

Here we focus on the case where there is one foreground object. To deal with multiple objects, equation 5.2 can be extended to have multiple foreground mixing proportions, as in Sudderth et al. (2005). This approach ignores occlusion, but it would be quite straightforward to use a layered model and to reason about occlusion so as to generate only from visible components.

Unlike correspondence-based methods (see section 5.2.4), the GTF model does not enforce generation from each part. The conditional independence assumption in the generative model gives a non-zero probability that a part is not chosen on any of the draws from the foreground.

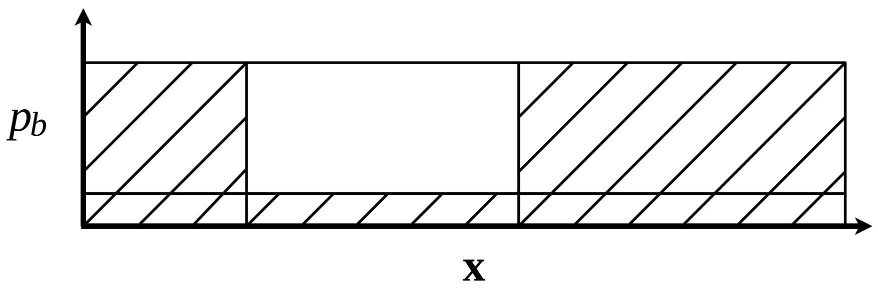


Figure 5.2: Background feature-location model: uniform across image plus uniform outside the object bounding box.

This is useful behaviour when part of an object is occluded, but it can also lead to incorrect detections. This problem was observed by Revow et al. (1996) where it was called the 'beads in white space' problem, as ink generators (beads) could occur in regions where there were no black (ink) pixels. One way to deal with this is to use the GTF to find promising regions of θ space, and then evaluate measures of model fit for instantiated models which can be fed to discriminatively-trained classifiers. This strategy was used by Revow et al. (1996) on the digit recognition problem, and more recently has been used, for example, by Fritz et al. (2005) for the recognition of cars, motorbikes, cows and horses.

Most scanning window methods (see section 5.2.2) use discriminatively-trained classifiers, but we can also use a scanning window approach with the GTF. Unlike discriminative classifiers, the GTF is capable of being trained in an unsupervised manner. Note that even if we scan an object hypothesis across an image, the GTF's likelihood term $p(X_m, W_m|\theta)$ still considers the probability of the 'background' features outside the bounding box, not just the features within the object hypothesis bounding box in question.

5.3.1 GTF background model

The background model used in our experiments below generates feature locations from a mixture, with probability β assigned to a uniform distribution across the image (in our experiments $\beta = 0.05$), and probability $1-\beta$ assigned to a distribution that generates approximately from a uniform distribution across locations in the image outside the object bounding box, as illustrated in figure 5.2.

If we have indicator functions $I_f(\mathbf{x})$, $I_b(\mathbf{x})$, $I(\mathbf{x})$ which are respectively one in the object bounding box, one in the background (outside the object bounding box), and one anywhere in the image, and which are zero elsewhere, we can declare a background feature-location distribution

$$\begin{aligned} p_b(\mathbf{x}) &= \frac{\beta}{A} I(\mathbf{x}) + \frac{1-\beta}{A-A_f} I_b(\mathbf{x}) \\ &= \frac{\beta}{A} I(\mathbf{x}) + \frac{1-\beta}{A-A_f} \left(I(\mathbf{x}) - I_f(\mathbf{x}) \right) \\ &= \left(\frac{\beta}{A} + \frac{1-\beta}{A-A_f} \right) I(\mathbf{x}) - \frac{1-\beta}{A-A_f} I_f(\mathbf{x}), \end{aligned} \tag{5.5}$$

where A is the area of the image and $A_f = s_1 s_2$ is the foreground area.

To give a differentiable function we approximate $I_f(\mathbf{x})/A_f$ using the GTF's foreground grid of Gaussians, such that

$$p_b(\mathbf{x}_{mi}, w_{mi}|\theta) = \left(\frac{\beta}{A} + \frac{1-\beta}{A-A_f} \right) p_b(w_{mi}) - \frac{1-\beta}{A-A_f} A_f p_h(\mathbf{x}_{mi}, w_{mi}|\theta), \tag{5.6}$$

where $p_h(\mathbf{x}_{mi}, w_{mi}|\theta)$ gives a 'hole' the same shape as the foreground:

$$p_h(\mathbf{x}_{mi}, w_{mi}|\theta) = \sum_{z_{mi}=1}^{P} p_f(\mathbf{x}_{mi}|z_{mi}, \theta) p_b(w_{mi}) p(z_{mi}). \tag{5.7}$$

5.3.2 Scale model

Objects of a given class tend to be seen at a particular range of sizes in an image data set. Photographs could be taken at any distance from an object, making the object any size in an image, but in practice the data sets we have used generally try to include some background scene in addition to the primary object, stopping it from filling the image, and tend not to include objects of interest that are very small in the images.

We define the GTF template as 1×1 pixels, and scale it by x and y scale factors s_1 and s_2 to fit each object bounding box. The grid of GTF component parts is also scaled by s_1 and s_2, so that the parts retain their positions relative to the template bounding box. The location variance with which each part generates feature locations is scaled similarly.

To model the probability of seeing an object bounding box of a given width s_1 and height s_2, we fit a Gaussian in log scale space:

$$p(s_1, s_2) = \frac{1}{2\pi\sqrt{|\mathbf{S}|}} \exp^{-\frac{1}{2}\left(\mathbf{S}_{11}(\log s_1 - \mu_1)^2 + \mathbf{S}_{22}(\log s_2 - \mu_2)^2 + 2\mathbf{S}_{12}(\log s_1 - \mu_1)(\log s_2 - \mu_2)\right)} \tag{5.8}$$

where μ and $\mathbf{S}$ are the mean and the inverse of the covariance matrix of the Gaussian.

We assume that the object centre is generated uniformly across the image.

5.3.3 Mixing proportion model

We learn a model for the mixing proportion α (see equation 5.2) from the training data, parameterised by the proportion of the image area covered by the foreground object. The model, learnt by linear regression, is of the form

$$\alpha = \gamma \frac{s_1 s_2}{A}, \tag{5.9}$$

where A is the area of the image.

5.4 Relation to scanning window methods

The approaches of Fergus et al. (2005a) and Sudderth et al. (2005) are closely related to the GTF model described above.

The translation and scale invariant probabilistic Latent Semantic Analysis model used in Fergus et al. (2005a) is similar to our grid of Gaussians model, except that it has the features associated with each part generated within hard ‘cells’ (or box basis functions) rather than from overlapping Gaussians, and is applied in an unsupervised learning context. Fergus et al. (2005a) concentrate on object categorisation; the average precision scores they report for object localisation (their Table 3) are quite poor.

The model in Sudderth et al. (2005) does use a mixture of Gaussians. They learn general spatial offsets for the parts rather than using a grid, though note that a sufficiently fine grid of parts can approximate the effect of any learnt part distribution. Their focus is on learning parts which can be shared across object categories such as various kinds of animal. Our system can learn to generate the same visual words for multiple classes, but they also use a Dirichlet process to share part visual word distributions across multiple classes.

Fergus et al. (2005a) and Sudderth et al. (2005) carry out training using unsupervised learning, but if we have training data for each object class annotated with bounding boxes, as in the PASCAL Visual Object Classes challenge data set[1] dataset, then supervised learning can be

[1] http://www.pascal-network.org/challenges/VOC/voc2005/

used, as described in section 6.3 below. Our experiments in Chapter 6 focus on examining the object category localisation performance that can be achieved by a GTF trained in a supervised manner.

5.5 Relation to making predictions in pose-space

To consider different possible object pose parameters in the localisation task, we have to compute $p(\theta|X_m, W_m)$. Taking logs of equation 5.1 we obtain

$$\log p(\theta, X_m, W_m) = \log p(\theta) + \sum_{i=1}^{N_m} \log p(\mathbf{x}_{mi}, w_{mi}|\theta). \tag{5.10}$$

As the data (X_m, W_m) are fixed we have $p(\theta, X_m, W_m) \propto p(\theta|X_m, W_m)$, with $p(\mathbf{x}_{mi}, w_{mi}|\theta)$ viewed as a function of θ. Thus the generative model can be used to hypothesize detections in θ-space by finding the maxima of $p(\theta|X_m, W_m)$, for example by hill-climbing. Such an explanation of the probabilistic Hough transform can be found, for example, in Stephens (1991), although without the use of specific visual word features, which provide more information and thus tighter distributions.

To spell this out further, consider a distinctive visual word which occurs in only one position on an object. This feature will be predictive of the location of the centre of the object, but as it can also be generated from the background part there is also an associated broad outlier distribution as derived from equation 5.3.

Equation 5.10 shows how to run the generative model backwards to provide predictions in parameter space. However, given training data with features $\{(\mathbf{x}_{mi}, w_{mi})\}$ it is natural to build predictors for $p(\theta|\mathbf{x}_{mi}, w_{mi})$, for example by creating a Parzen windows estimator for $p(\theta|\mathbf{x}_{mi}, w_{mi})$. How should we then combine these predictions from each feature in order to obtain $p(\theta|X_m, W_m)$? Fortunately Bayes' rule comes to our aid, as

$$p(\mathbf{x}_{mi}, w_{mi}|\theta) = \frac{p(\theta|\mathbf{x}_{mi}, w_{mi}) p(\mathbf{x}_{mi}|w_{mi}) p(w_{mi})}{p(\theta)}. \tag{5.11}$$

Here $p(\theta|\mathbf{x}_{mi}, w_{mi})$ is obtained from the predictive model, $p(w_{mi})$ is just the marginal probability of visual word w_{mi} over the training set, and $p(\mathbf{x}_{mi}|w_{mi})$ is the probability of seeing a visual word of type w_{mi} in position $\mathbf{x}_i$. This could be estimated using a density estimator for the location of features of a given type in the collection of training data. Alternatively, if $p(\theta)$ has a non-informative location component, then we might expect that $p(\mathbf{x}_{mi}|w_{mi})$ should be uniform across locations in the image. This use of Bayes' theorem to replace likelihood terms with predictive distributions has been called the *scaled likelihood* method, see for example Morgan and Bourlard (1995).

Putting equations 5.10 and 5.11 together we obtain

$$\log p(\theta|X_m, W_m) = \sum_{i=1}^{N_m} \log p(\theta|\mathbf{x}_{mi}, w_{mi}) - (N_m - 1)\log p(\theta) + c, \tag{5.12}$$

where c is a constant independent of θ. Thus we have shown rigorously how to obtain $p(\theta|X_m, W_m)$ from individual predictions $p(\theta|\mathbf{x}_{mi}, w_{mi})$ up to a normalization constant. Note, however, that to compute the marginal likelihood (equation 5.4) from equation 5.12 additional terms involving $p(\mathbf{x}_{mi}|w_{mi})$ and $p(w_{mi})$ must be included.

Recently, Leibe et al. (2004) have used such ideas to predict an object's location based on the observed position of visual words. However, we note that the equation they use (their equation 6), is, in our notation,

$$\text{score}_m(\theta) = \sum_{i=1}^{N_m} p_f(\theta|\mathbf{x}_{mi}, w_{mi}). \tag{5.13}$$

Equation 5.13 does not at first sight agree with equation 5.1: for a start it sums probabilities rather than multiplying probabilities or summing log probabilities. However, using equation 5.2 we have

$$\begin{aligned}\prod_{i=1}^{N_m} p(\mathbf{x}_{mi}, w_{mi}|\theta) = &\prod_{i=1}^{N_m} p_b(\mathbf{x}_{mi}, w_{mi}) \times \qquad (5.14)\\ &\left[(1-\alpha)^{N_m} + \alpha(1-\alpha)^{N_m-1}\sum_{i=1}^{N_m}\frac{p_f(\mathbf{x}_{mi}, w_{mi}|\theta)}{p_b(\mathbf{x}_{mi}, w_{mi})} + O(\alpha^2)\right].\end{aligned}$$

If α is small and $p(\theta)$ is non-informative w.r.t. location then using equation 5.11 for $p_f(\mathbf{x}_{mi}, w_{mi}|\theta)$ we obtain to first order

$$p(\theta|X_m, W_m) = c_0 + c_1 \sum_{i=1}^{N_m} \frac{p_f(\mathbf{x}_{mi}, w_{mi})}{p_b(\mathbf{x}_{mi}, w_{mi})} p_f(\theta|\mathbf{x}_{mi}, w_{mi}), \tag{5.15}$$

where c_0 and c_1 depend on the image features but not on θ, and $p_f(\mathbf{x}_{mi}, w_{mi}) = \int p_f(\mathbf{x}_{mi}, w_{mi}|\theta)p(\theta)d\theta$. Minka (2003) has also discussed how a robustified product of probabilities gives rise to a sum of probabilities to first order.

Furthermore, if $p(\theta)$ has a non-informative location component then the spatial part of $p_f(\mathbf{x}_{mi}, w_{mi})$ will be non-informative and we can refine equation 5.15 to obtain

$$p(\theta|X_m, W_m) = c_0 + c_2 \sum_{i=1}^{N_m} \frac{p_f(w_{mi})}{p_b(w_{mi})} p_f(\theta|\mathbf{x}_{mi}, w_{mi}), \tag{5.16}$$

where $p_f(w) = \sum_{z=1}^{P} p(w|z)p(z)$, the weighted average of the multinomial vectors in the foreground parts. Equation 5.16 is close to equation 5.13, though note the weighting of each predictive distribution $p_f(\theta|\mathbf{x}_{mi}, w_{mi})$ by the factor $p_f(w_{mi})/p_b(w_{mi})$. If visual word w_{mi} is more probable under the background model then its prediction will be discounted. We note that Dorko and Schmid (2005) have discussed selecting discriminative foreground features for use in equation 5.13, but that their criterion is based on intuitive arguments rather than on a formal derivation.

5.6 Summary

This chapter described the Generative Template of Features model for object class recognition, and looked at related work, including a discussion of the relationship between the GTF and pose-clustering methods. The next chapter describes some object class recognition experiments using the GTF.

Chapter 6

Object localisation: implementation and results

6.1 Introduction

This chapter describes some experimental results on learning and recognising object classes, using invariant features, with the Generative Template of Features described in chapter 5.

In the experiments in this chapter we use the data from the PASCAL 2005 Visual Object Classes challenge[1] (Everingham et al., 2005). The data set consists of a large set of images, each of which contains at least one labelled object against cluttered backgrounds of many unlabelled objects. The labelled objects belong to four categories: bicycles, cars, motorbikes, and people. In the first set of experiments we use the 'train' and 'val' data sets as training and test sets respectively to see how the GTF's performance varies with different parameter choices, while in the second set of experiments we use 'train' and 'val' combined as a training set, and the 'test1' data set as test data. Table 6.1 shows the number of training and test examples for each class.

Note that the PASCAL data set has different properties from many other data sets used in image classification tasks, such as the 'Caltech 5' data: there may be multiple objects in each image, and there is a high degree of background clutter.

The task is to detect objects of the four categories in test images: each detection should

[1] http://www.pascal-network.org/challenges/VOC/voc2005/

	Bicycle	Car	Motorbike	Person
Training	57 (63)	136 (161)	107 (109)	42 (81)
Validation	57 (60)	136 (163)	107 (108)	42 (71)
Test	114 (123)	275 (341)	216 (220)	84 (149)

Table 6.1: Number of images (and objects) for each object class in the PASCAL VOC 2005 data.

state the type of the object, as well as its position in the image and the width and height of its bounding box. A detection is accepted as correct if the intersection between the prediction and true object covers at least half the area of a bounding box drawn to enclose both, as in Everingham et al. (2005). Each detection must be assigned a confidence value. The PASCAL challenge uses two evaluation measures to compare object detection systems: localisation performance is measured by average precision, while image classification performance is measured by the area under the receiver-operating-characteristics curve. The GTF is primarily an object localisation system, but by assigning confidences to the detections it makes we can also use it as a classifier.

Section 6.2 below describes the image features we use, explains how we cluster the features to learn visual words, and looks at an example clustering. Section 6.3 gives some details about the GTF implementation we have used in this chapter. Section 6.4 examines how our GTF implementation's performance varies with the choice of parameters, using the PASCAL Visual Object Classes Challenge 2005 training and validation data. Section 6.5 looks in more detail at the GTF's performance on the challenge test data.

6.2 Learning visual words

This section describes the features we extract from the data set of images, explains how we cluster the extracted features to learn visual words, and looks at the clusters learnt in an example clustering.

6.2.1 Features

We preprocess the data by scaling down larger images to fit within a 640×640 pixel square, preserving their aspect ratios, and then use two interest point detectors to find the Harris affine and maximally stable extremal regions of interest, described in sections 2.3.1 and 2.3.3 above.[2]

For each image we run the two region detectors, combine the lists of regions which they find, then calculate a descriptor for each region. The detected regions can optionally have a scaling applied before descriptor calculation. We use a 128-dimensional SIFT descriptor to represent each region's appearance, as described in section 2.4 above.

Figure 6.1 shows as an example how the number of regions detected within car bounding boxes varies with the object size. It can be seen that more regions are generally detected for bigger objects, but for some objects rather fewer regions are detected than the general trend might suggest.

Figure 6.2 shows how the number of regions detected outside all objects in images containing cars varies with the area of the image outside all objects. The sharp cut-off in the background area is due to the down-scaling of images larger than 640×640. It can be seen that very few features are detected in some large image background areas. Some images contain large background areas without the kind of texture which triggers the region detectors we are using.

6.2.2 Clustering

To create 'visual words' we cluster features from training images. We take separately the features found within the bounding boxes of each object type, and the features found in the background of images outside all object bounding boxes, running k-means clustering on the descriptors for each set of features. With four object classes, we run five separate clusterings, one for each class and one for background features, then finally combine the five sets of cluster centres. For example, if we use k-means clustering to find 120 cluster centres for each class, we then combine these to obtain an overall clustering with 600 cluster centres.

Each object's k-means clustering was run 12 times, with the cluster centres which gave

[2]We thank the Oxford Visual Geometry Group for making their feature detector code available at `http://www.robots.ox.ac.uk/~vgg/research/affine/`.

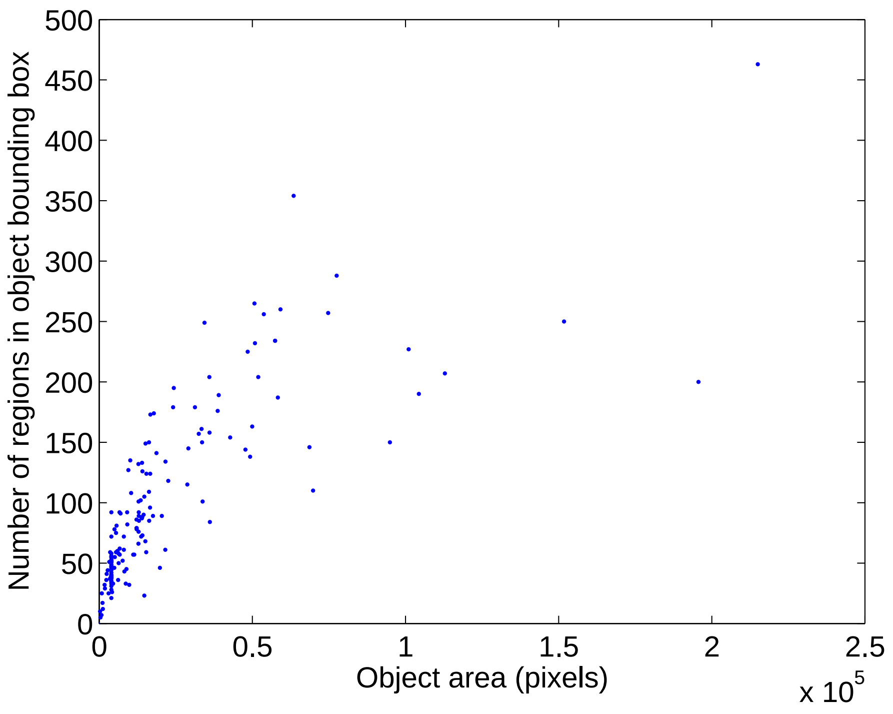


Figure 6.1: Number of regions detected within car bounding boxes.

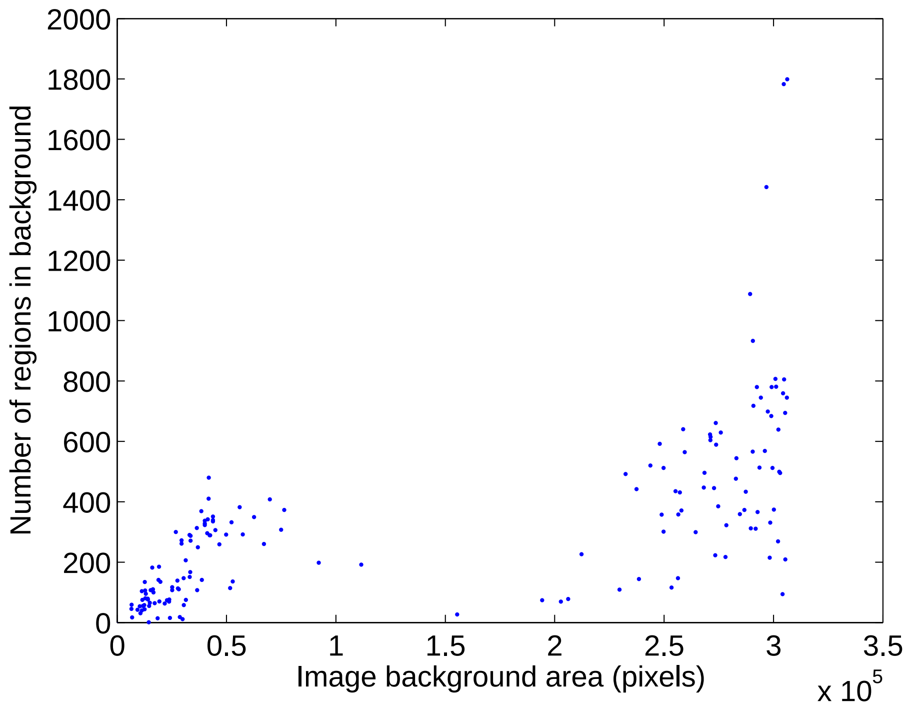


Figure 6.2: Number of regions detected in the background of images containing cars.

the lowest mean descriptor-to-centre distances chosen to go into the final combined clustering. On each run the cluster centres were initialised to a different randomly-chosen set of k feature descriptors.

Figure 6.3 shows an example plot of the mean distance from each feature descriptor to its nearest cluster centre against the number of iterations of k-means clustering. 12 runs are shown, with only small variations in descriptor-to-centre distances between the runs. Figure 6.4 shows the distance from each feature descriptor to its nearest cluster centre for one of these runs.

Figure 6.5 shows the foreground feature count per cluster within car training set bounding boxes, the background feature count per cluster in training set images where cars occur, and the foreground count as a proportion of the sum of the two counts. This proportion is the ‘purity’ of the cluster, describing how strongly cluster membership identifies a feature as foreground rather than background, and thus how useful it is in object localisation. Since the cars are the second object class, clusters 81–160 of the 400 are those derived from k-means clustering of the car foreground features. These clusters have the highest counts in the foreground, and the highest car purity, while clusters 321-400, which were derived from clustering background features, have the highest counts in the background, and lowest purity.

Figure 6.6 shows a representative image for each cluster of a combined clustering with 80 k-means clusters per class, giving a total of 400 clusters. For each cluster, the image region whose SIFT descriptor is nearest the cluster centre is displayed, rotated and scaled from the original image region to a fixed-size square image. The images are shown sorted by the clusters’ purity for the car class, left-to-right, top-to-bottom.

Figure 6.7 shows the clusters in the same order, choosing the representative images from car foreground regions only. By comparing these two figures it can be seen how some car features are matched to less specific clusters.

Using the same clustering, Figure 6.8 shows the 12 highest purity clusters for the car class. The features from the training set cars which match to each cluster are shown, with each region represented by a scaled image of the region contents displayed at the region centre’s position in a normalised object bounding box. Normalising the object bounding boxes to a unit square brings the objects into approximate correspondence, although the objects vary in shape, their pose is not fully labelled, and the bounding boxes sometimes exclude parts of objects that are

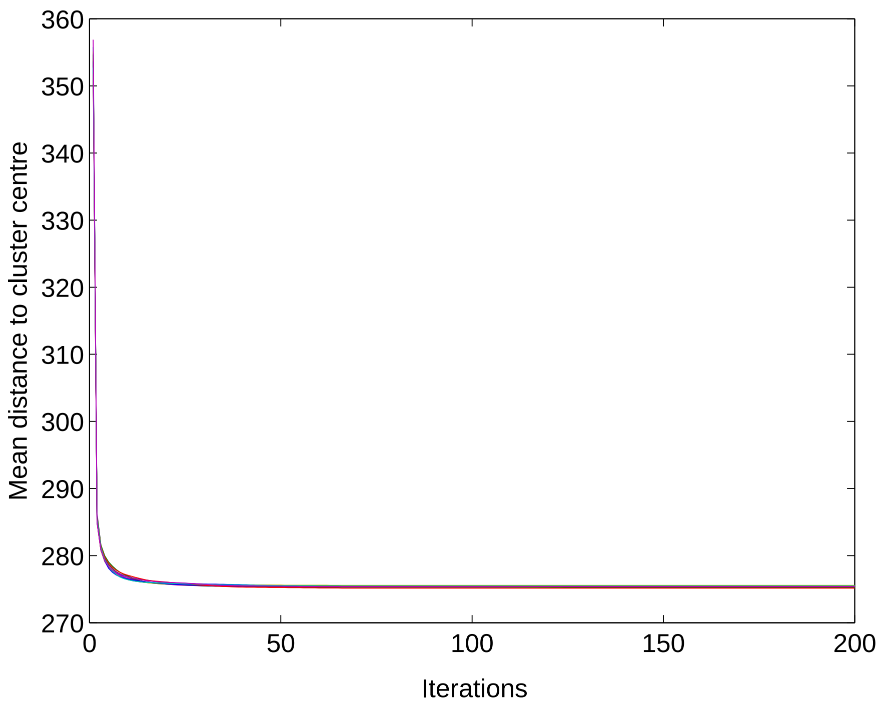


Figure 6.3: Mean distance from feature descriptors to their nearest cluster centre during k-means clustering, for an example clustering.

