Online Structured Learning for Real-Time Computer Vision Gaming Applications

Sam Hare

Thesis submitted in partial fulfilment of the requirements of the award of

Doctor of Philosophy

Oxford Brookes University

in collaboration with Sony Computer Entertainment Europe

2012

Abstract

In recent years computer vision has played an increasingly important role in the development of computer games, and it now features as one of the core technologies for many gaming platforms. The work in this thesis addresses three problems in real-time computer vision, all of which are motivated by their potential application to computer games.

We first present an approach for real-time 2D tracking of arbitrary objects. In common with recent research in this area we incorporate online learning to provide an appearance model which is able to adapt to the target object and its surrounding background during tracking. However, our approach moves beyond the standard framework of tracking using binary classification and instead integrates tracking and learning in a more principled way through the use of structured learning. As well as providing a more powerful framework for adaptive visual object tracking, our approach also outperforms state-of-the-art tracking algorithms on standard datasets.

Next we consider the task of keypoint-based object tracking. We take the traditional pipeline of matching keypoints followed by geometric verification and show how this can be embedded into a structured learning framework in order to provide principled adaptivity to a given environment. We also propose an approximation method allowing us to take advantage of recently developed binary image descriptors, meaning our approach is suitable for real-time application even on low-powered portable devices. Experimentally, we clearly see the benefit that online adaptation using structured learning can bring to this problem.

Finally, we present an approach for approximately recovering the dense 3D structure of a scene which has been mapped by a simultaneous localisation and mapping system. Our approach is guided by the constraints of the low-powered portable hardware we are targeting, and we develop a system which coarsely models the scene using a small number of planes. To achieve this, we frame the task as a structured prediction problem and introduce online learning into our approach to provide adaptivity to a given scene. This allows us to use relatively simple multi-view information coupled with online learning of appearance to efficiently produce coarse reconstructions of a scene.

Acknowledgements

The work in this thesis would not have been possible without the help and support of a large number of people, to whom I am extremely grateful.

I would like to thank my supervisor Phil Torr whose guidance, insight and energy managed to keep me going throughout the highs and lows of the past few years. I would equally like to thank Diarmid Campbell and Amir Saffari, my supervisory team at Sony Computer Entertainment Europe, for all of their patience and effort on my behalf. Thanks also go to my examiners Roberto Cipolla and Nigel Crook for their careful evaluation of this research.

My home during the course of this work was the Vision R&D group of Sony Computer Entertainment Europe, and I would like to thank my colleagues Henrik Bäärnhielm, Patrick Bricout, Ben Bradford, Graham Clemo, Simon Hall, Nilay Kothari, Nick Lord, Tom Lucas-Woodley, Antonio Martini, James Oates, Andrew Stoddart and Will Sykes for their support and for all of the fun times.

The members of the Oxford Brookes Vision Group who overlapped with me provided many interesting discussions about research and life in general, and I would like to thank Karteek Alahari, Fabio Cuzzolin, David Jarzebowski, Lubor Ladicky, Jon Rihan, Gregory Rogez, Chris Russell, Michael Sapienza, Sunando Sengupta, Glenn Sheasby, Paul Sturgess, Julien Valentin, Vibhav Vineet, Jonathan Warrell, Ziming Zhang and Kyle Zheng.

I am deeply grateful to my parents John Hare and Lisa Jardine for their support during all of my life's endeavours. I am particularly indebted to Lisa for her guidance during the writing-up phase of this thesis.

Above all I would like to thank Yuli Takatsuki for her support, love and encouragement which gave me the determination required to complete this research. There is no way I would have been able to do it without her.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This thesis addresses a number of real-time computer vision problems, all of which are motivated by their potential application to computer games. The work we present has been carried out as part of a collaboration between academia and industry and has therefore been influenced by factors from both of these fields. Throughout this thesis, the desire has been to produce results which are both academically interesting and rigorous, and which also lay the groundwork for useful real-world applications of computer vision.

1.1 Motivation

In recent years, computer vision has played an increasingly important role in the development of computer games, and it now features as part of the core technology for many gaming platforms. Aside from the obvious factors contributing to this such as the availability of cheaper camera hardware and more powerful processors, there have been two major factors affecting the development of computer games which have placed increasing emphasis on computer vision.

The first of these is the shift towards casual gaming, which aims to make more accessible and social games which can be played by non-expert users. Many of these are 'physical' games, meaning a user interacts with them using their body, rather than having to learn less intuitive button presses on a traditional controller. Besides being accessible to non-expert users, physical games have proved popular for computer game publishers wishing to change the image of gaming as an anti-social, unhealthy pastime.

The second factor is the rise in mobile gaming, caused by an explosion in the number of portable devices such as smartphones and tablets, which means that mobile gaming is no longer restricted to users who choose to carry around a dedicated gaming device. The fact that most current smartphones, tablets and portable games consoles also include a camera provides opportunities for computer vision to be used in games for these devices.