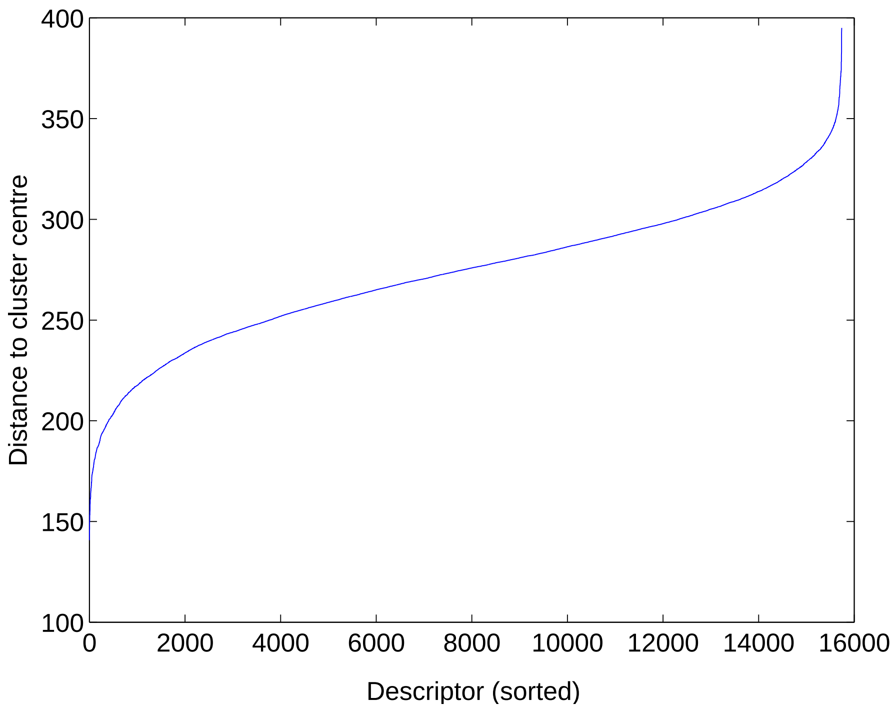


Figure 6.4: Distance from each feature descriptor to its nearest cluster centre, for an example clustering.

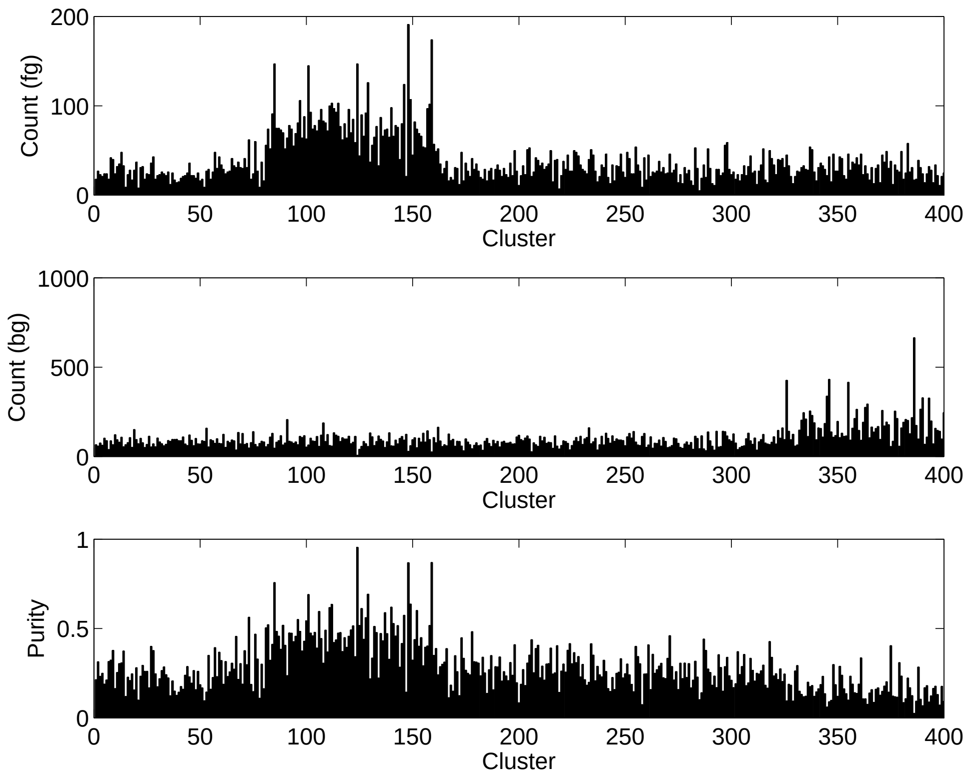


Figure 6.5: Cluster purity for car features in example clustering.

Figure 6.6: Representative image for each cluster in an example clustering, with clusters sorted left-to-right, top-to-bottom by purity for cars.

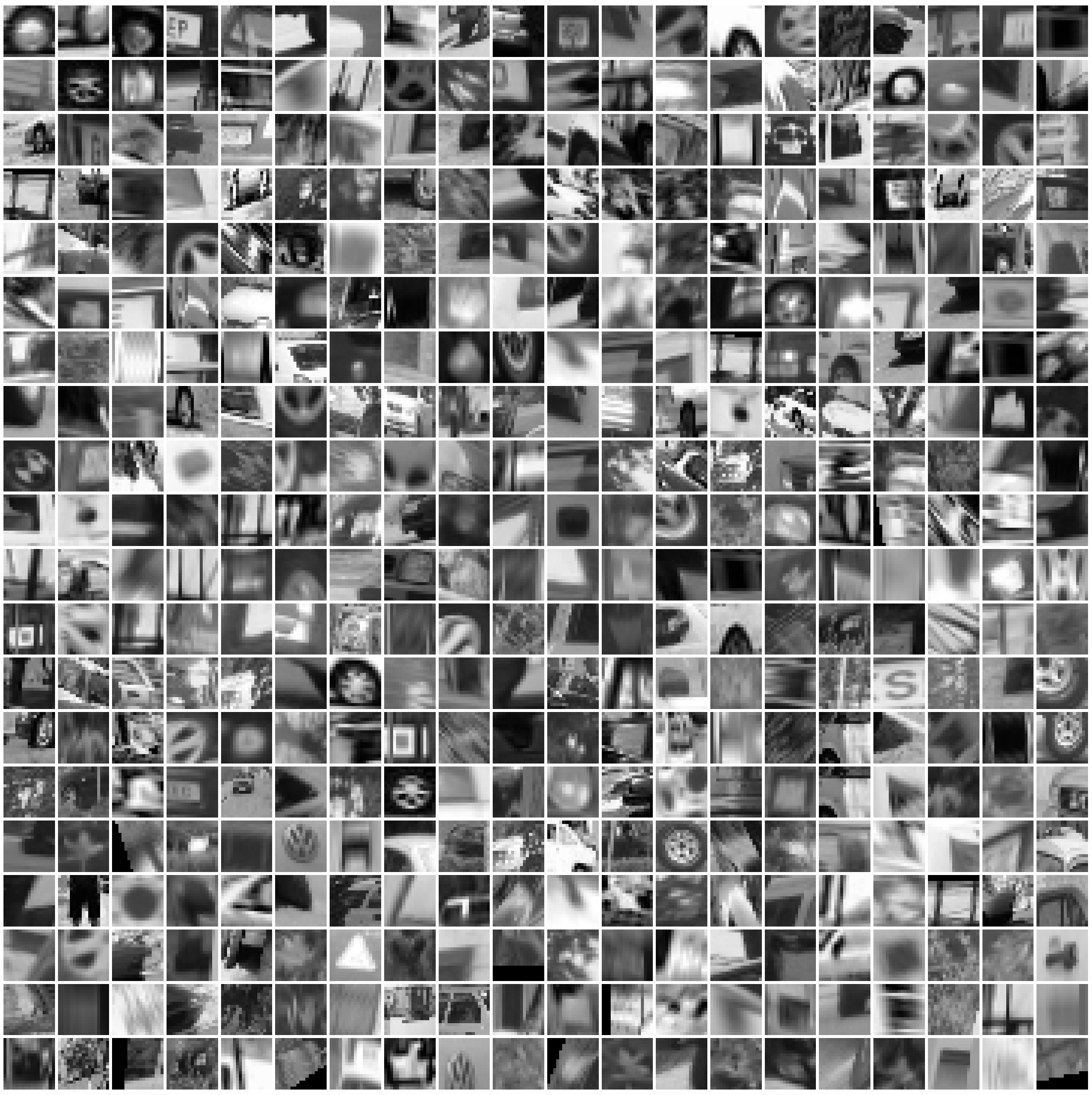


Figure 6.7: Representative image from car foreground for each cluster in an example clustering, with clusters sorted left-to-right, top-to-bottom by purity for cars.

occluded or extend outside the image.

The highest-purity clusters largely correspond to wheel-like features, which are rarely found in the background. These features tend to occur towards the bottom left and bottom right of the bounding box. Note that the multi-scale character of the features in use means that some features describe large regions of objects, such as a wheel in context with the car bodywork, or even an entire car. Features belonging to the same cluster may look dissimilar, since cluster membership is based on distance in SIFT descriptor space rather than on direct comparisons between image patch appearances.

As a comparison Figure 6.9 shows the lowest-purity clusters for the car class. These clusters match the car foreground fewer times than the high-purity clusters, and do not correspond to readily-identifiable parts of cars.

6.3 GTF implementation

This section gives some details about the GTF implementation used in the experiments later in this chapter.

We used a GTF with a grid of 8 by 8 Gaussian components for the GTF parts. x and y scale factors s_1 and s_2 are used to bring the template into registration with objects in training images, and to fit it to object instantiation hypotheses in test images. For any given object centre and scale factors we can translate and scale the template and its component Gaussians to calculate $p(\theta, X_m, W_m)$, where $\theta = (t_1, t_2, s_1, s_2)$.

To search for θ to optimize $p(X_m, W_m|\theta)$, we initially carry out a grid search on a coarse grid of positions at a number of scales, and then use hill-climbing to refine θ. The scales for grid search are chosen based on the range of scales seen in the training data, with a factor of $\sqrt{2}$ between each scale. Figure 6.10 shows the scales of the training example cars (blue dots), and the scales chosen for the grid search (green circles). The grid for the search at each selected scale is given by tiling the image with object hypotheses of that scale. We then use the maximum probability object centres found at the various scales as initialisations for gradient ascent. Expectation maximisation could also be used here. We perform gradient ascent in a four dimensional space, searching on location (t_1 and t_2) and scale (s_1 and s_2). After finding a

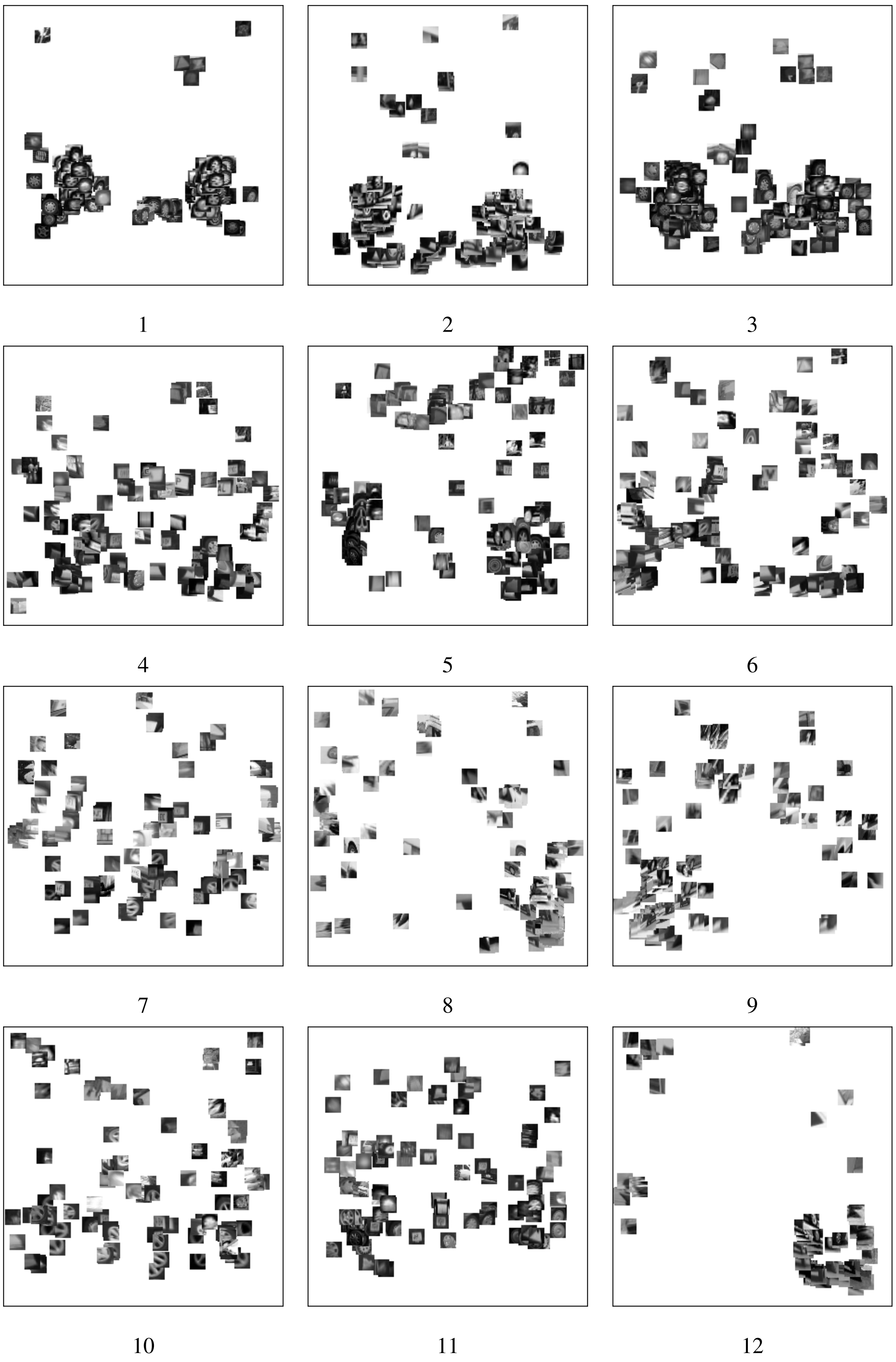


Figure 6.8: Highest-purity car clusters for an example clustering.

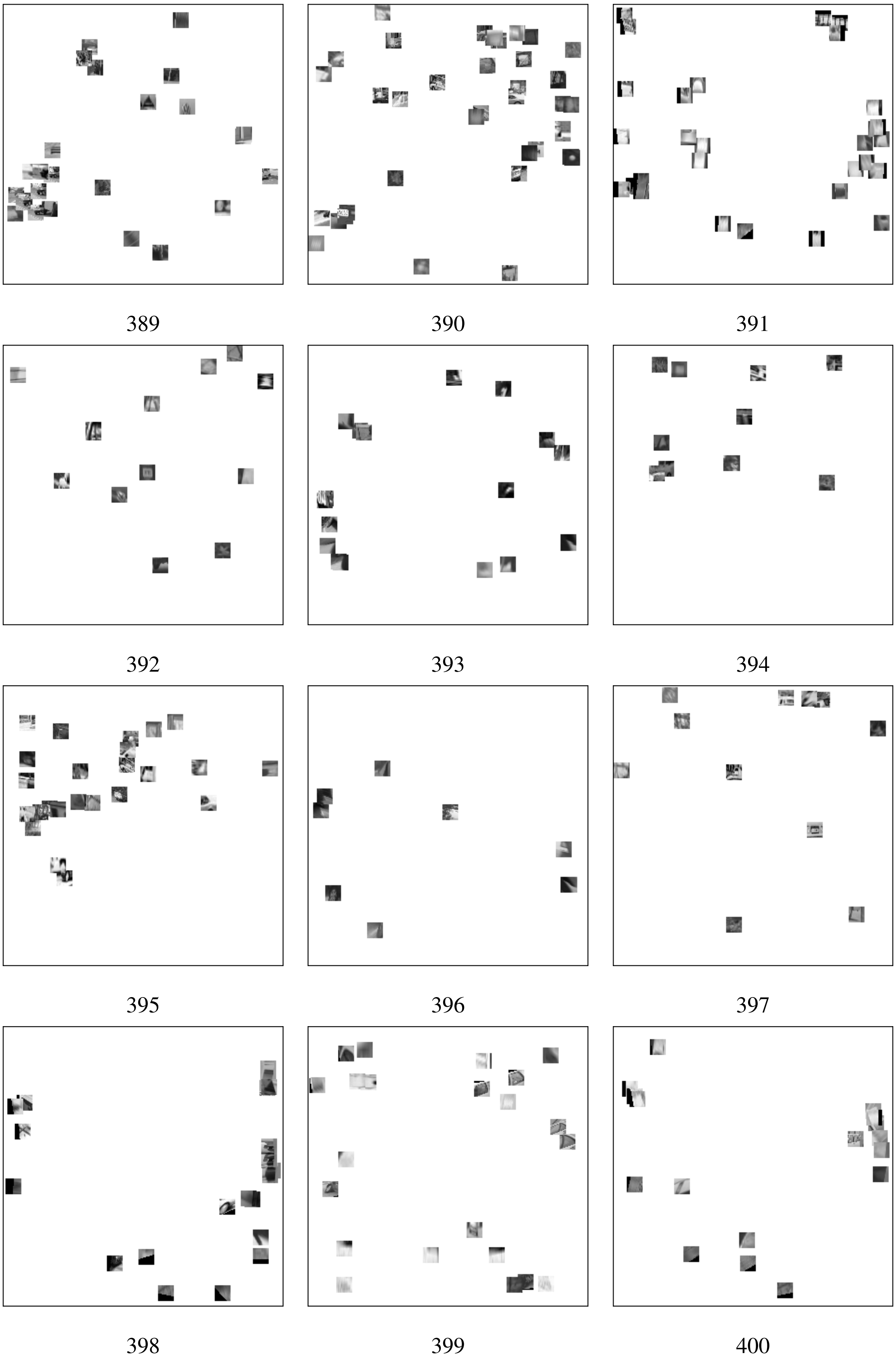


Figure 6.9: Lowest-purity car clusters for an example clustering.

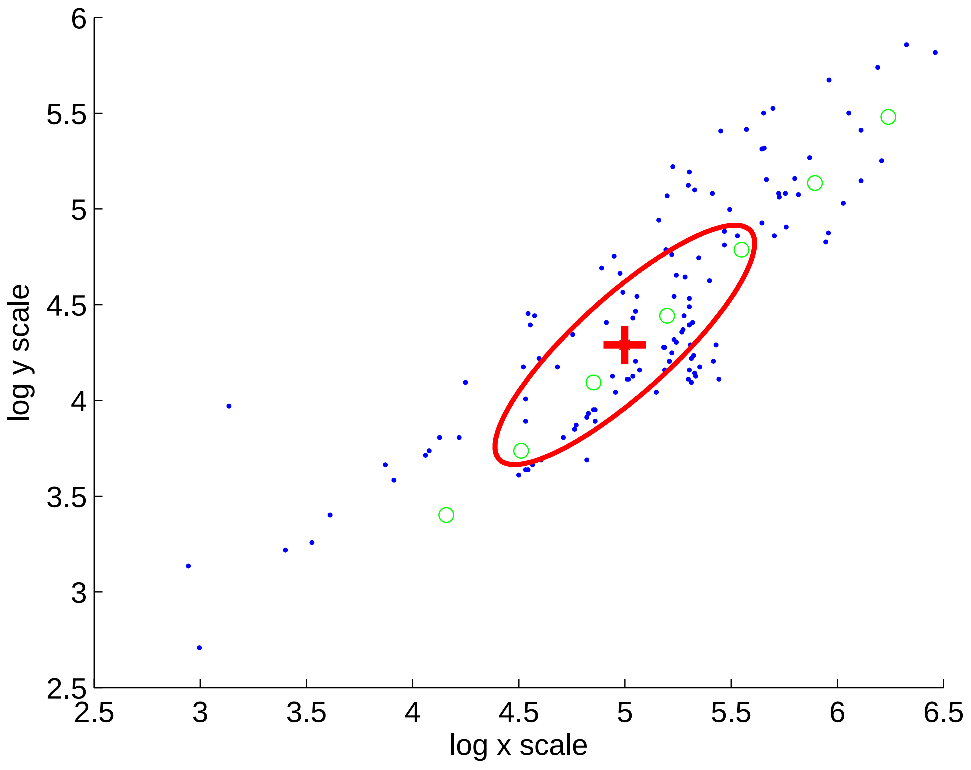


Figure 6.10: Bounding box sizes for training data cars.

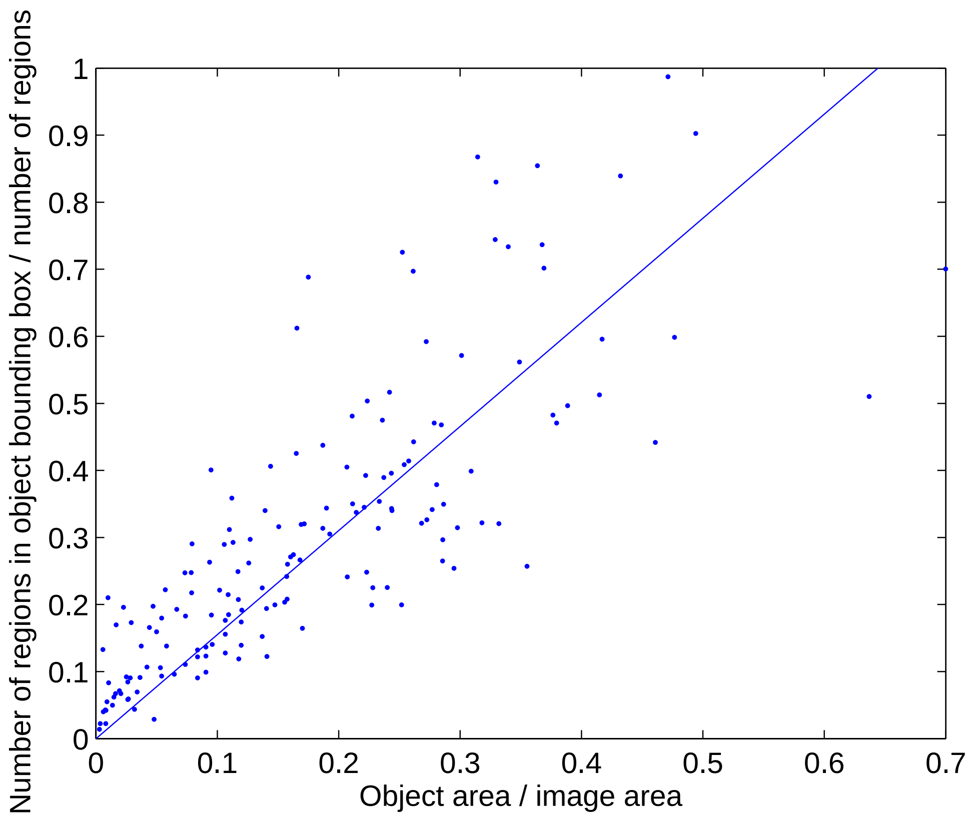


Figure 6.11: Foreground mixing proportion model for training data cars.

local maximum from each scale, we use the Laplace approximation to compare the probability mass in each corresponding region, and choose the maximum corresponding to the region with highest mass as the best detection for the image. The partial derivatives used in gradient ascent and in the Laplace approximation are given in appendix A.

In general learning the GTF requires estimation of the distributions $p(z)$, $p(\mathbf{x}|z)$ and $p(w|z)$ for each part. However, we use a spatial grid of Gaussians for $p(\mathbf{x}|z)$, so this part is fixed. Given training images for each object class annotated with bounding boxes we can use supervised learning to estimate $p(w|z)$. Each bounding box for a given object class is rescaled so as to be centered and have the same area as the template. (We use separate x and y scaling factors, so the rectangular bounding boxes can be brought into perfect alignment.) Given these aligned data it is straightforward to learn the parameters of the template by EM. Since we keep the background model's uniform distribution mixing proportion, β, small (see section 5.3.1), we can learn the foreground and background visual word distributions separately, using training features from only inside or only outside bounding boxes appropriately, making training much faster. $p(w|z)$ is found by the following update equation:

$$p(w = a|z = j) \leftarrow \frac{\sum_{m=1}^{M} \sum_{i=1}^{N_m} p(z_{mi} = j|\mathbf{x}_{mi}, w_{mi})\delta(w_{mi} = a)}{\sum_{m=1}^{M} \sum_{i=1}^{N_m} p(z_{mi} = j|\mathbf{x}_{mi}, w_{mi})}. \quad (6.1)$$

Figure 6.12 illustrates a trained 600-cluster 8×8 GTF for each object class, by showing the three visual words most strongly associated with each component part (highest $p(z|w)$; sorting by $p(w|z)$ favours frequently-seen uninformative visual words). A representative image is chosen for each visual word from the foreground for the object class in question. The bicycle GTF shows a variety of wheel features all over the template, as bicycles are seen in many different poses in the training images, from diverse viewpoints. For the car GTF various wheel features can be seen at the bottom left and bottom right of the template. Wheel and handlebar features can be seen on the motorbike GTF. The person GTF shows some face features near the top of the template, and foot features at the bottom.

As described in section 5.3.3, the foreground mixing proportion α is set according to what proportion of the image is filled by the object. Figure 6.11 plots the the the number of regions in the object bounding box as a proportion of the total number of regions in the image (the true mixing proportion) against the object area as a proportion of the total image area, and shows the line $\alpha = \gamma \frac{s_1 s_2}{a_e}$.

As described in section 5.3.2, the probability of a scale (s_1, s_2) for $p(\theta)$ is given by a Gaussian fitted to the object scales seen for the relevant class in the training data. The red ellipse in Figure 6.10 shows this Gaussian scale model learnt from the car training examples. There is a uniform prior over translations within the image.

To generate precision-recall curves we need to assign a confidence to each hypothesis. We set this confidence value based on the ratio between the probability of the hypothesis under the fitted GTF model and its probability under a GTF where the foreground and background components share the same ‘background’ visual word distribution. Making this comparison between the probability under class and non-class models prevents the confidence values being dominated by the probability assigned to the locations of the image features. We find the log of the ratio of the probabilities, then set the confidence to its average per region of interest, to allow a fair comparison between images where different numbers of regions are detected.

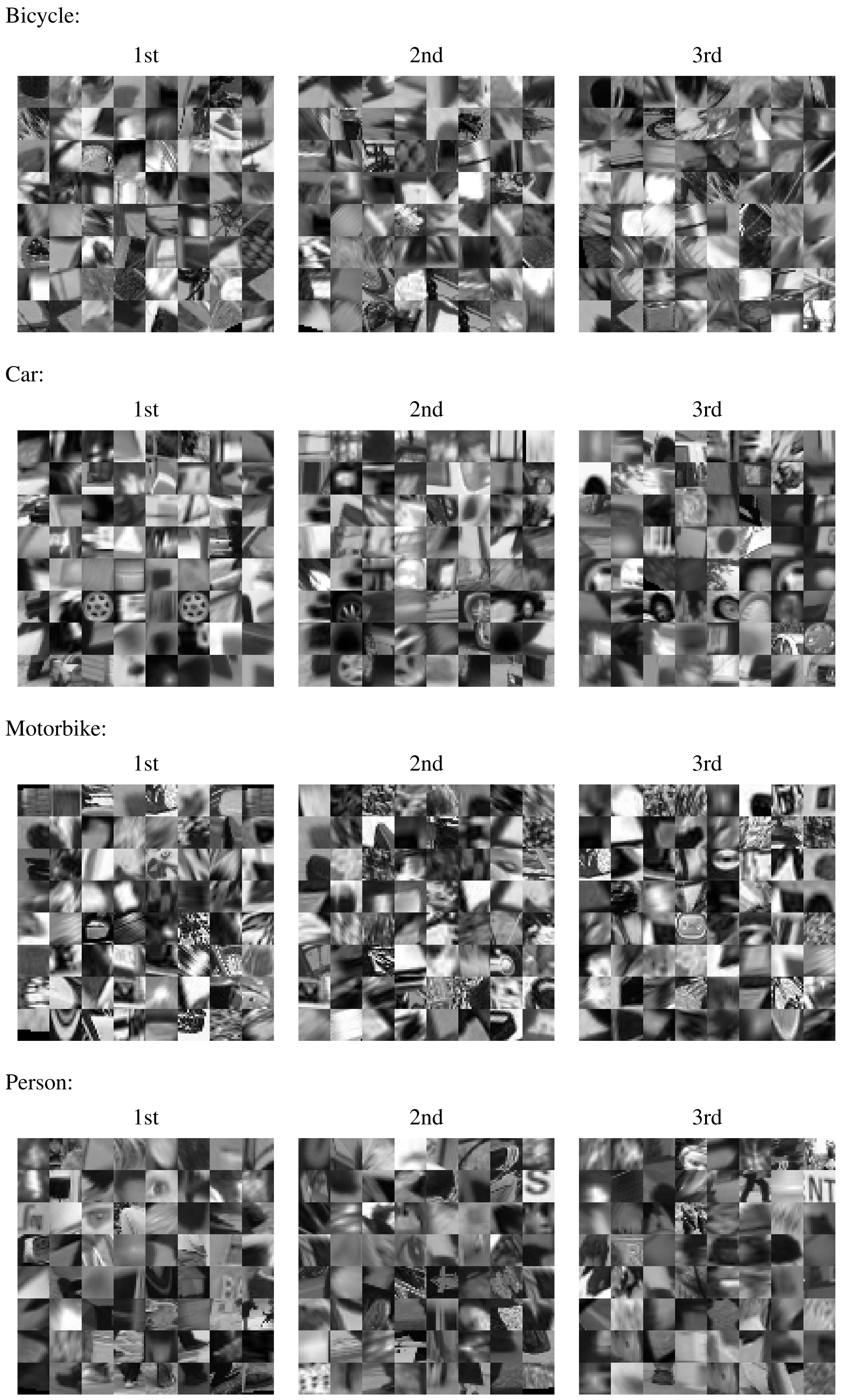


Figure 6.12: Visual words most strongly associated with the component parts of example 600-cluster GTFs. Each GTF has an 8×8 grid of component parts.

	bicycle	car	motorbike	person
AP	0.268	0.472	0.743	0.030
AP2	0.493	0.488	0.865	0.034
AUC	0.817	0.937	0.902	0.797
AUC2	0.935	0.939	0.985	0.846
recall	44	95	99	6
objects	60	163	108	71
images	57	136	107	42

Table 6.2: Validation data performance evaluation of 8×8 GTF, variance = $\left(\frac{1}{8}\right)^2$, region scale factor = 1, 120 clusters per class.

As in Everingham et al. (2005), the performance of each object category detector is measured by calculating the average precision (AP) and the area under the receiver-operating-characteristics curve (AUC), for detections made on the whole set of test images, including images where there is no object of the given category. The average precision here is the mean precision at a set of 11 equally-spaced recall levels. The AP evaluates object detection and localisation performance, while the AUC evaluates image classification performance.

In the experiments below we only look for a single object in each image. Higher recall could be achieved by allowing multiple detections per image.

6.4 Choice of GTF parameters

This section looks at how the GTF performance varies with the choice of parameters, using the PASCAL Visual Object Classes Challenge 2005 training and validation data.

Table 6.2 shows evaluation scores for the performance of four GTF object category detectors trained using the same parameter settings. The AP and AUC rows give the average precision and area under the ROC curve. A GTF with 8×8 component parts was used, with each part generating feature locations from a Gaussian with variance $\left(\frac{1}{8}\right)^2$. The image regions were left at their detected scales, and visual words were created with 120 clusters for each of the four classes and 120 for background features.

The AP2 and AUC2 rows make the assumption that there is only one class of object present in the image, and compare each model's output against the maximum output from the other class models on the same image. Since in the PASCAL 2005 Visual Object Classes data there

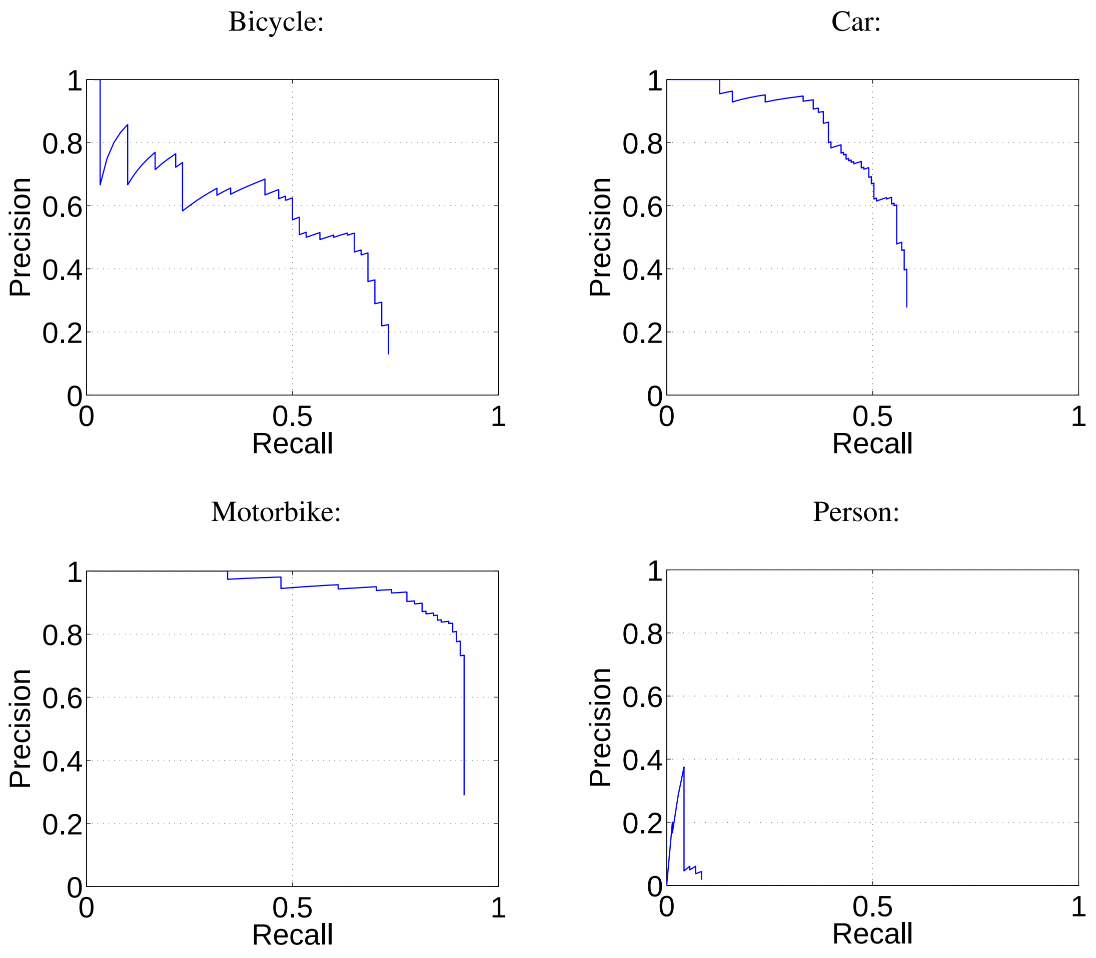


Figure 6.13: Precision-recall curves for GTF as in Table 6.2, as summarised in AP2 scores.

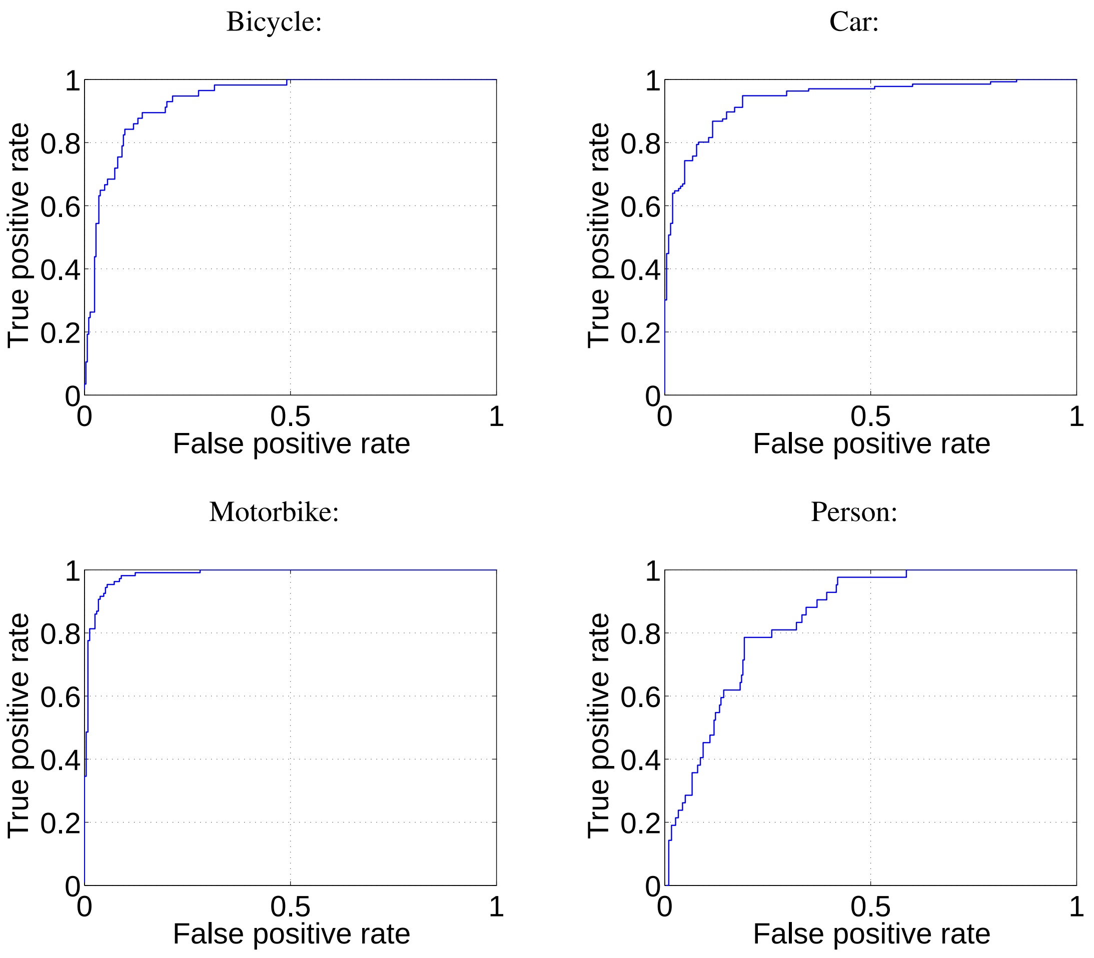


Figure 6.14: ROC curves for GTF as in Table 6.2, as summarised in AUC2 scores.

are relatively few images with more than one object, this increases performance in all cases except when the model for the class in question is much better than the other classes' models.

The recall row shows the number of images where the chosen bounding box prediction corresponded to a true object of the class in question. Since we only make one detection per image, the maximum number of objects we could detect would be the number of images, while the maximum AP score we could achieve would be the number of images divided by the number of objects, if we detected this proportion of the objects with precision one.

The AP and recall numbers in Table 6.2 show that the best detection performance is achieved for the motorbikes, with the cars then bicycles following, and only a few good detections are made for the people. The AP2 scores show an improvement from taking the other class detectors into account; once this is done, the bicycles overtake the cars in detection performance. Figure 6.13 shows the precision-recall curves for these AP2 numbers.

The AUC numbers show that the best classification performance is achieved for the cars, followed by the motorbikes, the bicycles and then the people. When the other class detectors are taken into account in the AUC2 numbers, the motorbikes overtake the cars. The classification performance for the people is relatively good compared to the localisation performance for that class: even when the predicted bounding box is too inaccurate to meet the detection criterion, the visual words present in the image sometimes allow a correct classification to be made. Figure 6.14 shows the ROC curves for these AUC2 numbers.

Figures 6.15, 6.16, 6.17 and 6.18 show the highest-confidence images for the four AUC2 classifiers. The blue rectangles show the bounding boxes annotations from the data set, while the red rectangles show the bounding box prediction for each image from the object detector in question. The yellow dots show feature locations. All of the top detections for cars and motorbikes are correct. There is one incorrect bicycle detection, for an image where the only relevant object present (a car) is very small in the image, and where few features were detected. Three of the six highest-confidence images for the person classifier contain people, but in only one of these is the predicted bounding box accepted as correct.

Table 6.3 shows the effect of changing the scale factor applied to the regions of interest before clustering regions to create visual words. For the bicycles and motorbikes there is some benefit from expanding the regions by a scale factor of two or three, but the car detector AP

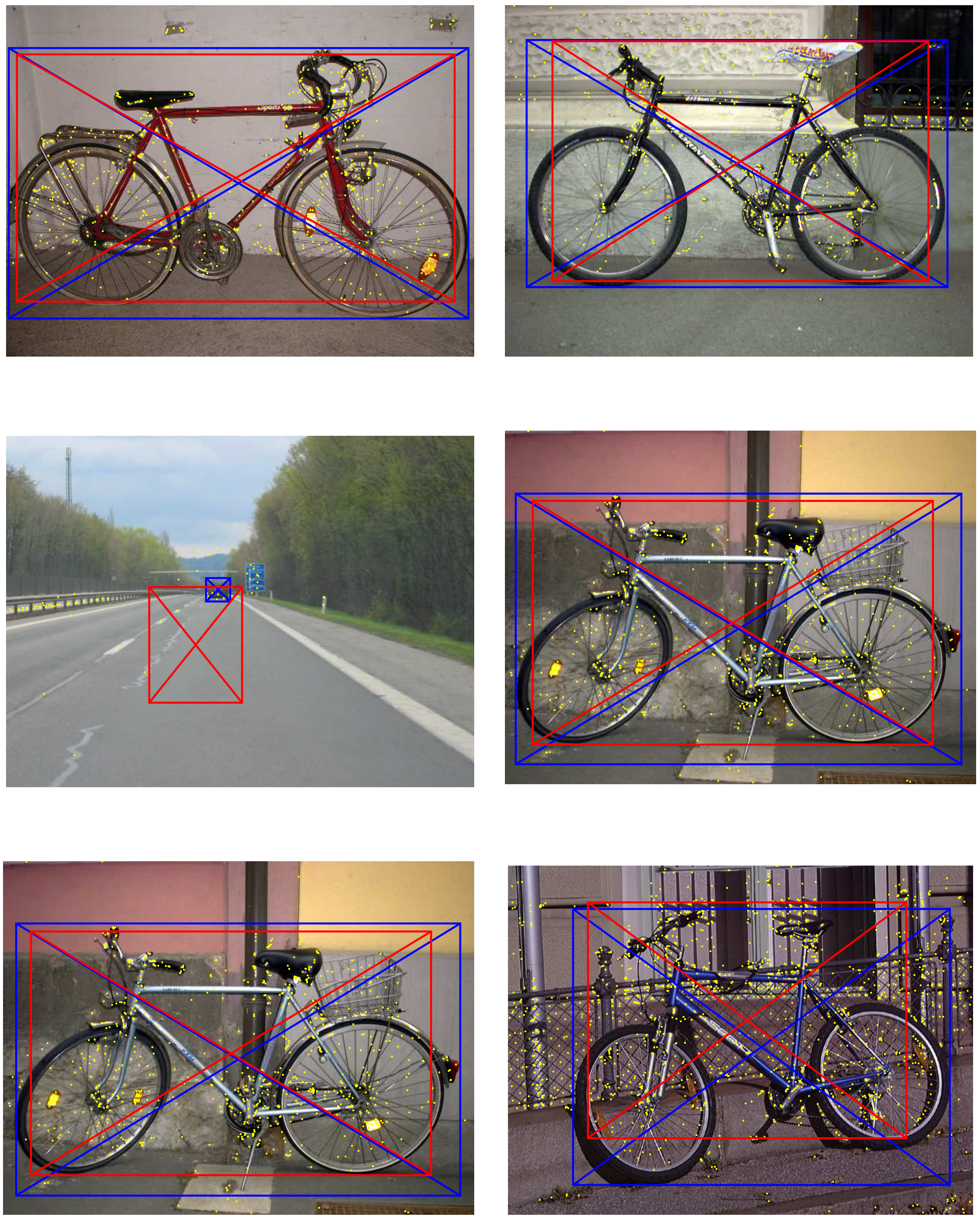


Figure 6.15: Highest-confidence images for bicycle classifier for GTF as in Table 6.2.

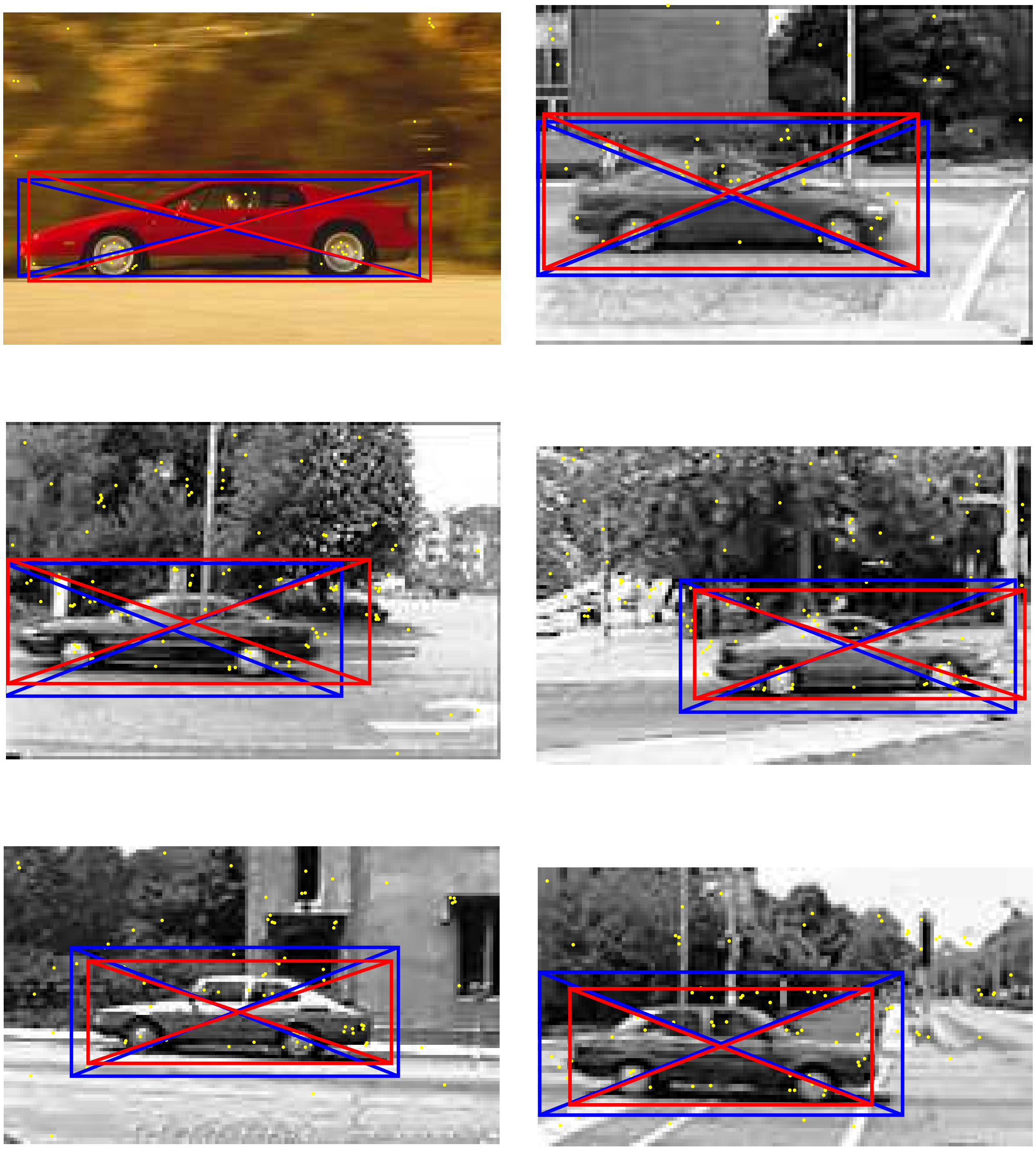


Figure 6.16: Highest-confidence images for car classifier for GTF as in Table 6.2.

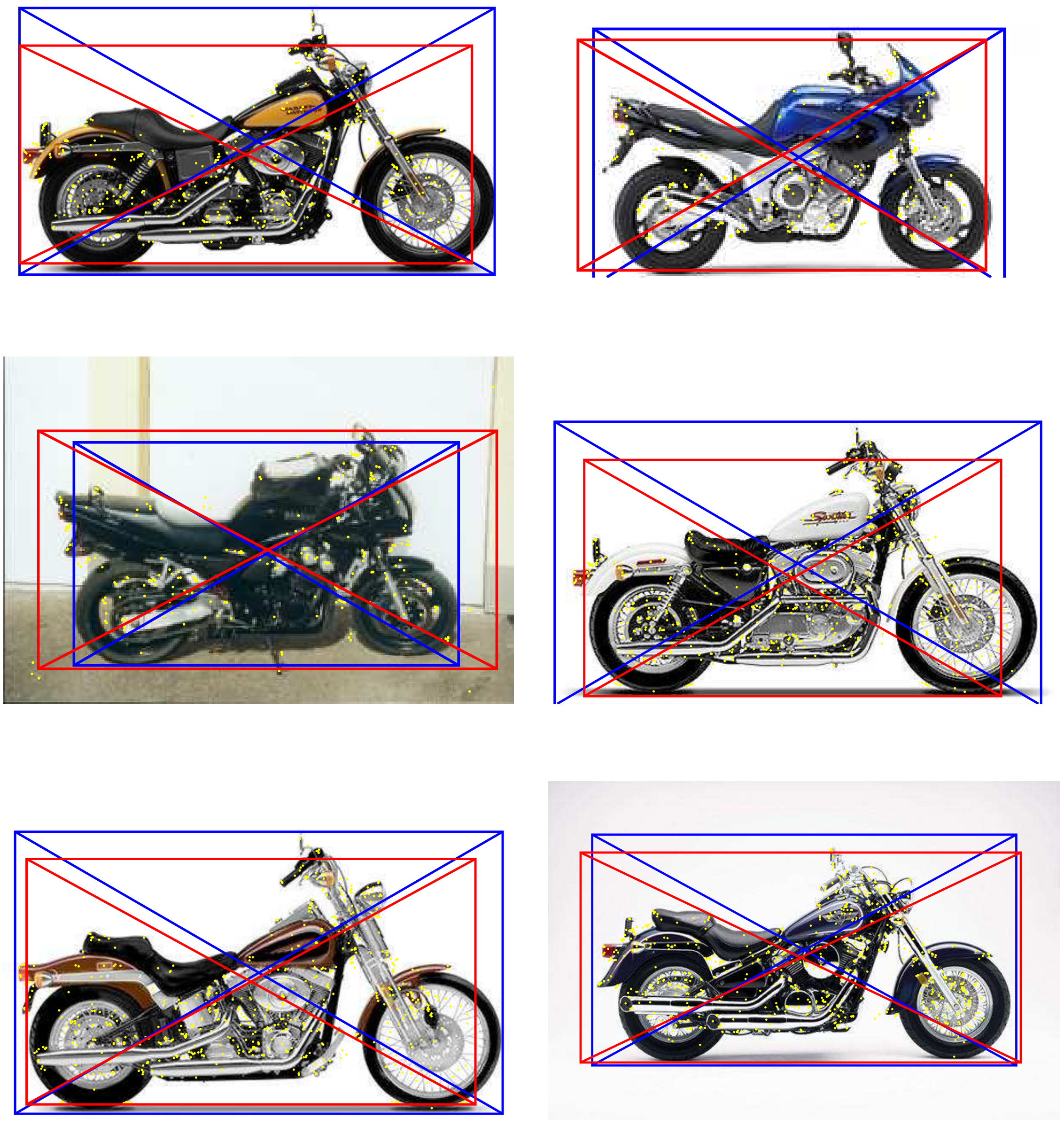


Figure 6.17: Highest-confidence images for motorbike classifier for GTF as in Table 6.2.

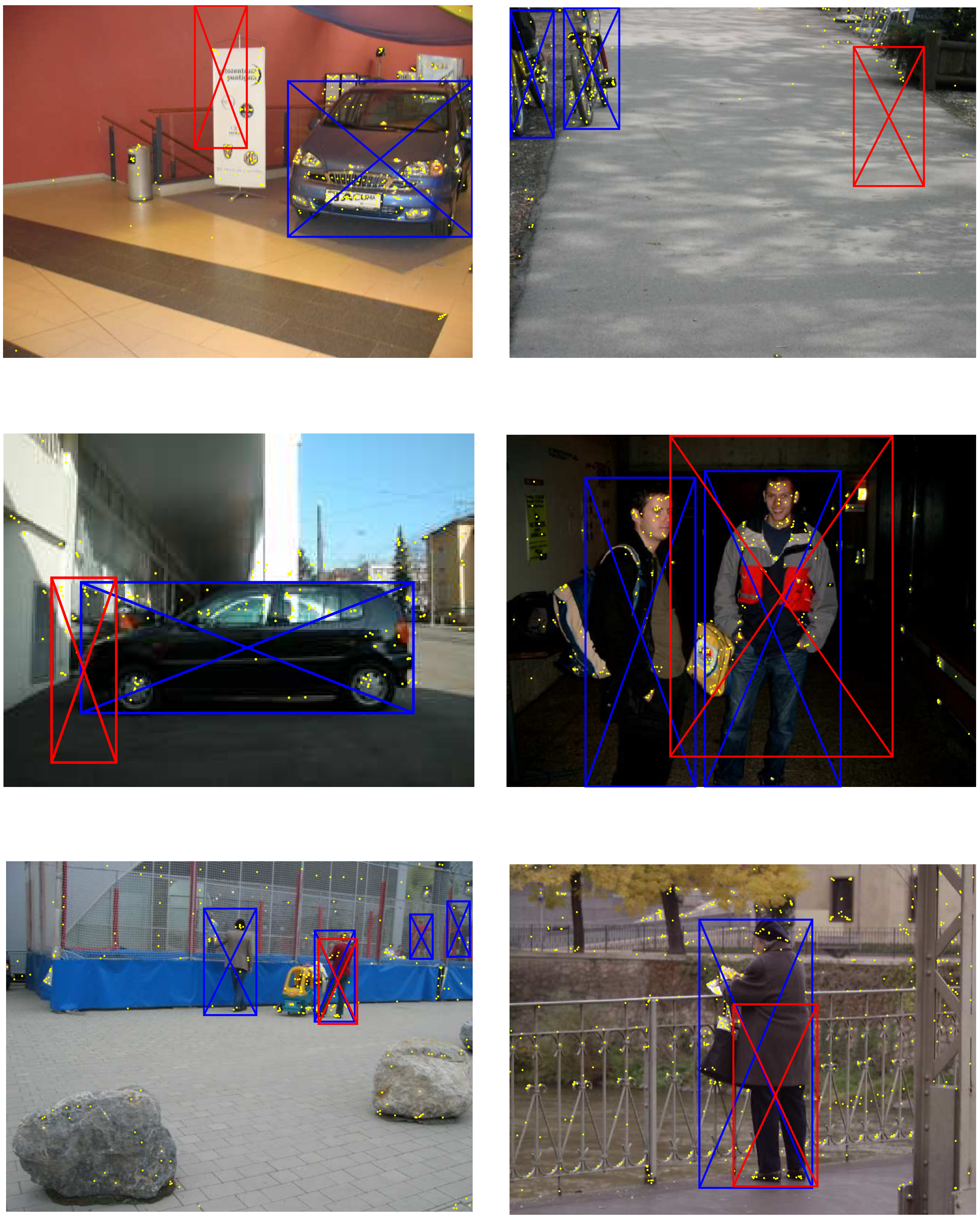


Figure 6.18: Highest-confidence images for person classifier for GTF as in Table 6.2.

Region scale factor = 0.5:

	bicycle	car	motorbike	person
AP	0.167	0.391	0.560	0.030
AP2	0.383	0.383	0.747	0.002
AUC	0.723	0.907	0.828	0.746
AUC2	0.899	0.857	0.952	0.808
recall	42	85	97	3

Region scale factor = 0.75:

	bicycle	car	motorbike	person
AP	0.249	0.380	0.657	0.035
AP2	0.459	0.392	0.778	0.052
AUC	0.838	0.928	0.895	0.779
AUC2	0.945	0.926	0.984	0.869
recall	41	82	97	8

Region scale factor = 1:

	bicycle	car	motorbike	person
AP	0.268	0.472	0.743	0.030
AP2	0.493	0.488	0.865	0.034
AUC	0.817	0.937	0.902	0.797
AUC2	0.935	0.939	0.985	0.846
recall	44	95	99	6

Region scale factor = 2:

	bicycle	car	motorbike	person
AP	0.311	0.353	0.812	0.012
AP2	0.522	0.371	0.903	0.045
AUC	0.836	0.957	0.929	0.761
AUC2	0.942	0.956	0.991	0.894
recall	45	81	102	6

Region scale factor = 3:

	bicycle	car	motorbike	person
AP	0.309	0.352	0.822	0.006
AP2	0.507	0.372	0.904	0.045
AUC	0.811	0.961	0.935	0.724
AUC2	0.934	0.965	0.995	0.907
recall	43	81	102	5

Table 6.3: Effect of changing region scale factor (8×8 GTF, variance = $\left(\frac{1}{8}\right)^2$, 120 clusters per class).

and AP2 numbers drop significantly when this is done. Reducing the regions to three quarters of their original size improves the person detector performance slightly, but given the poor overall performance of this classifier, this improvement may be due to chance. The overall recall numbers are best when the regions are left at their originally-detected scales.

Table 6.4 shows the effect of changing the number of clusters used to create visual words. Initially increasing the number of clusters increases performance significantly for the bicycle and car detectors, although classification performance does not vary greatly. By 160 clusters per class (800 clusters in total) the bicycle and car detectors' performance drops off. Initially there is a localisation benefit from having more specific clusters, but if there are too many clusters then there are not enough examples for each cluster to maintain performance levels. The overall recall numbers are best with 120 clusters per class.

Table 6.5 shows the effect of changing the variance used in the Gaussians $p(\mathbf{x}|z)$ in the GTF. Too small a variance performs badly, but there is comparatively little difference between the results for the final three variance settings. Different class detectors perform best at different variance settings.

Table 6.6 shows the effect of changing the size of the grid of component parts used in the GTF. The classification performance does not vary greatly across the different GTF sizes. Localisation performance seems to improve with GTF size, though it falls for the cars with the 12×12 GTF.

Table 6.7 compares the performance of the GTF examined above with that achieved by two variations on the model. The first comparison is with a GTF which has a single foreground distribution shared by all the component parts, instead of having $p(w_{mi}|z_{mi})$ learnt separately for each part. The performance on the people does not change greatly. The classification performance for the other three classes falls. The localisation performance for the cars and motorbikes falls significantly, with the recall level for the cars almost halving. The bicycle localisation performance actually improves: this is probably because we have a fairly small training data set, while bicycles are locally quite similar all over.

The second comparison in Table 6.7 is with a GTF with the distribution over background feature locations $p_b(\mathbf{x})$ is uniform across the whole image, instead of the background feature location distribution described in section 5.3.1 which has a 'hole' the same shape as the fore-

10 clusters per class:

	bicycle	car	motorbike	person
AP	0.251	0.276	0.700	0.095
AP2	0.465	0.274	0.847	0.034
AUC	0.773	0.870	0.865	0.793
AUC2	0.926	0.888	0.978	0.866
recall	43	62	99	8

20 clusters per class:

	bicycle	car	motorbike	person
AP	0.375	0.348	0.698	0.023
AP2	0.515	0.345	0.849	0.021
AUC	0.830	0.879	0.872	0.754
AUC2	0.934	0.894	0.974	0.842
recall	44	71	99	8

40 clusters per class:

	bicycle	car	motorbike	person
AP	0.349	0.393	0.699	0.032
AP2	0.574	0.384	0.855	0.040
AUC	0.835	0.908	0.874	0.763
AUC2	0.934	0.913	0.980	0.861
recall	44	76	99	11

80 clusters per class:

	bicycle	car	motorbike	person
AP	0.323	0.435	0.764	0.091
AP2	0.467	0.441	0.862	0.045
AUC	0.835	0.943	0.918	0.768
AUC2	0.934	0.946	0.988	0.853
recall	41	88	100	6

120 clusters per class:

	bicycle	car	motorbike	person
AP	0.268	0.472	0.743	0.030
AP2	0.493	0.488	0.865	0.034
AUC	0.817	0.937	0.902	0.797
AUC2	0.935	0.939	0.985	0.846
recall	44	95	99	6

160 clusters per class:

	bicycle	car	motorbike	person
AP	0.256	0.412	0.781	0.045
AP2	0.458	0.407	0.874	0.016
AUC	0.822	0.945	0.913	0.803
AUC2	0.923	0.948	0.985	0.880
recall	41	87	100	5

Table 6.4: Effect of changing number of clusters per class (8×8 GTF, variance = $\left(\frac{1}{8}\right)^2$, region-scale = 1).

Variance = $\left(\frac{1}{8}\right)^2 \times \frac{1}{2}$:

	bicycle	car	motorbike	person
AP	0.157	0.358	0.716	0.018
AP2	0.347	0.382	0.852	0.023
AUC	0.764	0.930	0.874	0.800
AUC2	0.913	0.917	0.979	0.799
recall	37	77	99	2

Variance = $\left(\frac{1}{8}\right)^2$:

	bicycle	car	motorbike	person
AP	0.268	0.472	0.743	0.030
AP2	0.493	0.488	0.865	0.034
AUC	0.817	0.937	0.902	0.797
AUC2	0.935	0.939	0.985	0.846
recall	44	95	99	6

Variance = $\left(\frac{1}{8}\right)^2 \times 2$:

	bicycle	car	motorbike	person
AP	0.360	0.480	0.765	0.055
AP2	0.558	0.486	0.873	0.048
AUC	0.863	0.950	0.914	0.812
AUC2	0.948	0.950	0.987	0.886
recall	46	92	100	10

Variance = $\left(\frac{1}{8}\right)^2 \times 3$:

	bicycle	car	motorbike	person
AP	0.346	0.466	0.753	0.056
AP2	0.520	0.487	0.855	0.055
AUC	0.878	0.953	0.914	0.825
AUC2	0.947	0.954	0.985	0.897
recall	44	91	99	11

Table 6.5: Effect of changing variance (8×8 GTF, region scale factor = 1, 120 clusters per class).

4×4 GTF:

	bicycle	car	motorbike	person
AP	0.293	0.453	0.741	0.030
AP2	0.564	0.463	0.860	0.030
AUC	0.823	0.941	0.892	0.795
AUC2	0.928	0.939	0.974	0.851
recall	48	91	99	6

6×6 GTF:

	bicycle	car	motorbike	person
AP	0.276	0.448	0.754	0.030
AP2	0.509	0.460	0.874	0.045
AUC	0.828	0.943	0.905	0.802
AUC2	0.942	0.940	0.985	0.844
recall	45	89	100	4

8×8 GTF:

	bicycle	car	motorbike	person
AP	0.268	0.472	0.743	0.030
AP2	0.493	0.488	0.865	0.034
AUC	0.817	0.937	0.902	0.797
AUC2	0.935	0.939	0.985	0.846
recall	44	95	99	6

10×10 GTF:

	bicycle	car	motorbike	person
AP	0.290	0.479	0.746	0.030
AP2	0.512	0.490	0.865	0.019
AUC	0.822	0.938	0.902	0.806
AUC2	0.939	0.939	0.986	0.840
recall	44	97	99	6

12×12 GTF:

	bicycle	car	motorbike	person
AP	0.290	0.469	0.747	0.030
AP2	0.553	0.487	0.875	0.036
AUC	0.824	0.943	0.901	0.803
AUC2	0.940	0.941	0.985	0.854
recall	46	95	100	7

Table 6.6: Effect of changing GTF size (variance = $\left(\frac{1}{8}\right)^2$, region scale factor = 1, 120 clusters per class).

Results as in Table 6.2 above:

	bicycle	car	motorbike	person
AP	0.268	0.472	0.743	0.030
AP2	0.493	0.488	0.865	0.034
AUC	0.817	0.937	0.902	0.797
AUC2	0.935	0.939	0.985	0.846
recall	44	95	99	6

Single shared foreground distribution:

	bicycle	car	motorbike	person
AP	0.345	0.224	0.667	0.032
AP2	0.498	0.213	0.767	0.022
AUC	0.883	0.898	0.893	0.801
AUC2	0.915	0.913	0.954	0.848
recall	45	50	99	9

Uniform background model:

	bicycle	car	motorbike	person
AP	0.278	0.350	0.712	0.010
AP2	0.432	0.349	0.794	0.004
AUC	0.846	0.909	0.899	0.757
AUC2	0.922	0.921	0.981	0.839
recall	39	67	96	2

Table 6.7: Comparison of the performance of the GTF as in Table 6.2 with variations on the model.

	bicycle	car	motorbike	person
AP	0.265	0.411	0.760	0.014
AP2	0.467	0.422	0.888	0.030
AUC	0.855	0.936	0.908	0.822
AUC2	0.966	0.943	0.995	0.890
recall	75	176	205	13
objects	123	341	220	149
images	114	275	216	84

Table 6.8: Test data performance evaluation of 8×8 GTF, variance = $\left(\frac{1}{8}\right)^2$, region scale factor = 1, 120 clusters per class.

ground bounding box. The classification performance and AP2 numbers fall for all the classes. The recall levels fall, most notably for the cars. With the uniform background model, the car predictions tend to expand to cover adjacent background features, whereas with the model we use elsewhere assigns a lower probability to background features within the object bounding box hypothesis.

6.5 Test data performance

This section looks in more detail at the GTF's performance on the PASCAL Visual Object Classes Challenge 2005 test data. We use a single set of parameters, chosen from inspection of the above validation data results, with the aim of finding a compromise which gives good localisation performance for all the object categories: we leave the regions at their original scales, find 120 clusters per class, set the variance of the Gaussians $p(\mathbf{x}|z)$ to $\left(\frac{1}{8}\right)^2$, and use an 8×8 grid of GTF component parts.

Table 6.8 shows evaluation scores for the performance on the PASCAL test data of four GTF object category detectors trained with the same parameters on the combined training and validation data. The AP scores for all the classes except the motorbikes are worse than the validation data ones shown in Table 6.2, suggesting that the test data set is slightly harder than the validation data for localisation. The AUC scores for classification are higher than for the validation data, suggesting that here the increased amount of training data is increasing performance.

Figures 6.19 and 6.20 show the precision-recall and ROC curves corresponding to the AP2

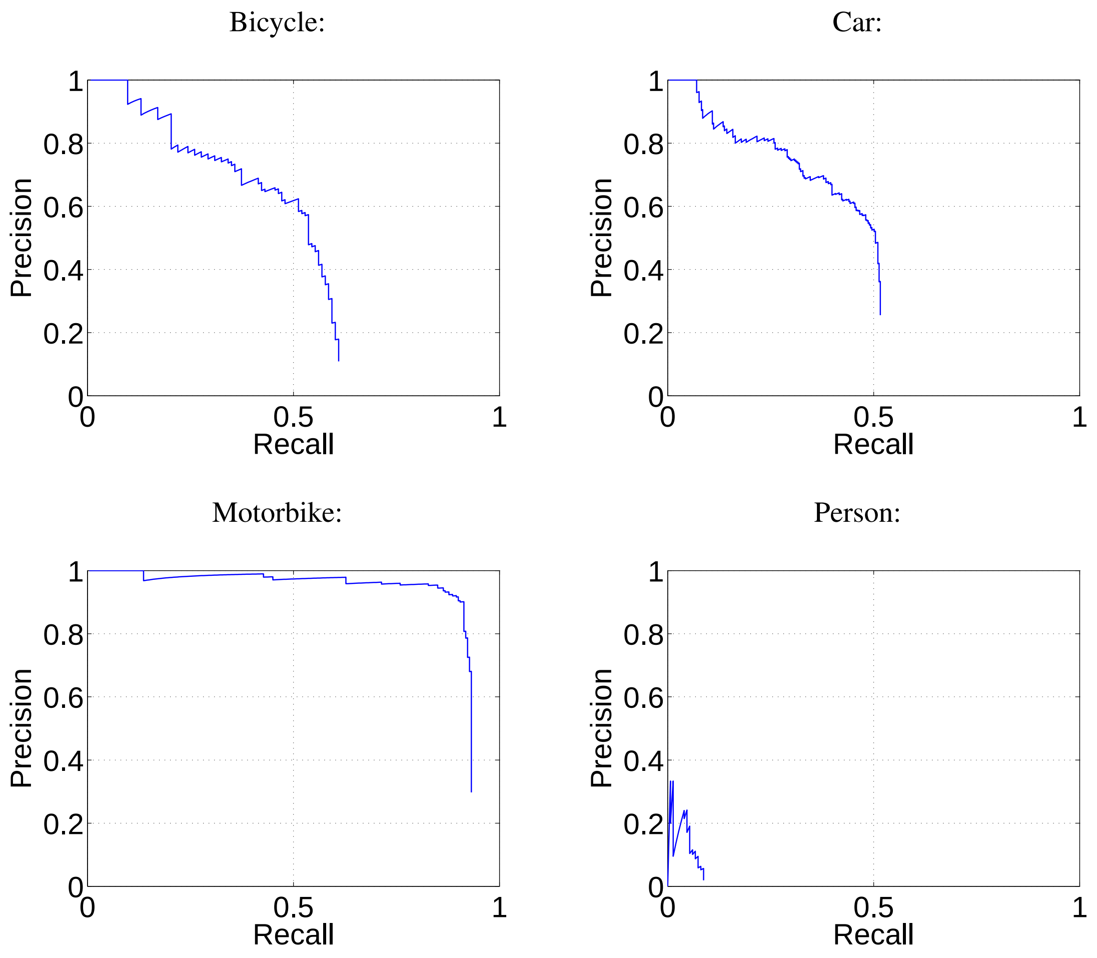


Figure 6.19: Precision-recall curves for GTF as in Table 6.8.

and AUC2 numbers in the Table. The ROC curves make clear that there is a large variation in classification performance across the classes, with the motorbike classifier by far the best, then the bicycle classifier, then the car classifier, with the person classifier significantly worse again. The precision-recall curves for the bicycle and car detectors show a similar fall-off to different recall levels. As well as reaching a much higher recall, the precision curve for the motorbike classifier remains much flatter, at a high precision level, until its final rapid fall. The precision-recall curve for the person detector shows that only a few people are found, with a low level of precision.

Figures 6.21, 6.22, 6.23 and 6.24 show the highest-confidence images with correct detections for each classifier. These figures illustrate what each class detector finds easiest. For the cars and motorbikes, these are all unoccluded side-views; for the motorbikes they also have

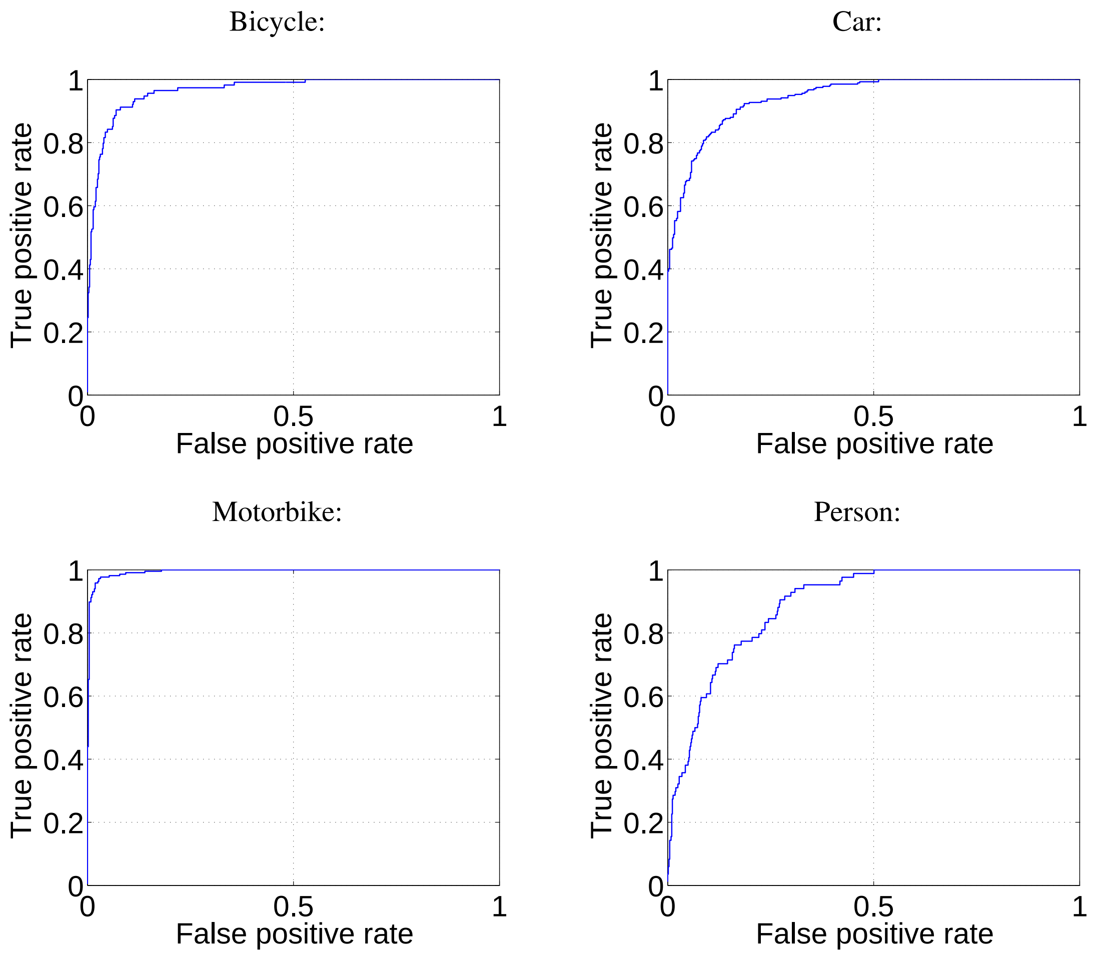


Figure 6.20: ROC curves for GTF as in Table 6.8.

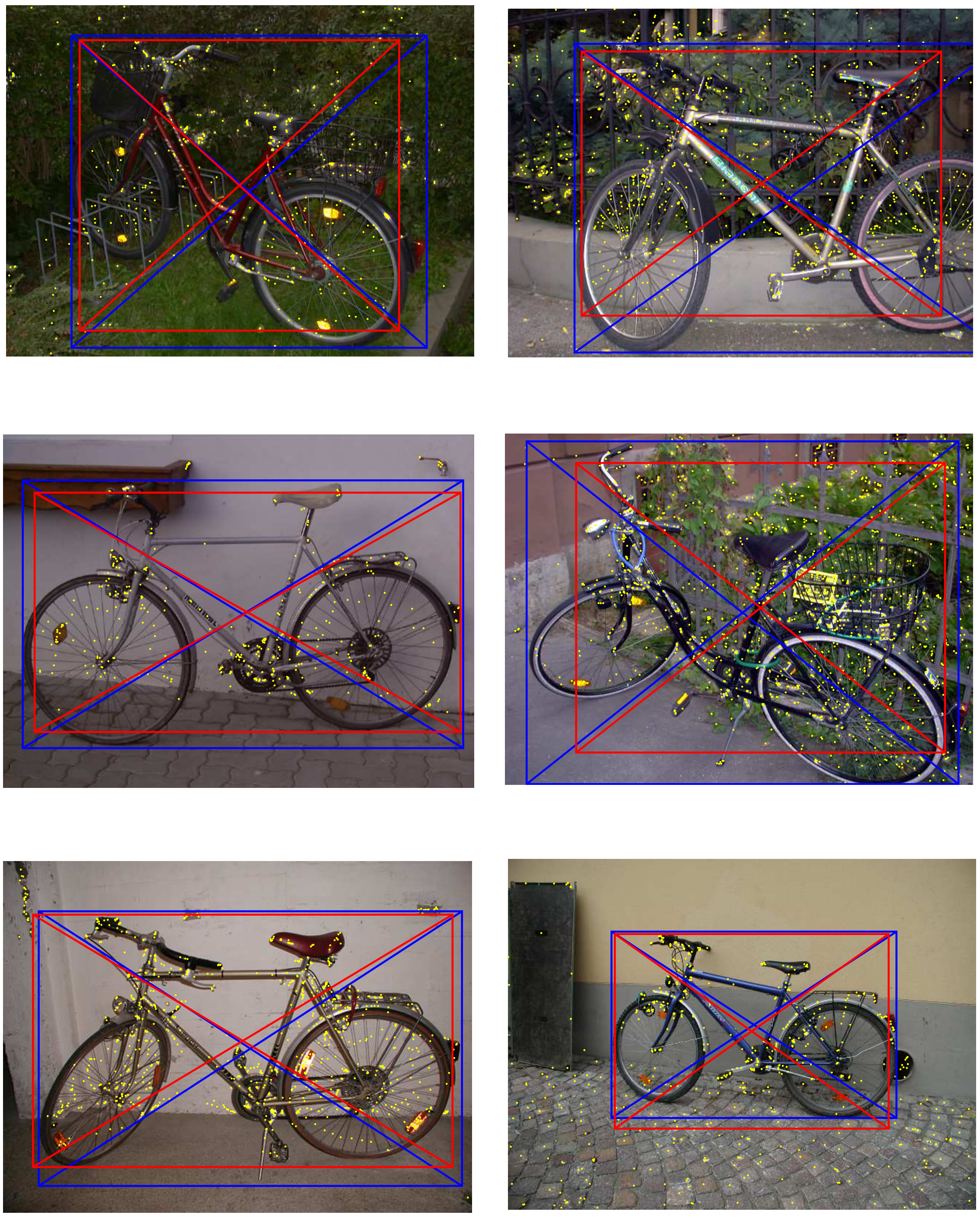


Figure 6.21: Highest-confidence images with correct detections for bicycle classifier for GTF as in Table 6.8.

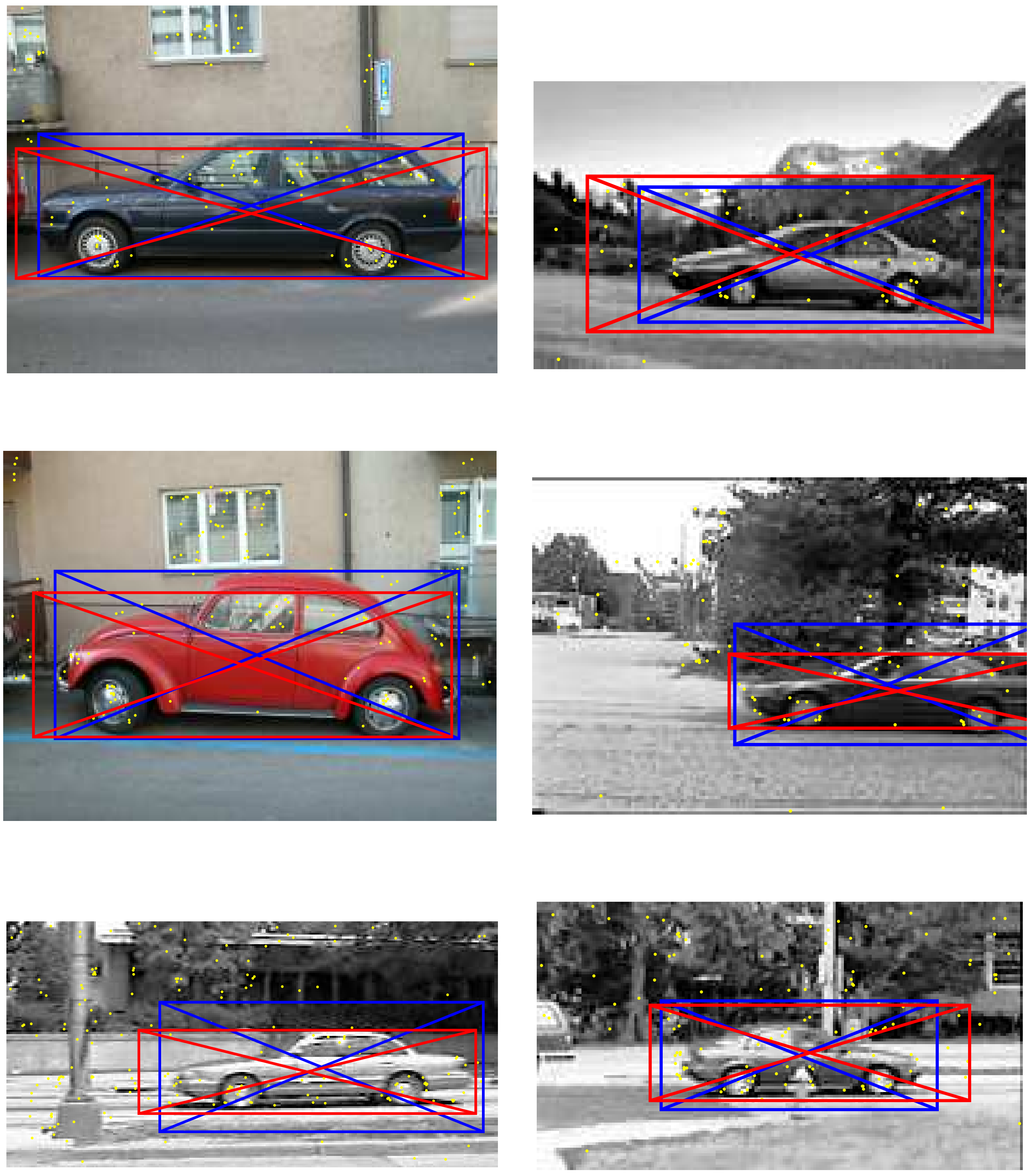


Figure 6.22: Highest-confidence images with correct detections for car classifier for GTF as in Table 6.8.

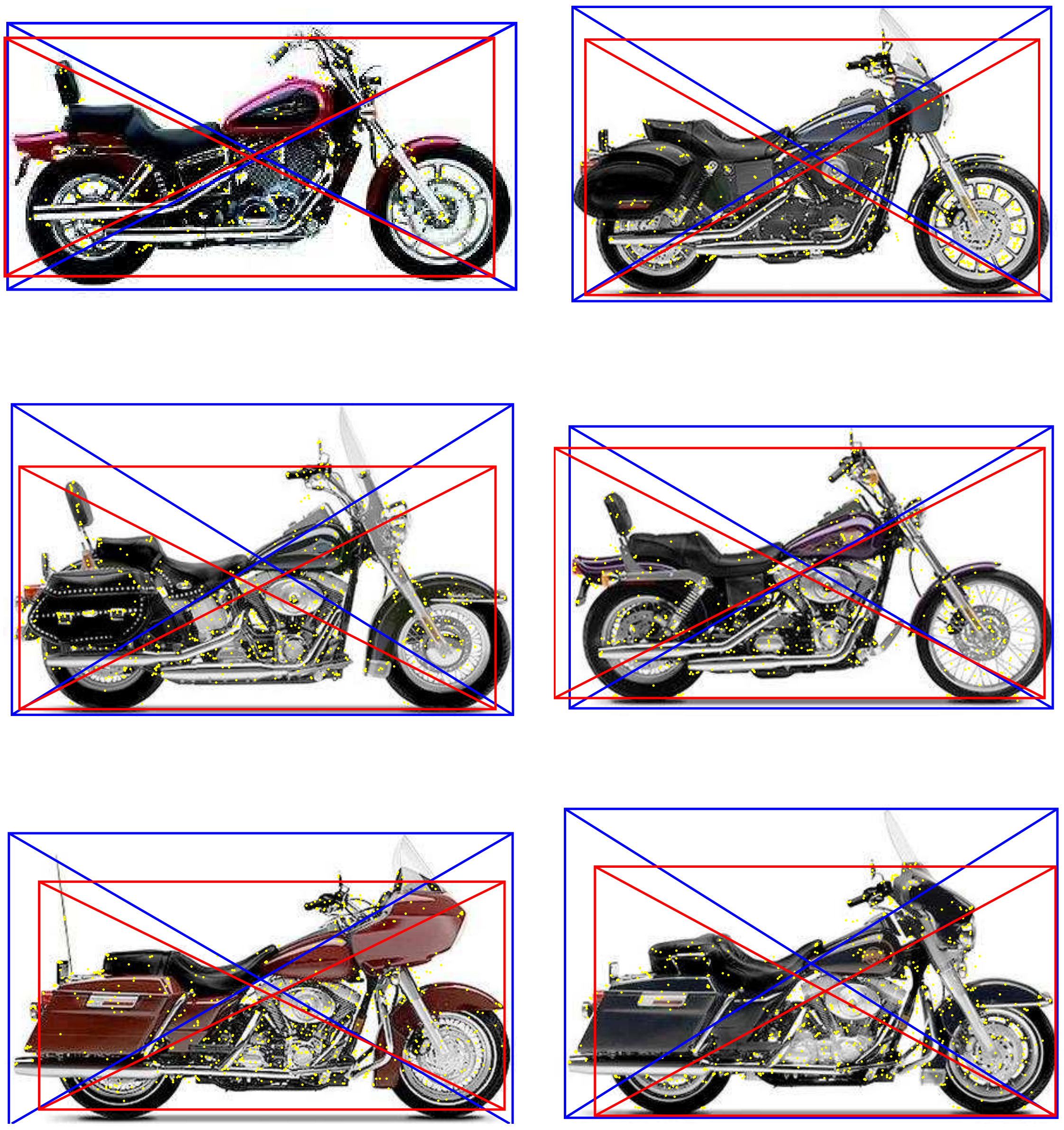


Figure 6.23: Highest-confidence images with correct detections for motorbike classifier for GTF as in Table 6.8.

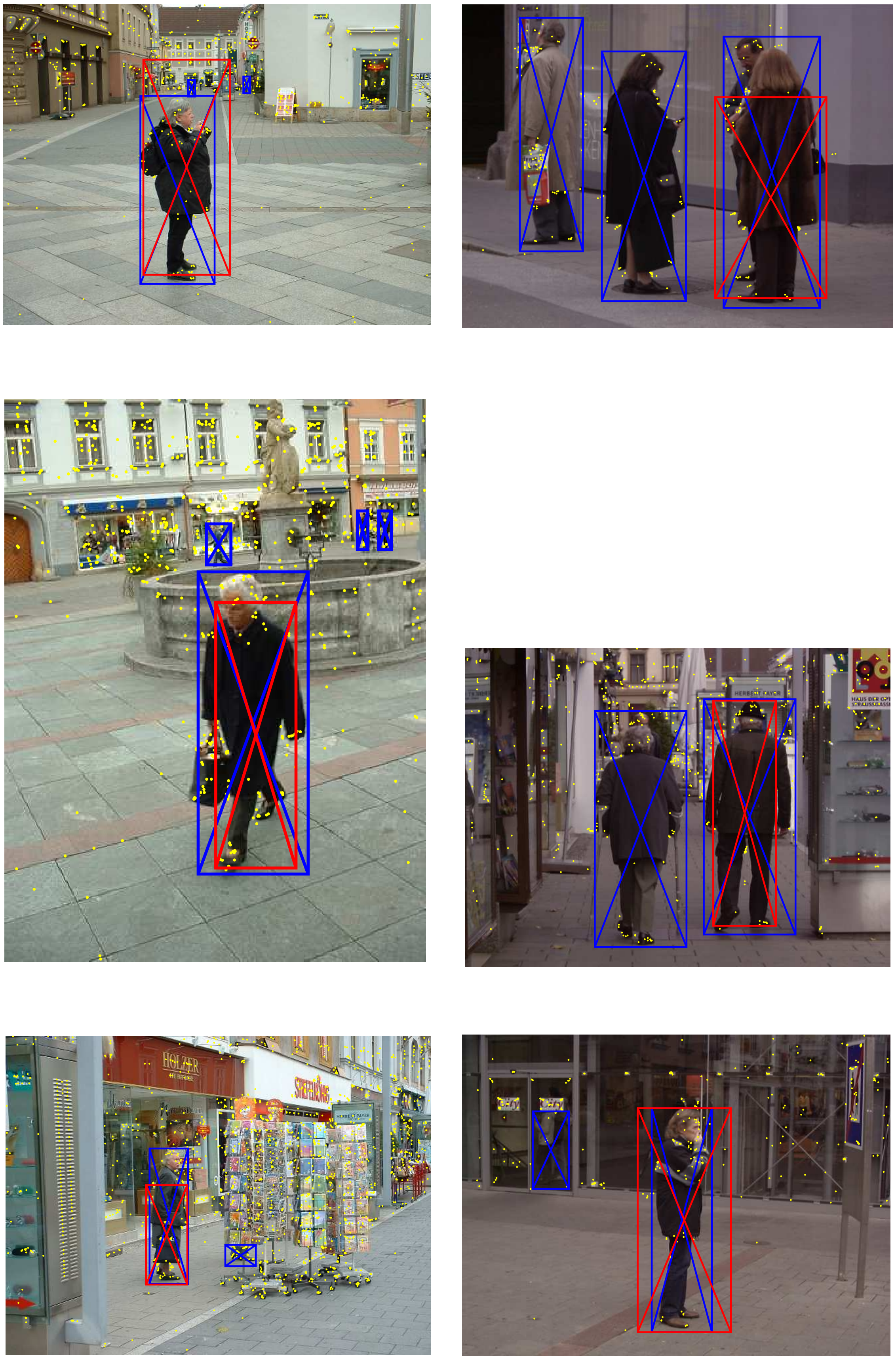


Figure 6.24: Highest-confidence images with correct detections for person classifier for GTF as in Table 6.8.

plain backgrounds. The bicycles are all fairly large in their images, with both wheels clearly visible. The people are also comparatively large in their images, and are dark against light-coloured backgrounds.

Figures 6.25, 6.26, 6.27 and 6.28 show the lowest-confidence images with correct detections for each classifier. These figures illustrate the limits of the capabilities of each class detector. Three of the bicycles are seen from oblique angles, showing a smaller surface area than a side-view and with fewer typical bicycle features visible. One bicycle has an unusual type of wheels, one is partly outside the image, and one is against railings, which make the local features harder to recognise as bicycle ones. The cars are comparatively small in their images; one of the cars is seen in a three-quarters view, while one is occluded by a lamp-post. Both the car and motorbike images here include vegetation and buildings in their backgrounds, presenting distracting background texture. The people are comparatively small in their images; they wear clothing that gives less contrast to the background, and in two of the images there are multiple people.

Figures 6.29, 6.30, 6.31 and 6.32 show the highest-confidence images with incorrect detections for each classifier. These figures illustrate what confuses each classifer. One of the images for the bicycle classifier is a spurious detection, as are two of the images for the motorbike classifier, and two of the images for the person classifer. The other bicycles are seen in unusual poses or from unusual viewpoints, or in one case only one wheel is visible in the image. For the rear-view bicycle the contents of the predicted bounding box do look like a bicycle by themselves, while in the case where only one wheel is visible the predicted bounding box would make sense if the bicycle was facing the other direction, with the rest occluded by the bushes rather than outside the image; in the other three bicycle images the predicted bounding box is in the right area of the image, but insufficiently accurate. All of the highest-confidence incorrect car detections, and two of the highest-confidence incorrect motorbike detections, are images where there are multiple objects of the class at similar scales, a case not dealt with by our implementation of the model. In one of the motorbike images a dark group of people behind the motorbike has been treated as an extension of the object, making the predicted bounding box insufficiently accurate. In the highest-confidence person images there are several insufficiently-accurate localisations where a head- or torso-sized bounding box prediction has

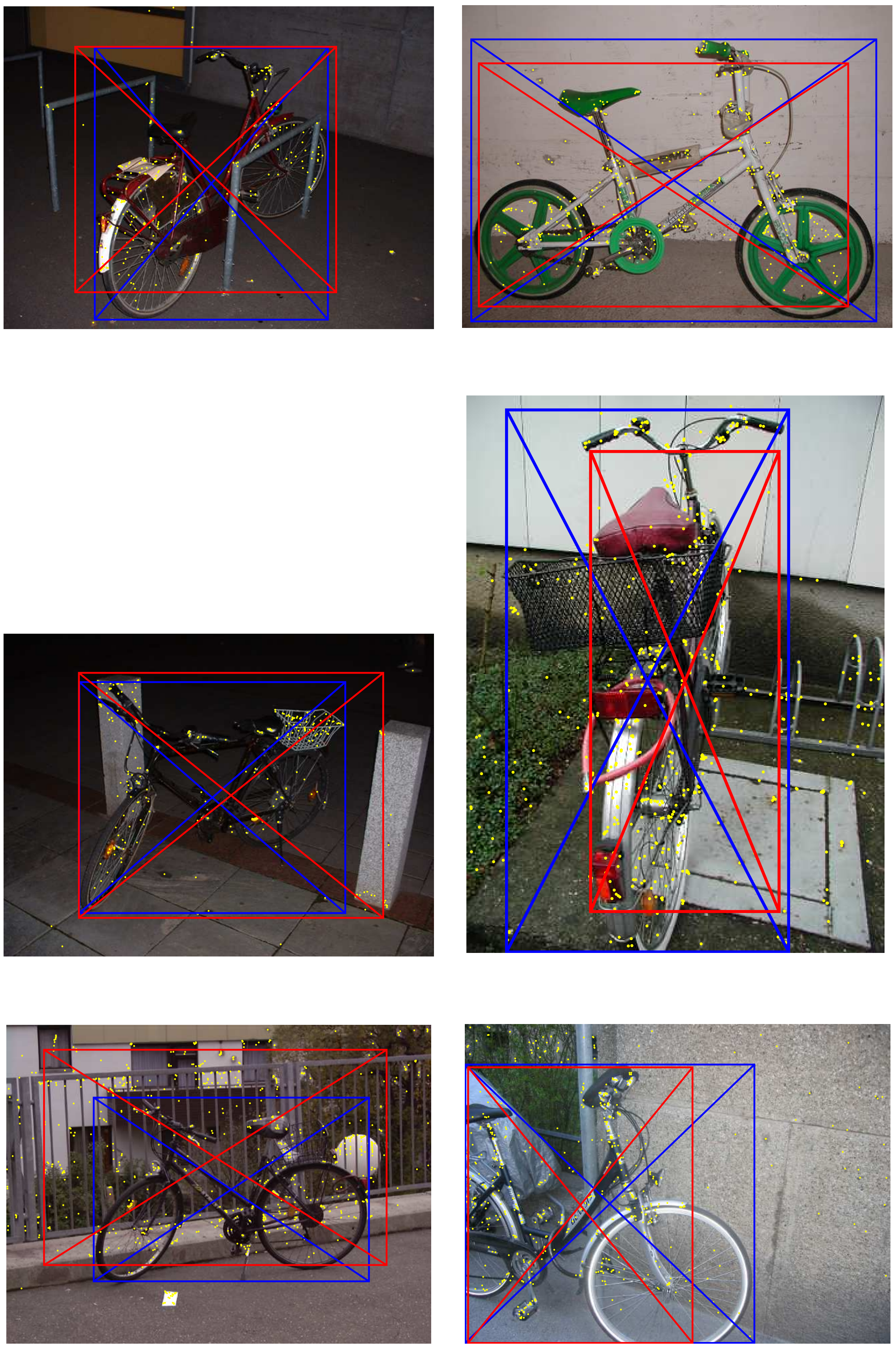


Figure 6.25: Lowest-confidence images with correct detections for bicycle classifier for GTF as in Table 6.8.

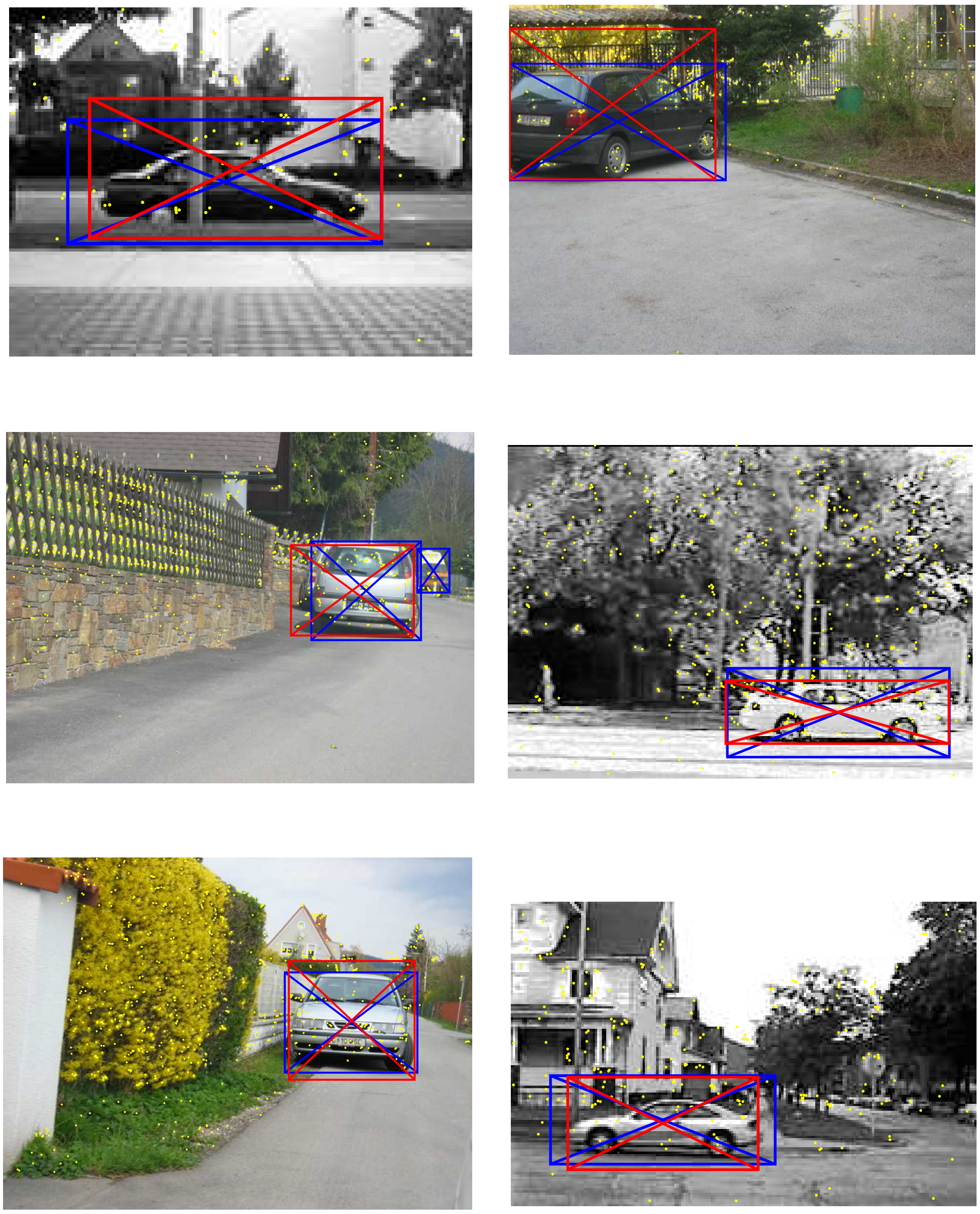


Figure 6.26: Lowest-confidence images with correct detections for car classifier for GTF as in Table 6.8.

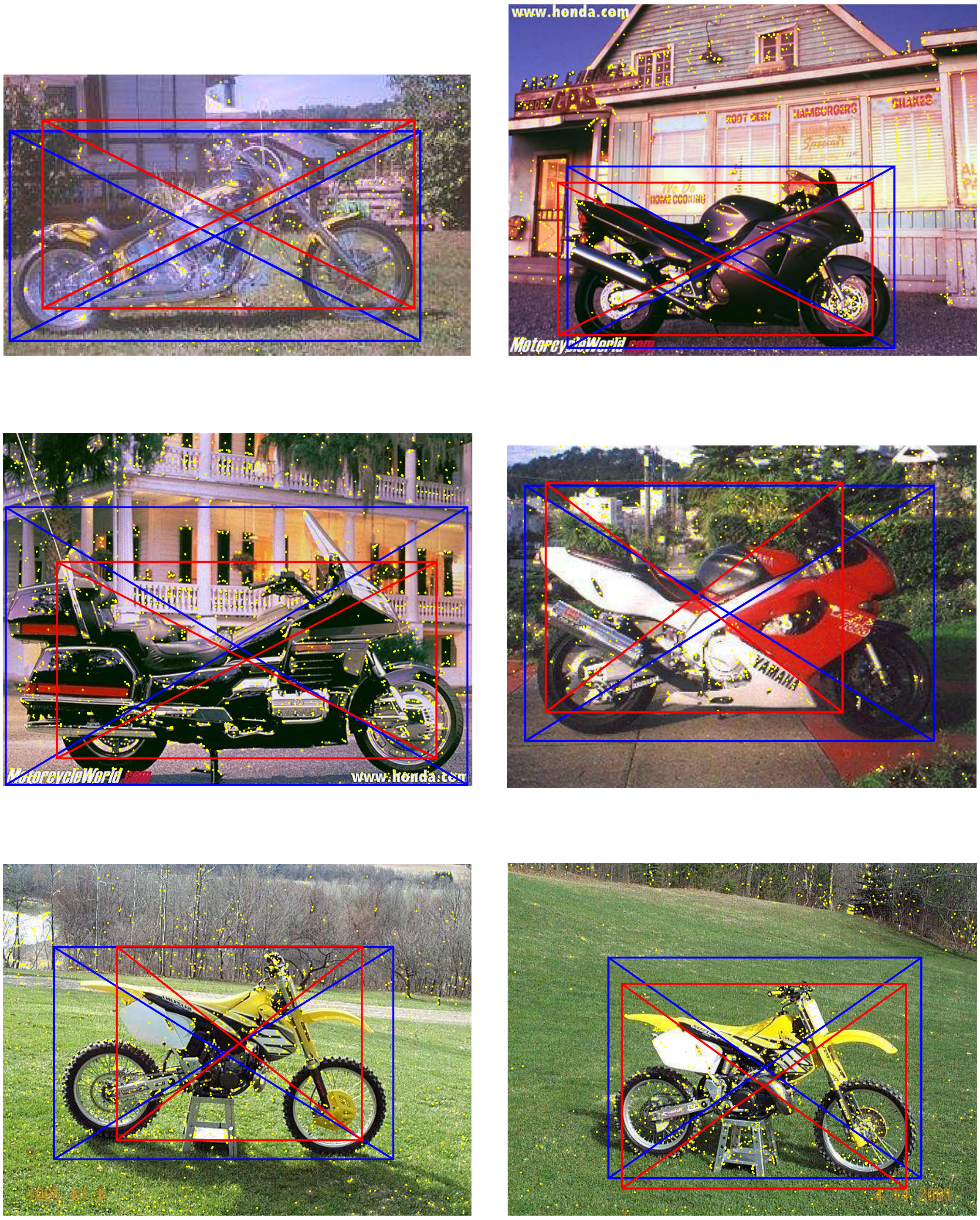


Figure 6.27: Lowest-confidence images with correct detections for motorbike classifier for GTF as in Table 6.8.

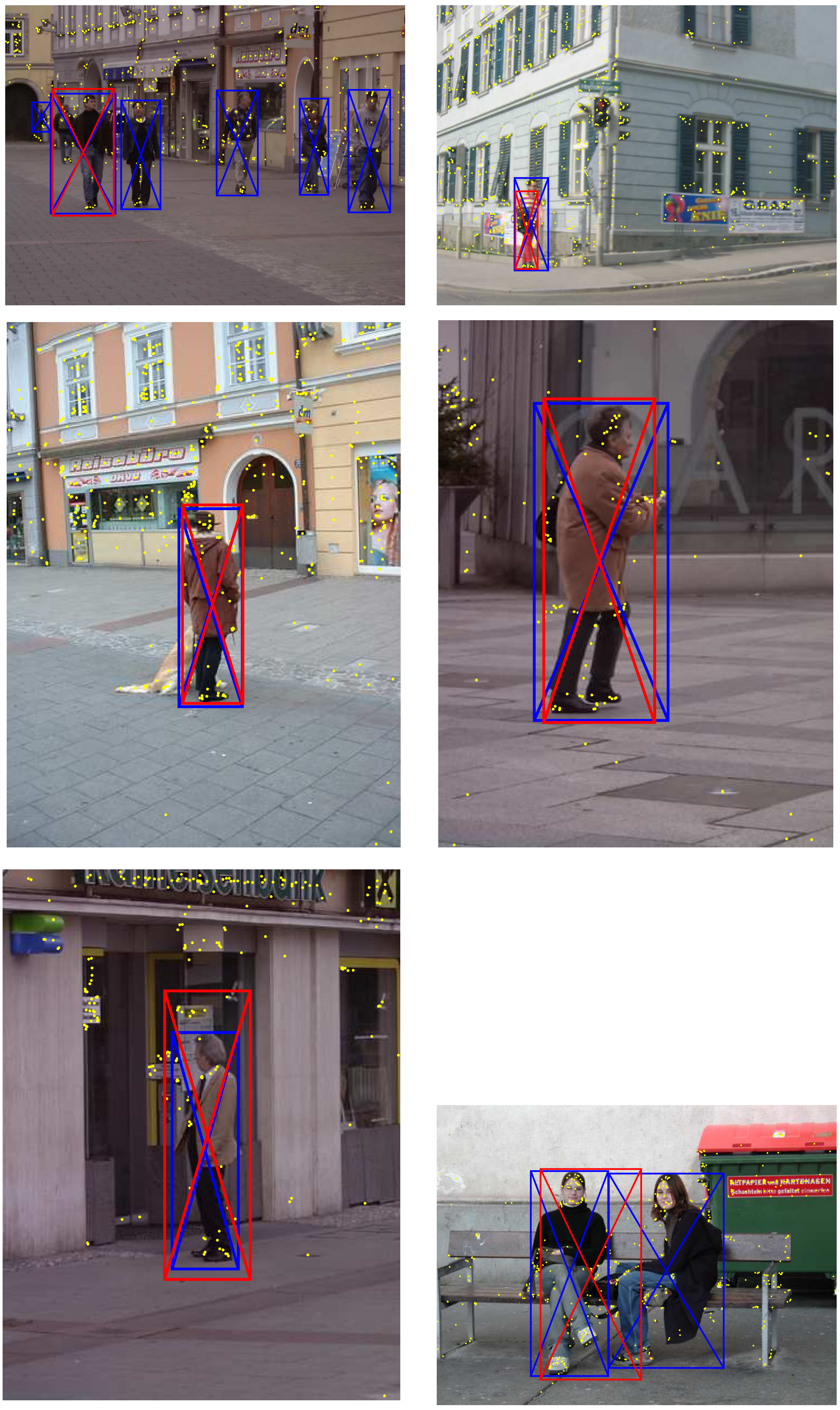


Figure 6.28: Lowest-confidence images with correct detections for person classifier for GTF as in Table 6.8.

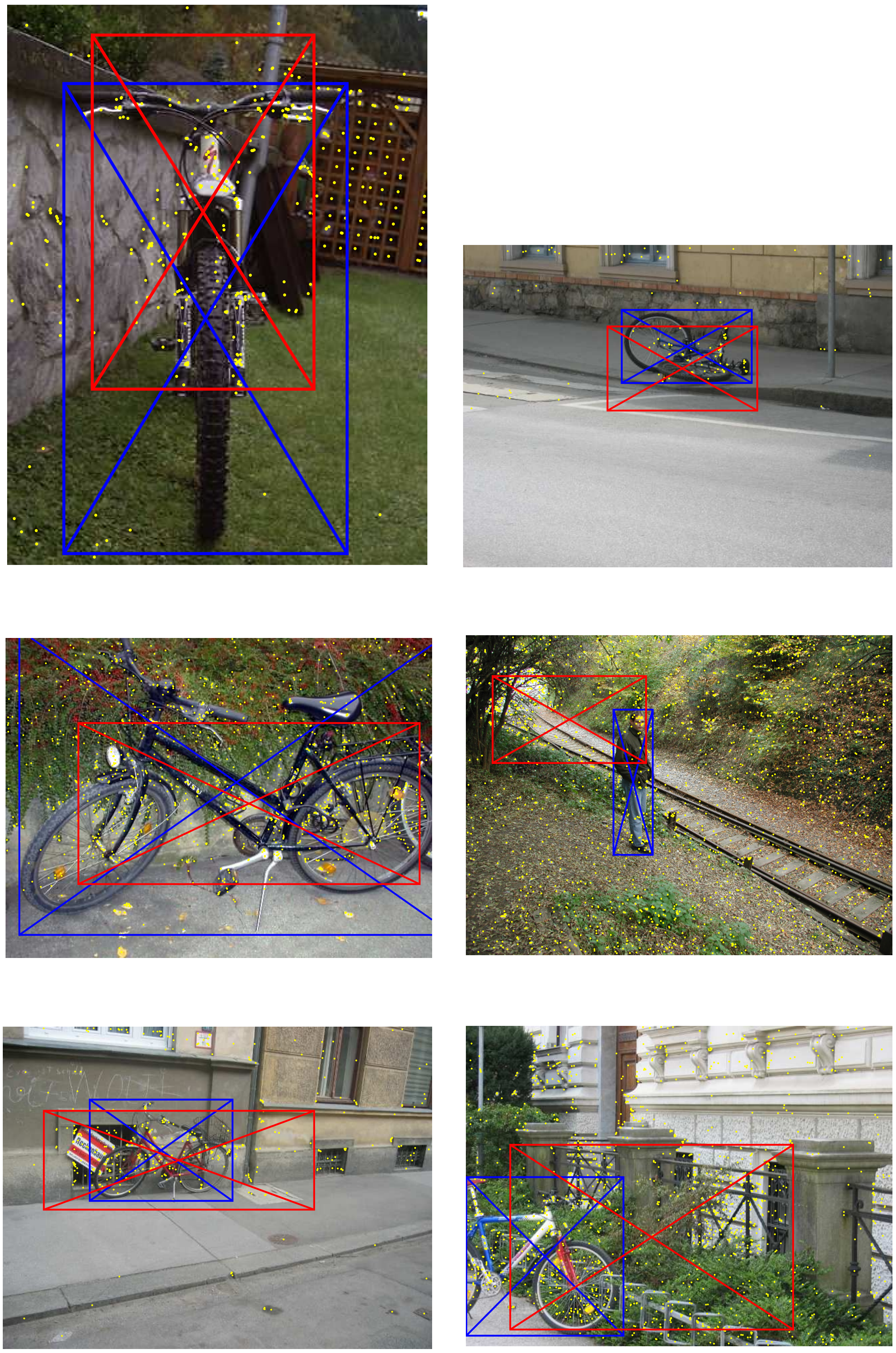


Figure 6.29: Highest-confidence images with incorrect detections for bicycle classifier for GTF as in Table 6.8.

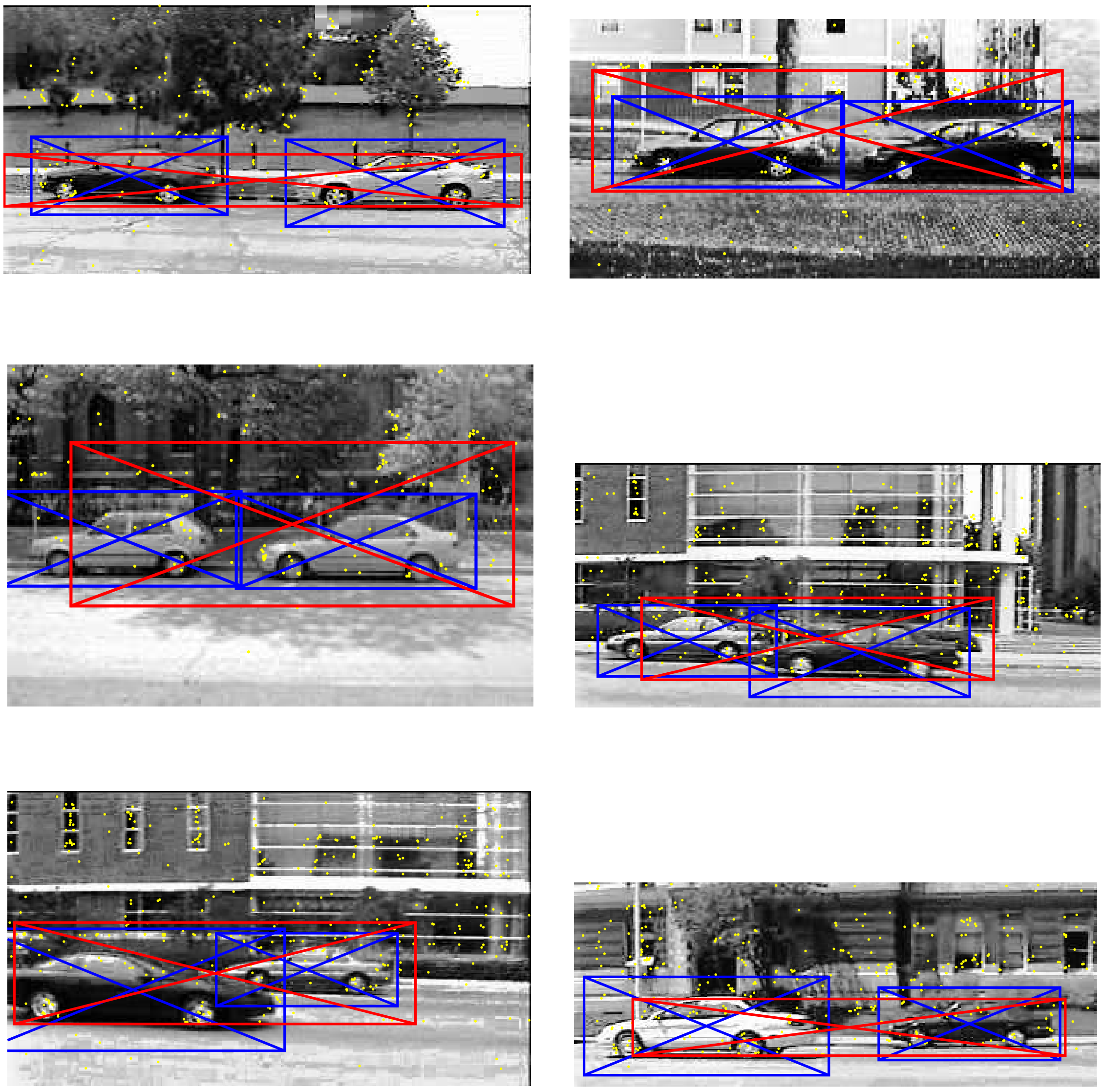


Figure 6.30: Highest-confidence images with incorrect detections for car classifier for GTF as in Table 6.8.

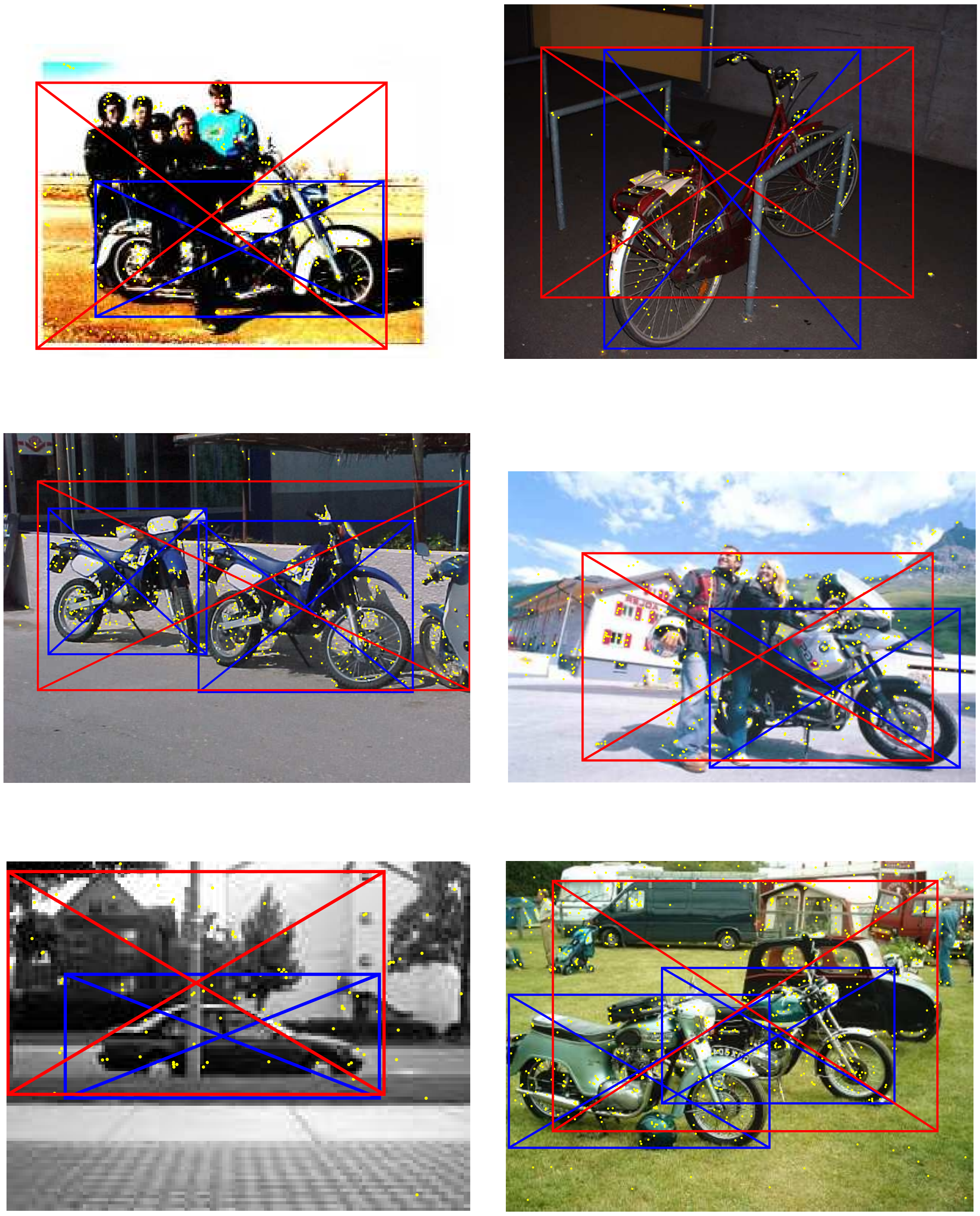


Figure 6.31: Highest-confidence images with incorrect detections for motorbike classifier for GTF as in Table 6.8.

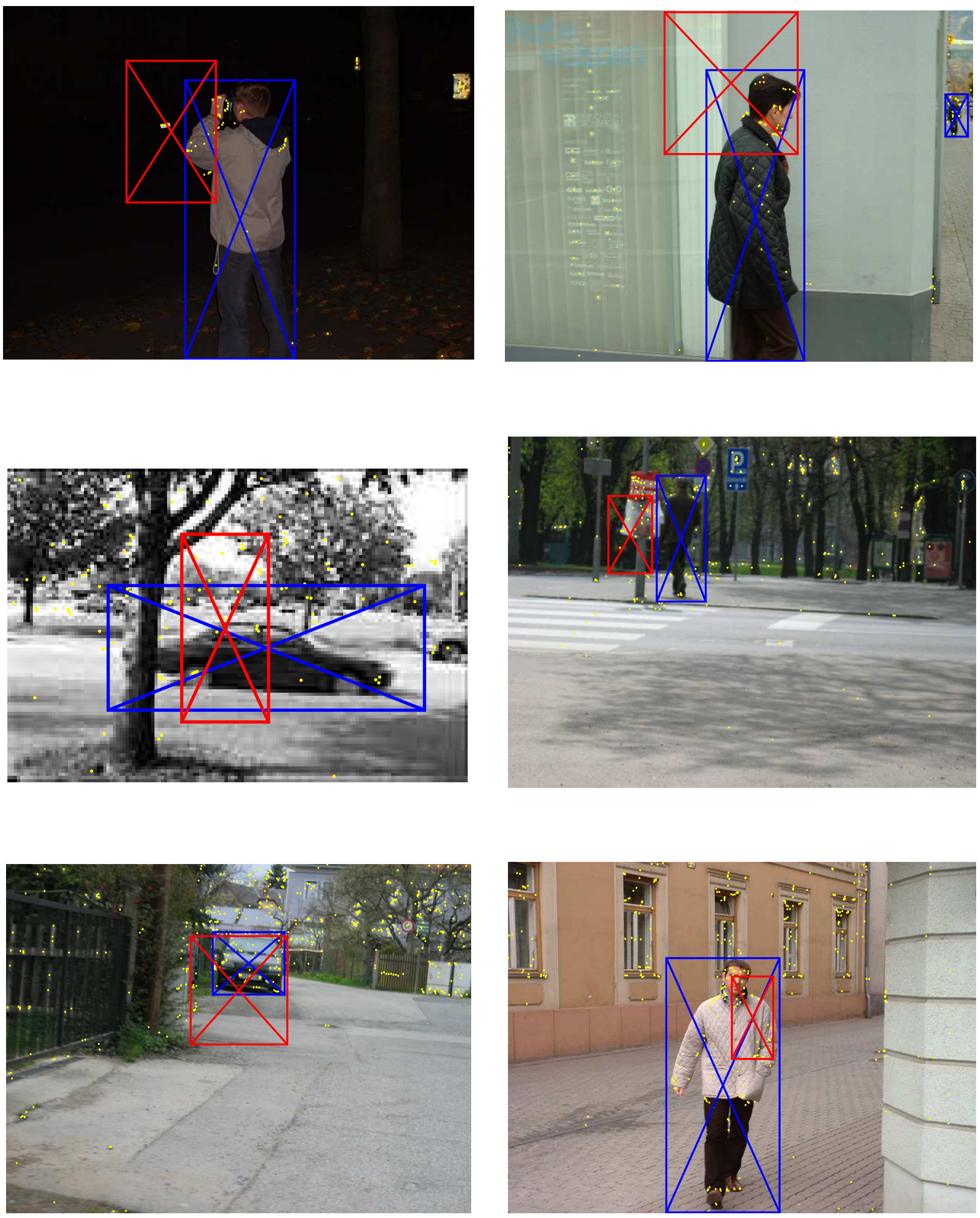


Figure 6.32: Highest-confidence images with incorrect detections for person classifier for GTF as in Table 6.8.

	bicycle	car	motorbike	person
	bicycle	car	motorbike	person
AP	0.547	0.775	0.823	0.345
AP2	0.766	0.781	0.897	0.455
AUC	0.859	0.970	0.929	0.877
AUC2	0.962	0.975	0.992	0.938
recall	114	275	216	84
objects	123	341	220	149
images	114	275	216	84

Table 6.9: Classification-only test data performance evaluation of 8×8 GTF, variance = $\left(\frac{1}{8}\right)^2$, region scale factor = 1, 120 clusters per class.

been made although more of the person is in fact visible in the image.

Table 6.9 shows the evaluation scores obtained when the true object locations are used instead of predicted object locations. This isolates classification from localisation. Since the overall system is not changed, we still only make one object detection per image. The AP and AP2 numbers improve significantly due to the higher recall level from using the true object locations. The AUC and AUC2 numbers are slightly improved. The classes where our detection performance is worse improve more: the scores improve least on motorbikes, and most on people.

Figure 6.33 shows $p(\theta|X_m, W_m)$ for some example images, scanning the object hypothesis centre across the image while keeping the object hypothesis scale fixed at the true object size. The pixel intensities represent the log probability for each location, normalised to use the whole range of intensities from full black to full white. The motorbike example shows a clear distinction between class and non-class features, so that the probability density is a roughly Gaussian blob around the true object location. The bicycle and car examples show significant noise in textured regions of the background, though the true object position can still be clearly seen. The person example has the highest relative probability level in the background, as some of the background texture such as the window shutters on the building match quite well with the person GTF.

Figure 6.34 compares $p(\theta|X_m, W_m)$ under each of the four class GTFs for the bicycle image from Figure 6.33. The four plots are shown using the same intensity scale. The probability mass for the correct object class's GTF is more concentrated than the other classes', as well as

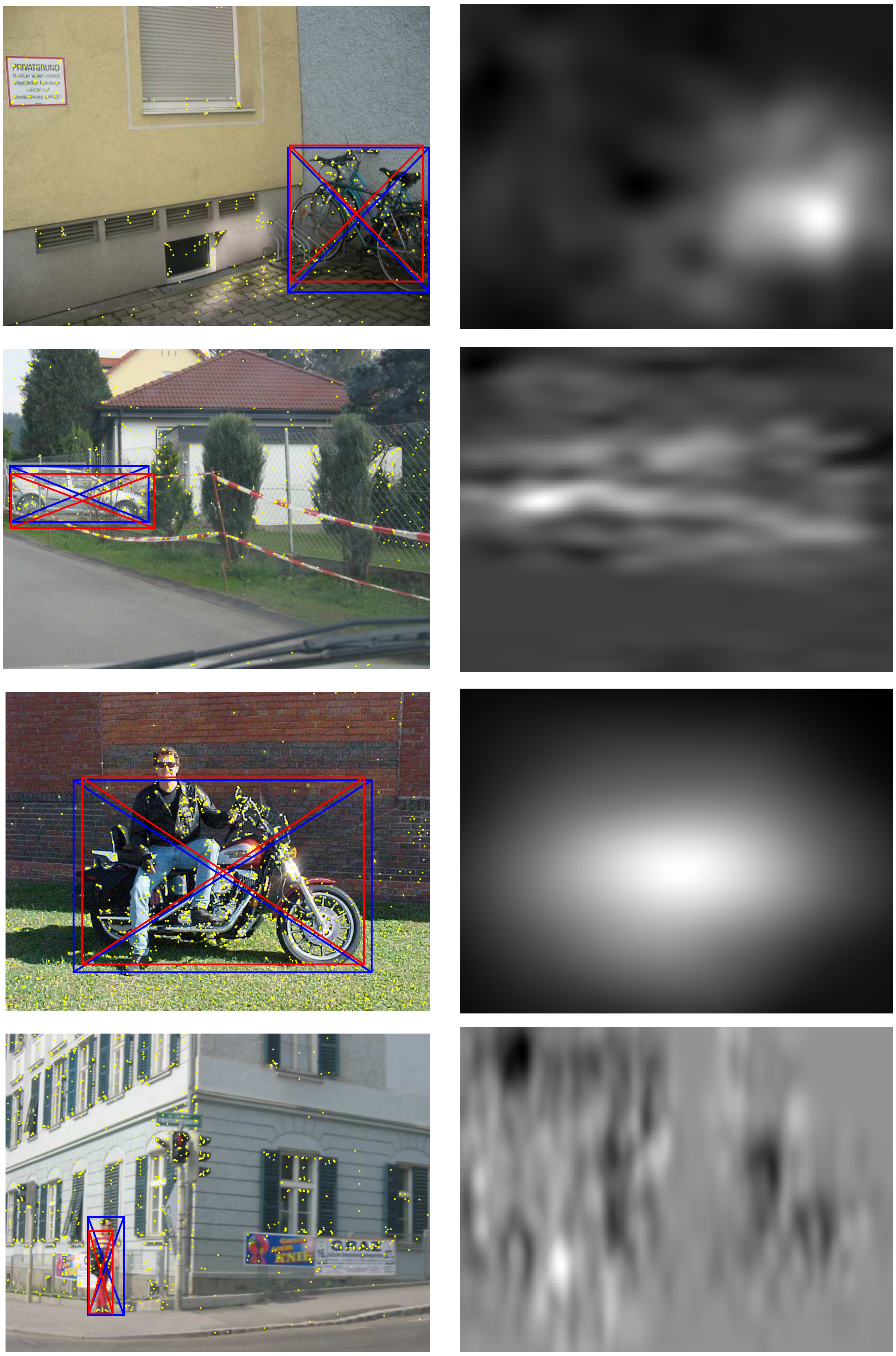


Figure 6.33: Example probability surface for each class GTF as in Table 6.8. Top to bottom: bicycle, car, motorbike, person.

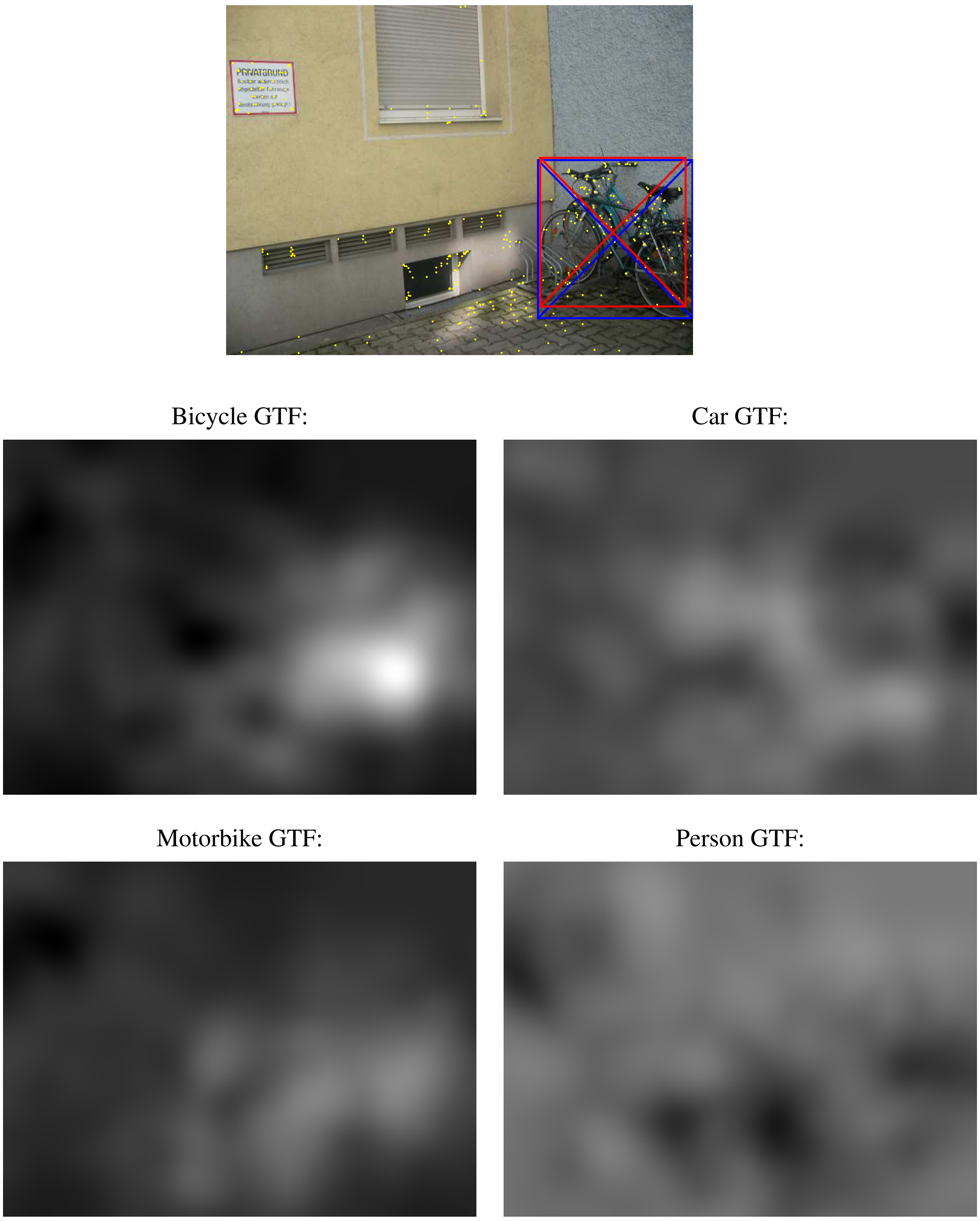


Figure 6.34: Example probability surfaces for bicycle image from Figure 6.33, for each class GTF as in Table 6.8.

this class distribution's peak being higher than the others'.

Figure 6.35 compares $p(\theta|X_m, W_m)$ as the scale of the object hypothesis changes, for the bicycle image as in Figures 6.33 and 6.34. Each plot shows the probability distribution across the image for an object hypothesis based on the true object bounding box's proportions, but scaled by some factor in x and y. The four plots are shown using the same intensity scale. There is a higher probability across the image at the correct object scale than for the smaller or larger object hypotheses. A maximum is visible at the true object location in each of the plots, but in the plots for smaller object hypotheses there are an increasing number of local maxima, as it is easier for the image background to match the learnt GTF when it is examined at smaller scales.

Figure 6.36 shows $p(\theta|X_m, W_m)$ for two example images with multiple objects. The two cars in the first image are in fact clearly visible in the plot of the probability distribution, and multiple maxima are visible for the image with people. This suggests that even without extending the model search to deal with multiple objects, a 'greedy' approach that removed image features responsible for a detection and searched again could find some additional objects and increase performance.

Figure 6.37 compares $p(\theta|X_m, W_m)$ for a number of object hypothesis scales, for the highest-confidence car image with an incorrect detection, the first in Figure 6.30. The top left plot shows the location probability distribution for the scale of the left-hand car, 132×53. As in the example in Figure 6.36, both cars are clearly visible; a lower peak can also be seen midway between the two cars, for a hypothesis which uses the back wheel of the first car and the front wheel of the second. Unlike the example in Figure 6.36, where a correct localisation was made, the global maximum for this image corresponds to a stretched bounding box of 365×35, much wider than, and less tall than the true objects. The plot for this scale is shown at the bottom right. Here the maximum probability location is between the two cars, with the bounding box now including both cars, as shown in Figure 6.30.

Table 6.10 compares the AP and AUC performance of the GTF as in Table 6.8 with the performance achieved by other methods. The Table includes AP and AUC scores for the Darmstadt ISM entry in the Visual Object Classes challenge, and for the best entry for each class in each category, taken from Everingham et al. (2005). The ISM result is included as it is the method

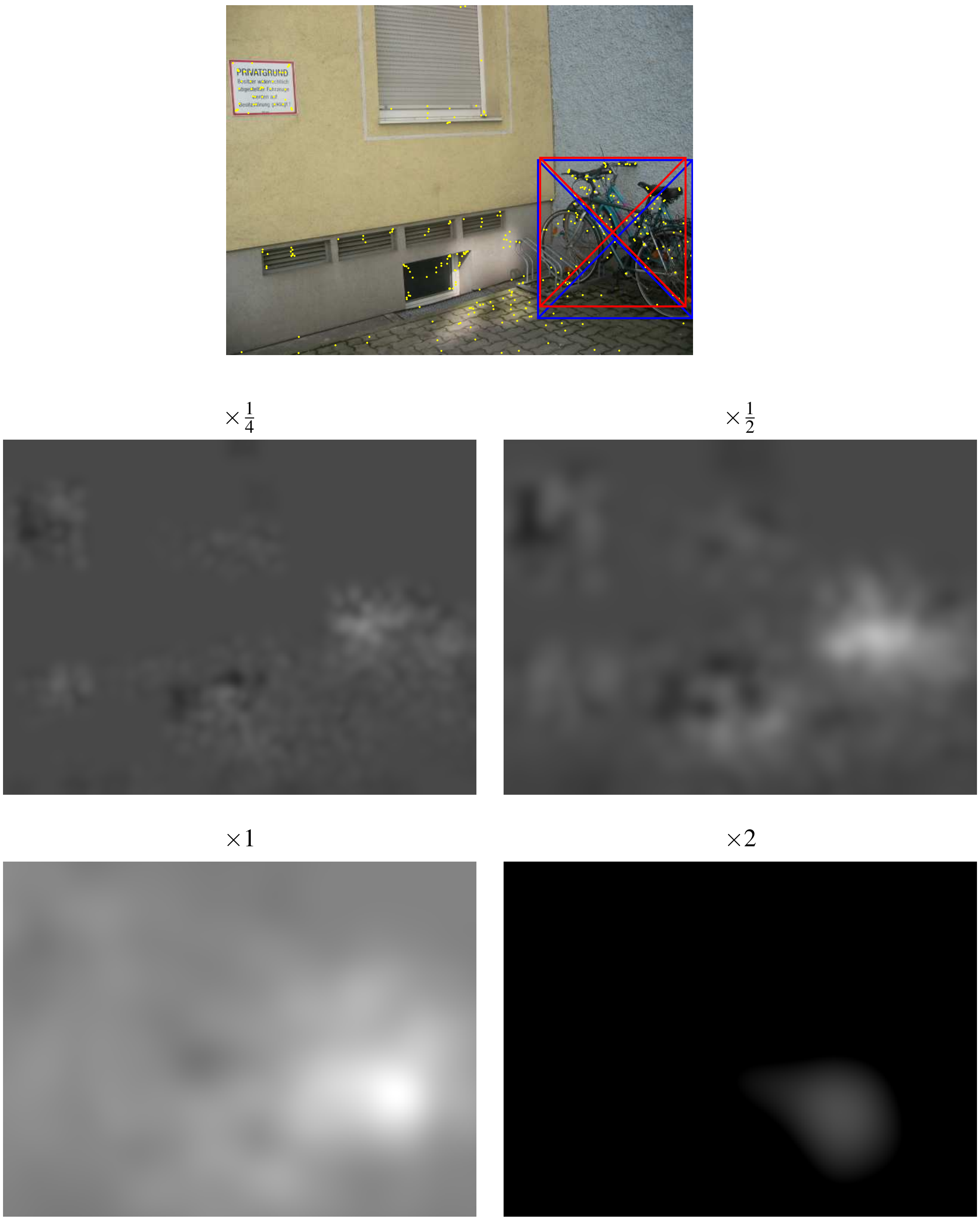


Figure 6.35: Example probability surfaces at various hypothesis scales for bicycle image from Figure 6.33, for GTF as in Table 6.8.

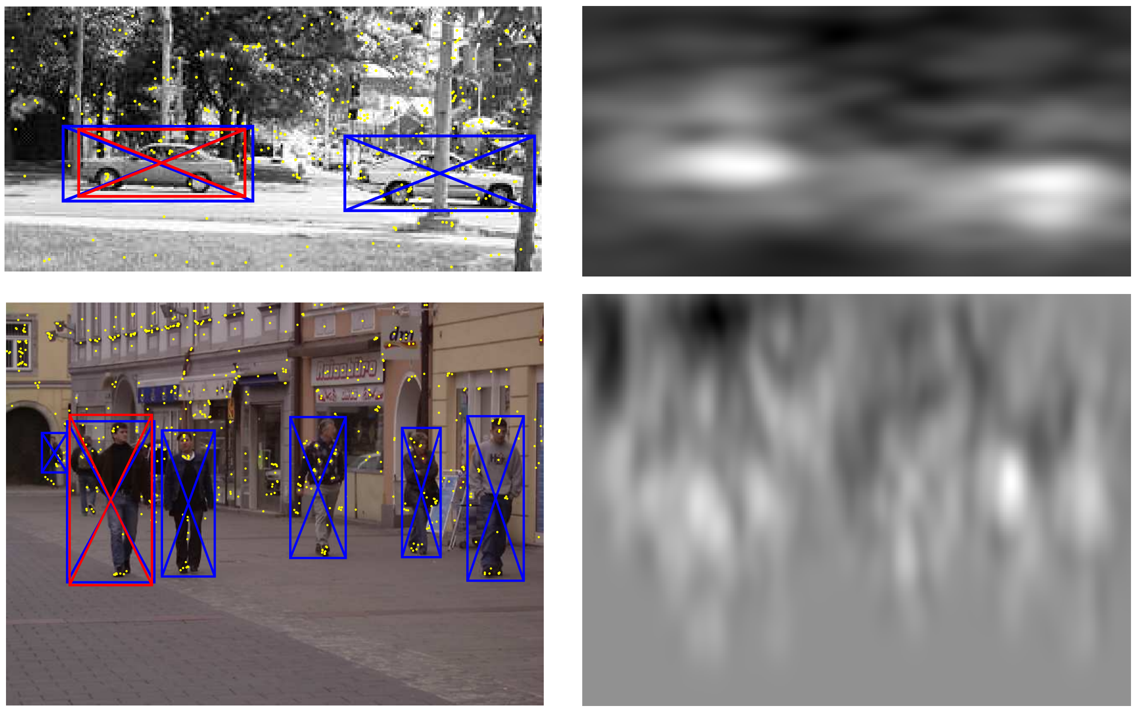
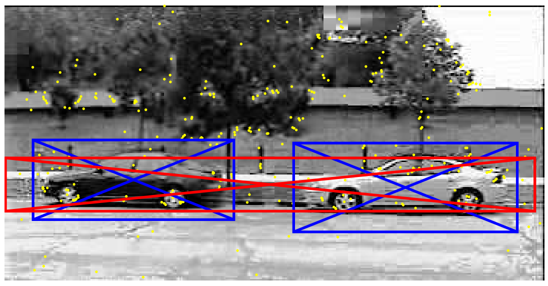


Figure 6.36: Example probability surfaces for car and person GTFs as in Table 6.8.

	bicycle	car	motorbike	person
GTF AP2	0.467	0.422	0.888	0.030
ISM AP	—	0.468	0.865	—
best PASCAL AP	0.119	0.613	0.886	0.013
GTF AUC2	0.966	0.943	0.995	0.890
ISM AUC	—	0.578	0.919	—
best PASCAL AUC	0.982	0.992	0.998	0.979

Table 6.10: Comparison of performance of GTF as in Table 6.8 with other methods' performance.



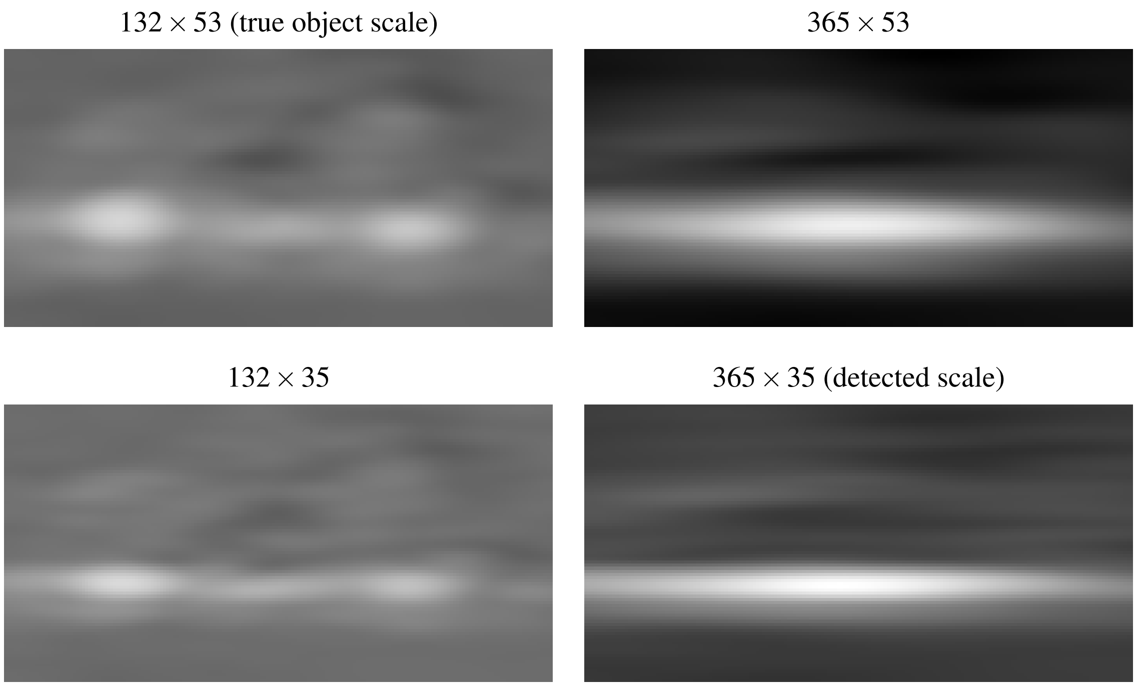


Figure 6.37: Example probability surfaces at various hypothesis scales for highest-probability car image from Figure 6.30, for GTF as in Table 6.8.

in the challenge most similar to the GTF.

For object detection and localisation, as measured by average precision, the GTF beats all the methods from the challenge on the bicycle, motorbike and person classes. On the cars, the class where we lose most from not dealing with multiple objects per image, we have a performance level similar to the ISM.

The GTF is primarily a detection method, and its classification performance, as measured by the area under the ROC curve, is less competitive. It performs better than the ISM, but is beaten by support vector machine-based bag-of-features methods. The GTF parameters we have been using here were chosen primarily to give good localisation performance; further exploration of the GTF parameter space would give improved classification results.

Both the AP and AUC scores here could be improved by optimising the parameters for each class GTF separately, rather than using the same compromise GTF parameters for all four classes, and by optimising the parameters for AUC separately from AP.

Chapter 7

Discussion and future work

7.1 Summary

This section summarises the contributions of this thesis:

- We developed a method for learning object motions from image sequences. The sequence RANSAC algorithm uses invariant features to learn object motions much more quickly than by searching over transformations using pixel information.

- We showed how to combine the sequence RANSAC algorithm with a pixel-based object model to learn appearance models for objects, and analysed the performance of the combined system, showing that it successfully learnt sprites and masks for the objects in example sequences.

- We described the Generative Template of Features model, and showed how it can be used to recognise and localise object classes, from invariant features clustered into 'visual words'. The relationship of the GTF to making predictions in pose-space was explained.

- We examined the behaviour of the GTF in response to parameter changes, and assessed its performance on the PASCAL Visual Object Classes Challenge data, showing that it achieves results competitive with other state-of-the-art methods.

7.2 Future work

This section sets out some directions for future work.

7.2.1 Image features

Using more informative features could give improved performance to both the object learning and the object category recognition methods we have described. Better feature sets might simply consist of more features, might use additional information, or might give feature descriptors better suited to the particular task. While image features need to be invariant across a range of transformations and lighting changes to be useful for sprite learning or object category recognition, it is possible that better performance would be given with a lower degree of invariance than provided by the features used above: for example in our object category recognition data no objects appear upside-down, so matching image regions with opposite orientations may be counterproductive.

An increased number of features slows down processing, but in general provides additional information and allows increased accuracy. An alternative to detecting regions of interest for object category recognition would be to sample image regions densely (see Nowak et al., 2006): even textureless areas of the image may be informative about the presence or absence of an object there.

One possible source of additional information that would almost certainly improve performance for both sprite learning and object category recognition is colour. We have used only greyscale information in feature extraction but, for example, van de Weijer and Schmid (2006) proposed extensions of the SIFT descriptor that use colour as well as texture, and showed that augmenting region descriptors with colour information can make feature matching more reliable.

There are many alternative feature descriptors to the SIFT descriptor we have used in our experiments, such as the shape context descriptor (Mori et al., 2005). It would be possible to optimise the choice of feature descriptor for a particular task, whether by adjusting the parameters of a single descriptor type to improve the performance of the overall system, or by comparing the system's performance with competing descriptor types.

7.2.2 Fast learning of sprites

The sequence RANSAC algorithm could be made more robust by avoiding the need to choose a fixed error threshold to be used in judging whether features belong to the same motion grouping. Instead it might be possible to automatically determine an error threshold for each motion grouping based on the motion variance distribution.

Rather than extracting only a single sprite for each object, we could extract a number of representative sprites, by clustering different views. This would be especially useful for longer sequences of images, where objects are more likely to undergo motion that cannot be adequately modelled as transformations of a single flat sprite.

Individual objects' appearance many vary considerably across scenes. Many objects' internal variability can be viewed in terms of connected 'parts' – for example, the limbs of a person. Modelling parts, rather than having a single geometric model for each object, would allow more complex object behaviour to be dealt with. For example, Felzenszwalb and Huttenlocher (2000, 2005) used 'pictorial structure models' to model the connected parts of objects, such as eyes, noses and the corners of the mouth in faces, representing an object as a collection of parts, related to each other in an acyclic graph. Titsias and Williams (2004) learn parts in an unsupervised manner, in the same way that they previously modelled entire objects, allowing overlapping parts to occlude each other. They demonstrate that their model can learn the mask and appearance for individual limbs from an example video sequence.

7.2.3 Object localisation

The GTF's performance could be improved by learning multiple aspects for each class, rather than combining all views of a class into a single GTF as we do here (see for example Seemann et al., 2006). For example, we could learn separate visual word distributions for front, side, and rear views of cars. It would also be possible to alter the GTF to use a 3D geometric model, either with additional annotation data or with unsupervised learning of the 3D object poses.

It is straightforward to extend the Generative Template of Features model to allow multiple objects. One way to handle multiple objects in a scene is to follow the treatment of Sudderth et al. (2005). They extend θ to hold the instantiation parameters for each object, and define mixing proportions for each object and the background. This approach ignores occlusion, but it would be quite straightforward to use a layered model and to reason about occlusion so as to only generate from visible components. Alternatively, we might expect that individual models could be run to find good regions of θ-space for the given model, and that the robust background model would explain features from other objects. This parallels the work of Williams and Titsias (2004) where such an approach was used to propose good locations for sprite models individually, and a layer ordering was determined in a second pass. Extending the model to allow multiple objects directly makes the search space much larger, but object detection can be sped up by using a greedy approximation to this model: we can start by searching for a single object, then discount image features which have been used in the foreground of the first detection and search again.

It would be interesting to make a direct experimental comparison to investigate how well the approximation we derive in equation 5.16 agrees with equation 5.13 as used in Leibe et al. (2004), using precisely the same feature information in both cases.

The GTF's performance could be improved by using additional contextual information from the rest of the image. The GTF can already use a class-specific background model, but this model is a 'bag of words' within the background area of the image, without any geometric model. This can be compared with, for example, the approach of Murphy et al. (2004) which links object category appearance to a global scene 'gist': particular object types are more likely to be seen in some locations than in others. Hoiem et al. (2006) use a geometric model for the relationship of objects in a scene, considering whether possible object detections in an image are at plausibly-consistent scales based on the distance of their locations below the labelled horizon. Even if we do not have an annotation labelling the horizon, Saxena et al. (2006) demonstrated that reasonable depth-maps can be recovered from single-view images using monocular cues such as the distribution of edge directions.

Finally, it would be possible to combine the GTF with an object learning method to detect and classify moving objects in video, using together cues from how feature motions cluster, from visual word distributions, and from object geometry.

Appendix A

Partial derivatives for GTF model

We can improve an object hypothesis by using gradient ascent to optimise its location and scale $\theta = (t_1, t_2, s_1, s_2)$. This appendix gives the partial derivatives needed to perform this gradient ascent efficiently. Since only one object class model is considered during gradient ascent, we omit the conditioning of the probabilities below on the object class O_j.

For a particular position and scale choice, the joint probability of the object hypothesis, feature locations X_m and the corresponding visual words W_m is as given above:

$$p(\theta, X_m, W_m) = p(\theta) \prod_{i=1}^{N_m} p(\mathbf{x}_{mi}, w_{mi}|\theta) \tag{=5.1}$$

$$p(\mathbf{x}_{mi}, w_{mi}|\theta) = (1-\alpha) p_b(\mathbf{x}_{mi}, w_{mi}) + \alpha p_f(\mathbf{x}_{mi}, w_{mi}|\theta) \tag{=5.2}$$

$$p_f(\mathbf{x}_{mi}, w_{mi}|\theta) = \sum_{z_{mi}=1}^{P} p_f(\mathbf{x}_{mi}|z_{mi}, \theta) p(w_{mi}|z_{mi}) p(z_{mi}) \tag{=5.3}$$

$$p_b(\mathbf{x}_{mi}, w_{mi}|\theta) = \left(\frac{\beta}{A} + \frac{1-\beta}{A - s_1 s_2}\right) p_b(w_{mi}) - \frac{1-\beta}{A - s_1 s_2} s_1 s_2 p_h(\mathbf{x}_{mi}, w_{mi}|\theta) \tag{=5.6}$$

$$= \left(\frac{\beta}{A} + \phi\right) p_b(w_{mi}) - \chi p_h(\mathbf{x}_{mi}, w_{mi}|\theta) \tag{A.1}$$

$$p_h(\mathbf{x}_{mi}, w_{mi}|\theta) = \sum_{z_{mi}=1}^{P} p_f(\mathbf{x}_{mi}|z_{mi}, \theta) p_b(w_{mi}) p(z_{mi}) \tag{=5.7}$$

with $\phi = \frac{1-\beta}{A - s_1 s_2}$, $\chi = \frac{1-\beta}{A - s_1 s_2} s_1 s_2$.

It is straightforward to calculate the partial derivatives of $p(\mathbf{x}_{mi}, w_{mi}|\theta)$:

$$\begin{aligned} \nabla_\theta(p(\mathbf{x}_{mi}, w_{mi}|\theta)) = &\nabla_\theta(\alpha) p_f(\mathbf{x}_{mi}, w_{mi}|\theta) + \alpha \nabla_\theta(p_f(\mathbf{x}_{mi}, w_{mi}|\theta)) \\ &- \nabla_\theta(\alpha) p_b(\mathbf{x}_{mi}, w_{mi}|\theta) + (1-\alpha) \nabla_\theta(p_b(\mathbf{x}_{mi}, w_{mi}|\theta)) \end{aligned} \tag{A.2}$$

$$\nabla_\theta(p_f(\mathbf{x}_{mi},w_{mi}|\theta)) = \sum_{z_{mi}=1}^{P} p(w_{mi}|z_{mi})p(z_{mi})\nabla_\theta(p(\mathbf{x}_{mi}|z_{mi},\theta)) \tag{A.3}$$

$$\begin{aligned}\nabla_\theta(p_b(\mathbf{x}_{mi},w_{mi}|\theta)) =& \nabla_\theta(\phi)p_b(w_{mi}) \\ & -\nabla_\theta(\chi)p_h(\mathbf{x}_{mi},w_{mi}|\theta)-\chi\nabla_\theta(p_h(\mathbf{x}_{mi},w_{mi}|\theta))\end{aligned} \tag{A.4}$$

$$\nabla_\theta(\phi) = \begin{pmatrix} 0 \\ 0 \\ \frac{(1-\beta)s_2}{(A-s_1s_2)^2} \\ \frac{(1-\beta)s_1}{(A-s_1s_2)^2} \end{pmatrix} \tag{A.5}$$

$$\nabla_\theta(\chi) = A\nabla_\theta(\phi) \tag{A.6}$$

$$\nabla_\theta(p_h(\mathbf{x}_{mi},w_{mi}|\theta)) = \sum_{z_{mi}=1}^{P} p_b(w_{mi})p(z_{mi})\nabla_\theta(p(\mathbf{x}_{mi}|z_{mi},\theta)). \tag{A.7}$$

Each GTF component part, with position $\mathbf{u}_{z_{mi}} = \begin{pmatrix} u_{1z_{mi}} \\ u_{2z_{mi}} \end{pmatrix}$ relative to the template centre, generates feature locations from a Gaussian $F(\theta) = p(\mathbf{x}_{mi}|z_{mi},\theta)$, with standard deviation $\sigma = \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}$:

$$F(\theta) = \frac{1}{2\pi\sigma_1\sigma_2 s_1 s_2}\exp\left(\frac{-(x_{i1}-s_1u_{1z_{mi}}-t_1)^2}{2(\sigma_1 s_1)^2} - \frac{(x_{i2}-s_2u_{2z_{mi}}-t_2)^2}{2(\sigma_2 s_2)^2}\right) \tag{A.8}$$

$$\frac{\partial F(\theta)}{\partial t_1} = \frac{x_{i1}-s_1u_{1z_{mi}}-t_1}{(\sigma_1 s_1)^2}F(\theta) \tag{A.9}$$

$$\frac{\partial F(\theta)}{\partial t_2} = \frac{x_{i2}-s_2u_{2z_{mi}}-t_2}{(\sigma_2 s_2)^2}F(\theta) \tag{A.10}$$

$$\frac{\partial F(\theta)}{\partial s_1} = \frac{(x_{i1}-s_1u_{1z_{mi}}-t_1)^2+s_1u_{1z_{mi}}(x_{i1}-s_1u_{1z_{mi}}-t_1)-(\sigma_1 s_1)^2}{(\sigma_1 s_1)^2 s_1}F(\theta) \tag{A.11}$$

$$\frac{\partial F(\theta)}{\partial s_2} = \frac{(x_{i2}-s_2u_{2z_{mi}}-t_2)^2+s_2u_{1z_{mi}}(x_{i2}-s_2u_{2z_{mi}}-t_2)-(\sigma_2 s_2)^2}{(\sigma_2 s_2)^2 s_2}F(\theta). \tag{A.12}$$

The second partial derivatives are as follows:

$$\begin{aligned}\frac{\partial^2 p(\mathbf{x}_{mi},w_{mi}|\theta)}{\partial\theta_j\theta_k} =& \frac{\partial^2\alpha}{\partial\theta_j\theta_k}p_f(\mathbf{x}_{mi},w_{mi}|\theta)+\frac{\partial\alpha}{\partial\theta_j}\frac{\partial p_f(\mathbf{x}_{mi},w_{mi}|\theta)}{\partial\theta_k} \\ & +\frac{\partial\alpha}{\partial\theta_k}\frac{\partial p_f(\mathbf{x}_{mi},w_{mi}|\theta)}{\partial\theta_j}+\alpha\frac{\partial^2 p_f(\mathbf{x}_{mi},w_{mi}|\theta)}{\partial\theta_j\theta_k} \\ & -\frac{\partial^2\alpha}{\partial\theta_j\theta_k}p_b(\mathbf{x}_{mi},w_{mi}|\theta)-\frac{\partial\alpha}{\partial\theta_j}\frac{\partial p_b(\mathbf{x}_{mi},w_{mi}|\theta)}{\partial\theta_k} \\ & -\frac{\partial\alpha}{\partial\theta_k}\frac{\partial p_b(\mathbf{x}_{mi},w_{mi}|\theta)}{\partial\theta_j}+(1-\alpha)\frac{\partial^2 p_b(\mathbf{x}_{mi},w_{mi}|\theta)}{\partial\theta_j\theta_k}\end{aligned} \tag{A.13}$$

$$H_\theta\left(p_f(\mathbf{x}_{mi}, w_{mi}|\theta)\right) = \sum_{z_{mi}=1}^{P} p(w_{mi}|z_{mi})p(z_{mi})H_\theta\left(p(\mathbf{x}_{mi}|z_{mi}, \theta)\right) \tag{A.14}$$

$$\begin{aligned}\frac{\partial^2 p_b(\mathbf{x}_{mi}, w_{mi}|\theta)}{\partial\theta_j\theta_k} =& \frac{\partial^2\phi}{\partial\theta_j\theta_k}p_b w_{mi} - \frac{\partial^2\chi}{\partial\theta_j\theta_k}p_h(\mathbf{x}_{mi}, w_{mi}|\theta) \\ & - \frac{\partial\chi}{\partial\theta_j}\frac{\partial p_h(\mathbf{x}_{mi}, w_{mi}|\theta)}{\partial\theta_k} + \frac{\partial\chi}{\partial\theta_k}\frac{\partial p_h(\mathbf{x}_{mi}, w_{mi}|\theta)}{\partial\theta_j} \\ & + \chi p_h(\mathbf{x}_{mi}, w_{mi}|\theta)\end{aligned} \tag{A.15}$$

$$H_\theta(\phi) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{2(1-\beta)s_2^2}{(A-s_1s_2)^3} & \frac{(1-\beta)(s_1s_2+A)}{(A-s_1s_2)^3} \\ 0 & 0 & \frac{(1-\beta)(s_1s_2+A)}{(A-s_1s_2)^3} & \frac{2(1-\beta)s_2^2}{(A-s_1s_2)^3} \end{pmatrix} \tag{A.16}$$

$$H_\theta(\chi) = AH_\theta(\chi) \tag{A.17}$$

$$H_\theta\left(p_h(\mathbf{x}_{mi}, w_{mi}|\theta)\right) = \sum_{z_{mi}=1}^{P} p_b(w_{mi})p(z_{mi})H_\theta\left(p(\mathbf{x}_{mi}|z_{mi}, \theta)\right) \tag{A.18}$$

$$\frac{\partial^2 F(\theta)}{\partial t_1^2} = \frac{x_{i1} - s_1u_{1z_{mi}} - t_1}{(\sigma_1 s_1)^2}\frac{\partial F(\theta)}{\partial t_1} - \frac{1}{(\sigma_1 s_1)^2}F(\theta) \tag{A.19}$$

$$\frac{\partial^2 F(\theta)}{\partial t_1 t_2} = \frac{x_{i1} - s_1u_{1z_{mi}} - t_1}{(\sigma_1 s_1)^2}\frac{\partial F(\theta)}{\partial t_2} \tag{A.20}$$

$$\begin{aligned}\frac{\partial^2 F(\theta)}{\partial t_1 s_1} =& \frac{x_{i1} - s_1u_{1z_{mi}} - t_1}{(\sigma_1 s_1)^2}\frac{\partial F(\theta)}{\partial s_1} \\ & - \frac{2(x_{i1} - s_1u_{1z_{mi}} - t_1) + s_1u_{1z_{mi}}}{(\sigma_1 s_1)^2 s_1}F(\theta)\end{aligned} \tag{A.21}$$

$$\frac{\partial^2 F(\theta)}{\partial t_1 s_2} = \frac{x_{i1} - s_1u_{1z_{mi}} - t_1}{(\sigma_1 s_1)^2}\frac{\partial F(\theta)}{\partial s_2} \tag{A.22}$$

$$\frac{\partial^2 F(\theta)}{\partial t_2 t_1} = \frac{\partial^2}{\partial t_1 t_2} \quad \text{(as the matrix of partial derivatives is symmetrical)} \tag{A.23}$$

$$\frac{\partial^2 F(\theta)}{\partial t_2^2} = \frac{x_{i2} - s_2u_{2z_{mi}} - t_2}{(\sigma_2 s_2)^2}\frac{\partial F(\theta)}{\partial t_2} - \frac{1}{(\sigma_2 s_2)^2}F(\theta) \tag{A.24}$$

$$\frac{\partial^2 F(\theta)}{\partial t_2 s_1} = \frac{x_{i2} - s_2u_{2z_{mi}} - t_2}{(\sigma_2 s_2)^2}\frac{\partial F(\theta)}{\partial s_1} \tag{A.25}$$

$$\begin{aligned}\frac{\partial^2 F(\theta)}{\partial t_2 s_2} =& \frac{x_{i2} - s_2u_{2z_{mi}} - t_2}{(\sigma_2 s_2)^2}\frac{\partial F(\theta)}{\partial s_2} \\ & - \frac{2(x_{i2} - s_2u_{2z_{mi}} - t_2) - s_2u_{1z_{mi}}}{(\sigma_2 s_2)^2 s_2}F(\theta)\end{aligned} \tag{A.26}$$

$$\frac{\partial^2 F(\theta)}{\partial s_1 t_1} = \frac{\partial^2}{\partial t_1 s_1} \tag{A.27}$$

$$\frac{\partial^2 F(\theta)}{\partial s_1 t_2} = \frac{\partial^2}{\partial t_2 s_1} \tag{A.28}$$

$$\frac{\partial^2 F(\theta)}{\partial s_1^2} = \frac{(x_{i1} - s_1u_{1z_{mi}} - t_1)^2 + s_1u_{1z_{mi}}(x_{i1} - s_1u_{1z_{mi}} - t_1) - (\sigma_1 s_1)^2}{(\sigma_1 s_1)^2 s_1}\frac{\partial F(\theta)}{\partial s_1}$$

$$+\left((\sigma_1 s_1)^2 + 2(x_{i1} - s_1 u_{1z_{mi}} - t_1 + s_1 u_{1z_{mi}}) s_1 u_{1z_{mi}}\right) F(\theta)$$
$$-\frac{3(x_{i1} - s_1 u_{1z_{mi}} - t_1 + s_1 u_{1z_{mi}})^2}{(\sigma_1 s_1)^2 s_1^2} F(\theta) \tag{A.29}$$

$$\frac{\partial^2 F(\theta)}{\partial s_1 s_2} = \frac{(x_{i1} - s_1 u_{1z_{mi}} - t_1)^2 + s_1 u_{1z_{mi}} x_{i1} - s_1 u_{1z_{mi}} - t_1 - (\sigma_1 s_1)^2}{(\sigma_1 s_1)^2 s_1} \frac{\partial F(\theta)}{\partial s_2} \tag{A.30}$$

$$\frac{\partial^2 F(\theta)}{\partial s_2 t_1} = \frac{\partial^2}{\partial t_1 t_2} \tag{A.31}$$

$$\frac{\partial^2 F(\theta)}{\partial s_2 t_2} = \frac{\partial^2}{\partial t_2 s_2} \tag{A.32}$$

$$\frac{\partial^2 F(\theta)}{\partial t_2 s_1} = \frac{\partial^2}{\partial s_1 s_2} \tag{A.33}$$

$$\frac{\partial^2 F(\theta)}{\partial t_2 s_2} = \frac{(x_{i2} - s_2 u_{2z_{mi}} - t_2)^2 + s_2 u_{1z_{mi}} x_{i2} - s_2 u_{2z_{mi}} - t_2 - (\sigma_2 s_2)^2}{(\sigma_2 s_2)^2 s_2} \frac{\partial F(\theta)}{\partial s_2}$$
$$+\left((\sigma_2 s_2)^2 + 2(x_{i2} - s_2 u_{2z_{mi}} - t_2 + s_2 u_{1z_{mi}}) s_2 u_{1z_{mi}}\right) F(\theta)$$
$$-\frac{3(x_{i2} - s_2 u_{2z_{mi}} - t_2 + s_2 u_{1z_{mi}})^2}{(\sigma_2 s_2)^2 s_2^2} F(\theta). \tag{A.34}$$

The mixing proportion α is modelled as a function of the object scale as in equation 5.9, so

$$\nabla_\theta(\alpha) = \begin{pmatrix} 0 \\ 0 \\ \gamma\frac{s_2}{A} \\ \gamma\frac{s_1}{A} \end{pmatrix} \tag{A.35}$$

$$H_\theta(\alpha) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\gamma}{A} \\ 0 & 0 & \frac{\gamma}{A} & 0 \end{pmatrix}. \tag{A.36}$$

In practice we work with $\log p(\theta, X_m, W_m)$ and use

$$\frac{\partial}{\partial\theta_j}\log f(\theta) = \frac{\frac{\partial f(\theta)}{\partial\theta_i}}{f(\theta)} \tag{A.37}$$

$$\frac{\partial^2}{\partial\theta_i\theta_j}\log f(\theta) = \frac{\frac{\partial^2 f(\theta)}{\partial\theta_i\theta_j} f(\theta) - \left(\frac{\partial f(\theta)}{\partial\theta_i}\right)^2}{f(\theta)^2} \tag{A.38}$$

$$= \frac{\frac{\partial^2 f(\theta)}{\partial\theta_i\theta_j}}{f(\theta)} - \left(\frac{\frac{\partial f(\theta)}{\partial\theta_i})}{f(\theta)}\right)^2. \tag{A.39}$$

To optimise the full joint probability $p(\theta, X_m, W_m)$ we also need the partial derivatives of the scale model given in equation 5.8:

$$\nabla_{\mathbf{s}}(\log P(\mathbf{s})) = \begin{pmatrix} \frac{\Sigma_{11}(\log s_1 - \mu_1) + \Sigma_{12}(\log s_2 - \mu_2)}{s_1} \\ \frac{\Sigma_{21}(\log s_1 - \mu_1) + \Sigma_{22}(\log s_2 - \mu_2)}{s_2} \end{pmatrix} \tag{A.40}$$

$$H_{\mathbf{s}}(\log P(\mathbf{s})) = \begin{pmatrix} \frac{\Sigma_{11}(\log s_1 - \mu_1 - 1) + \Sigma_{12}(\log s_1 - \mu_1)}{s_1} & \frac{-\Sigma_{12}}{s_1 s_2} \\ \frac{-\Sigma_{21}}{s_1 s_2} & \frac{\Sigma_{21}(\log s_1 - \mu_1) + \Sigma_{22}(\log s_2 - \mu_2 - 1)}{s_2} \end{pmatrix}. \tag{A.41}$$

Bibliography

S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26 (11):1475–1490, 2004.

Moray Allan, Michalis K. Titsias, and Christopher K. I. Williams. Fast learning of sprites using invariant features. In *Proceedings of the British Machine Vision Conference*, pages 40–49, 2005.

N. E. Apostoloff and A.W. Fitzgibbon. Bayesian video matting using learnt image priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 407–414, 2004.

D. H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

Matthew J. Beal, Nebojsa Jojic, and Hagai Attias. A graphical model for audiovisual object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):828–836, July 2003.

Michael J. Black and P. Anandan. A framework for the robust estimation of optic flow. In *Proceedings of the International Conference on Computer Vision*, pages 231–236, 1993.

Michael J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, January 1996.

A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive GMMRF model. In *Proceedings of the European Conference on Computer Vision*, pages 428–441, 2004.

Matthew Brown and David Lowe. Invariant features from interest point groups. In *Proceedings of the British Machine Vision Conference*, pages 656–665, 2002.

Michael C. Burl, Markus Weber, and Pietro Perona. A probabilistic approach to object recognition using local photometry and global geometry. In *Proceedings of the European Conference on Computer Vision*, volume 2, pages 628–641, 1998.

Michael T. Chan, Anthony Hoogs, Rahul Bhotika, and Amitha Perera. Joint recognition of complex events and track matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1615–1622, 2006.

Olivier Chapelle, Patrick Haffner, and Vladimir Vapnik. SVMs for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, October 1999.

T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *Proceedings of the European Conference on Computer Vision*, volume 2, pages 484–498, 1998.

João Paulo Costeira and Takeo Kanade. A multi-body factorization method for motion analysis. In *Proceedings of the International Conference on Computer Vision*, pages 1071–1076, 1995.

David Crandall, Pedro Felzenszwalb, and Daniel Huttenlocher. Spatial priors for part-based recognition using statistical models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 10–17, 2005.

David J. Crandall and Daniel P. Huttenlocher. Weakly supervised learning of part-based spatial models for visual object recognition. In *Proceedings of the European Conference on Computer Vision*, pages 16–29, 2006.

Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, at ECCV*, 2004.

Thomas Deselaers, Daniel Keysers, and Hermann Ney. Discriminative training for object recognition using image patches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 157–162, 2005.

G. Dorko and C. Schmid. Object class recognition using discriminative local features. Technical Report RR-5497, INRIA Rhône Alpes, 2005.

M. Everingham, A. Zisserman, C. Williams, L. Van Gool, M. Allan, C. Bishop, O. Chapelle, N. Dalal, T. Deselaers, G. Dorko, S. Duffner, J. Eichhorn, J. Farquhar, M. Fritz, C. Garcia, T. Griffiths, F. Jurie, D. Keysers, M. Koskela, J. Laaksonen, D. Larlus, B. Leibe, H. Meng, H. Ney, B. Schiele, C. Schmid, E. Seemann, J. Shawe-Taylor, A. Storkey, S. Szedmak, B. Triggs, I. Ulusoy, V. Viitaniemi, and J. Zhang. *Selected Proceedings of the First PASCAL Challenges Workshop*, chapter on the Pascal Visual Object Classes Challenge. LNAI. Springer-Verlag, April 2005.

L. Fei-Fei, R. Fergus, and P. Perona. A Bayesian approach to unsupervised one-shot learning of object categories. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1134–1141, 2003.

P. F. Felzenszwalb and D. P. Huttenlocher. Efficient matching of pictorial structures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 66–73, 2000.

P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

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

R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from Google's image search. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1816–1823, 2005a.

R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 380–387, 2005b.

Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach.* Prentice Hall, Upper Saddle River, New Jersey, 2003.

B. J. Frey and N. Jojic. Transformation Invariant Clustering Using the EM Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intellligence*, 25(1):1–17, 2003.

Mario Fritz, Bastian Leibe, Barbaba Caputo, and Bernt Schiele. Integrating representative and discriminant models for object category detection. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1363–1370, 2005.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

A.H. Gee and R. Cipolla. Fast visual tracking by temporal consensus. Technical Report CUED/F-INFENG/TR 207, Cambridge University Department of Engineering, February 1995.

R. Ginhoux and J. Gutmann. Model-based object tracking using stereo vision. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1226–1232, 2001.

Amara Graps. An introduction to wavelets. *IEEE Computational Sciences and Engineering*, 2 (2):50–61, 1995.

W. E. L. Grimson. *Object Recognition by Computer.* MIT Press, Cambridge, MA, 1990.

Amit Gruber and Yair Weiss. Multibody factorization with uncertainty and missing data using the EM algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 707–714, 2004.

Scott Helmer and David G. Lowe. Object class recognition with many local features. In *Workshop on Generative Model Based Vision, at CVPR 2004*, 2004.

Derek Hoiem, Alexei A. Efros, and Martial Hebert. Putting objects in perspective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2137–2144, 2006.

P. V. C. Hough. Methods and means for recognizing complex patterns, December 1962. U.S. patent 3069654.

Michal Irani, Benny Rousso, and Shmuel Peleg. Computing occluding and transparent motions. *International Journal of Computer Vision*, 12(1):5–16, January 1994.

N. Jojic and B. Frey. Learning flexible sprites in video layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 199–206, 2001.

Nebojsa Jojic, Brendan J. Frey, and Anitha Kannan. Epitomic analysis of appearance and shape. In *Proceedings of the International Conference on Computer Vision*, volume 1, pages 34–41, 2003.

Nebojsa Jojic, John Winn, and Larry Zitnick. Escaping local minima through hierarchical model selection: Automatic object discovery, segmentation, and tracking in video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 117–124, 2006.

Michael J. Jones and Paul Viola. Face recognition using boosted local features. Technical Report TR2003-25, Mitsubishi Electric Research Laboratories, April 2003.

Timor Kadir and Michael Brady. Scale, saliency and image description. *International Journal of Computer Vision*, 45(2):83–105, November 2001.

Anitha Kannan, Nebojsa Jojic, and Brendan Frey. Generative model for layers of appearance and deformation. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, pages 166–173, 2005.

M. P. Kumar, P. H. S. Torr, and A. Zisserman. Learning layered pictorial structures from video. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, pages 158–163, 2004.

M. P. Kumar, P. H. S. Torr, and A. Zisserman. Learning layered motion segmentations of video. In *Proceedings of the International Conference on Computer Vision*, volume 1, pages 33–40, 2005.

Jorma Laaksonen, Markus Koskela, and Erkki Oja. PicSOM—self-organizing image retrieval with MPEG-7 content descriptors. *IEEE Transactions on Neural Networks*, 13(4):841–8, 2002.

Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems 2 (NIPS 1989)*, pages 396–404, 1990.

Bastian Leibe and Bernt Schiele. Scale-invariant object categorization using a scale-adaptive mean-shift search. In *Proceedings of the DAGM Symposium on Pattern Recognition*, pages 145–153, August 2004.

Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an Implicit Shape Model. In *Workshop on Statistical Learning in Computer Vision, at ECCV 2004*, pages 17–32, May 2004.

Tony Lindeberg. Scale-space theory: A basic tool for analysing structures at different levels. *Journal of Applied Statistics*, 21(2):225–270, 1994. Supplement on Advances in Applied Statistics: Statistics and Images: 2.

David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999.

David G. Lowe. Local feature view clustering for 3D object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 682–688, 2001.

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

J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. *Computer Vision and Image Understanding*, 22(10):761–767, 2004.

K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proceedings of the European Conference on Computer Vision*, pages 128–142, 2002.

K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

K. Mikolajczyk, T. Tuytelaars, C. Schmid, and A. Zisserman. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1/2):43–72, 2005.

Thomas P. Minka. The ‘summation hack’ as an outlier model. URL `http://research.microsoft.com/~minka/papers/minka-summation.pdf`. Technical note, August 2003.

N. Morgan and H. A. Bourlard. Neural networks for statistical recognition of continuous speech. *Proceedings of the IEEE*, 83(5):742–770, 1995.

Greg Mori, Serge Belongie, and Jitendra Malik. Efficient shape matching using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1832–1837, November 2005.

Kevin Murphy, Antonio Torralba, and William Freeman. Using the forest to see the trees: A graphical model relating features, objects and scenes. In *Advances in Neural Information Processing Systems 16 (NIPS 2003)*, 2004.

Eric Nowak, Frédéric Jurie, and Bill Triggs. Sampling strategies for bag-of-features image classification. In *Proceedings of the European Conference on Computer Vision*, pages 490–503, 2006.

Till Quack, Vittorio Ferrari, and Luc Van Gool. Video mining with frequent itemset configurations. In *Proceedings of the 4th International Conference on Image and Video Retrieval (CIVR 2005)*, pages 360–369, 2006.

M. Revow, C. K. I. Williams, and G. E. Hinton. Using generative models for handwritten digit recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6): 592–606, 1996.

Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Segmenting, modeling, and matching video clips containing multiple moving objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 914–921, 2004.

S. Rowe and A. Blake. Statistical background modelling for tracking with a virtual camera. In *Proceedings of the British Machine Vision Conference*, volume 2, pages 423–432, 1995.

Harpreet S. Sawhney and Serge Ayer. Compact representations of videos through dominant and multiple motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):814–830, 1996.

Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. Learning depth from single monocular images. In *Advances in Neural Information Processing Systems 18 (NIPS 2005)*, pages 1161–1168, 2006.

F. Schaffalitzky and A. Zisserman. Automated scene matching in movies. In *Proceedings of the Challenge of Image and Video Retrieval*, pages 186–197, 2002.

Edgar Seemann, Bastian Leibe, and Bernt Schiele. Multi-aspect detection of articulated objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1582–1588, 2006.

Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 1–15, 2006.

Josef Sivic, Frederik Schaffalitzky, and Andrew Zisserman. Object level grouping for video shots. In *Proceedings of the European Conference on Computer Vision*, pages 724–734, 2004.

Josef Sivic, Bryan C. Russell, Alexei A. Efros, Andrew Zisserman, and William T. Freeman. Discovering objects and their location in images. In *Proceedings of the International Conference on Computer Vision*, volume 1, pages 370–377, 2005.

R. S. Stephens. Probabilistic approach to the Hough transform. *Image and Vision Computing*, 9(1):66–71, 1991.

Erik B. Sudderth, Antonio Torralba, William T. Freeman, and Alan S. Willsky. Learning hierarchical models of scenes, objects, and parts. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1331–1338, 2005.

Michalis K. Titsias and Christopher K. I. Williams. Fast unsupervised learning of multiple objects and parts from video. In *Workshop on Generative-Model Based Vision, at CVPR 2004*, July 2004.

Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 762–769, 2004.

M. Turk and A. Pentland. Face recognition using eigenfaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991.

Régis Vaillant, Christophe Monrocq, and Yann Le Cun. An original approach for the localisation of objects in images. *IEE Proceedings on Vision, Image, and Signal Processing*, 141 (4):245–250, August 1994.

Joost van de Weijer and Cordelia Schmid. Coloring local feature extraction. In *Proceedings of the European Conference on Computer Vision*, pages 334–348, 2006.

René Vidal and Richard Hartley. Motion segmentation with missing data using PowerFactorization and GPCA. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 310–316, 2004.

René Vidal and Shankar Sastry. Optimal segmentation of dynamic scenes from two perspective views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 281–286, 2003.

P. A. Viola and M. J. Jones. Robust Real-time Face Detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

Gang Wang, Ye Zhang, and Li Fei-Fei. Using dependent regions for object categorization in a generative framework. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1597–1604, 2006.

J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, 1994.

M. Weber, M. Welling, and P. Perona. Towards automatic discovery of object categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 101–108, 2000a.

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

Y. Weiss and E. H. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 321–326, 1996.

Yair Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 520–527, 1997.

Y. Wexler, A. Fitzgibbon, and A. Zisserman. Bayesian estimation of layers from multiple images. In *Proceedings of the European Conference on Computer Vision*, pages 487–501, 2002.

C. K. I. Williams and M. K. Titsias. Greedy learning of multiple objects in images using robust statisics and factorial learning. *Neural Computation*, 16(5):1039–1062, 2004.

Christopher K. I. Williams and Moray Allan. On a connection between object localization with a generative template of features and pose-space prediction methods. Technical Report EDI-INF-RR-0719, Informatics Research Report, University of Edinburgh, January 2006.

Christopher K. I. Williams and Michalis K. Titsias. Learning about multiple objects in images: Factorial learning without factorial search. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, pages 1391–1398, 2003.

Josh Wills, Sameer Agarwal, and Serge Belongie. What went where. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 37–44, 2003.

John Winn and Andrew Blake. Generative affine localisation and tracking. In *Advances in Neural Information Processing Systems 17 (NIPS 2004)*, pages 1505–1512, 2005.

Jiangjian Xiao and Mubarak Shah. Accurate motion layer segmentation and matting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 698–703, 2005.