The work in this thesis is motivated by both of these factors, and our contributions fall into two broad categories:

- **Human-computer interaction**. The work in Chapter 3 deals with tracking, which is motivated by the need to track the face of a player interacting with a camera-based game. By tracking the player, games can be developed which take their input directly from the player's physical movements, providing a more intuitive form of human-computer interaction than would be available using a traditional controller.

- **Augmented reality (AR)**. Mobile devices provide an excellent platform for augmented reality, as it feels both natural and magical to hold them up as a 'window' to the world through which a user can see a modified version of reality. The work in Chapter 4 deals with the task of providing robust detection and tracking of an object in 3D, which is an essential core component of an AR system. Chapter 5 deals with the higher-level task of trying to infer the real-world structure of a scene, which can be used to enhance the AR experience and provide a platform for more compelling games.

1.2 Challenges

Being driven by applications for computer games means that certain important challenges had to be taken into account when developing the approaches in this thesis.

1.2.1 Diverse environments

A key factor when developing vision algorithms for use in computer games is that they are expected to be deployed to a large audience in a wide variety of environments. This principle has guided much of the work in this thesis, and a common theme is that algorithms should incorporate an element of *adaptability* to a given environment.

The approaches we develop all incorporate machine learning at their core, in common with much of the recent research across the entire field of computer

vision. Importantly, building on these well-studied and principled techniques from the machine learning community provides us with a natural mechanism for incorporating adaptability into our algorithms: *online learning*. Significant progress has been made by the machine learning community in recent years in order to handle the vast, distributed datasets which arise from an increasingly digital and connected world. Traditional learning approaches which require access to all training data at once are being superseded by those which are able to learn incrementally using only portions of the dataset and, in the extreme case, using only individual training examples. In this thesis we take advantage of this progress in order to provide adaptability to diverse environments.

1.2.2 Computational constraints

The other significant factor which must be considered in our setting is computational cost and, in particular, the desire for algorithms to be *real-time*. This is of course an imprecise term, but in general the algorithms developed in this thesis are designed to be run interactively as frames are received from a camera. This requirement places fundamental constraints on the types of approaches which can be developed and affects the way that success is measured. Such real-time requirements have even more significant consequences when targeting portable devices. While the portable computing revolution has been made possible in large part by the development of more powerful and efficient processors, these devices still possess only a fraction of the computational power of a typical desktop computer. The computational constraints placed on the vision algorithms are compounded by the fact that in practice not all of the available processing power is available, since it is also necessary to run a game on the same device. The goal is therefore to produce algorithms which achieve acceptable accuracy, whilst using as little processing resources as possible.

We again find that we are able to benefit from progress in online machine learning in order to work within these constraints. Because these learning algorithms are intended to work with extremely large datasets, they too must be designed to be as computationally efficient as possible. Often, this is achieved

using the philosophy of the 'unreasonable effectiveness of data' [50], which states that simple learning algorithms trained with large quantities of data often outperform more sophisticated and expensive learning algorithms trained with smaller quantities of data. Using these simple learning algorithms, we are able to produce algorithms which achieve our goal of providing adaptability in a principled way, whilst still remaining computationally efficient, even for low-powered devices.

1.3 Contributions

In this thesis we address three problems in real-time computer vision. These problems have been chosen because they have been identified as being useful from an industrial perspective, in that they can provide the building blocks for vision-based computer games. The approaches which we develop make use of recent progress in online machine learning, and in particular structured learning, in order to tackle these problems in a principled academic manner.

1.3.1 2D object tracking

Chapter 3 presents a novel approach for 2D tracking of arbitrary objects. In common with recent research in visual object tracking we incorporate online learning to provide an appearance model which is able to adapt to the target object and its surrounding background during tracking. However, our approach moves beyond the standard framework of treating tracking as a binary classification problem and instead integrates tracking and learning in a more principled way through the use of structured learning. We use a structured output support vector machine (SVM) to perform learning, and in order to allow for real-time application we also introduce a budgeting mechanism which constrains the computational cost of our approach. As well as providing a more powerful framework for adaptive visual object tracking, our approach also outperforms state-of-the-art tracking algorithms on standard datasets.

1.3.2 Keypoint-based object tracking

Chapter 4 deals with the task of keypoint-based object tracking, which is a core component required for AR applications. We take the traditional pipeline for this task of matching keypoints using image descriptors followed by geometric verification using random sampling and show how this can be embedded into a structured learning framework in order to provide adaptivity to a given environment. Similarly to the work of Chapter 3, the use of structured learning allows tracking and learning to be tightly integrated in a principled way. We also propose an approximation method allowing us to take advantage of recently developed binary image descriptors, meaning our approach is suitable for real-time application even on low-powered portable devices. Experimentally, we clearly see the benefit that online adaptation using learning can bring to this problem.

1.3.3 Scene reconstruction

Chapter 5 continues the theme of AR on low-powered devices from Chapter 4 and presents an approach for approximately recovering the dense 3D structure of a scene which has been mapped by a simultaneous localisation and mapping (SLAM) system. Our approach is guided by the constraints of the hardware we are targeting, and we develop a system which coarsely models the scene using a small number of planes. In common with the work in other chapters, we frame the task as a structured prediction problem and introduce online learning into our approach. This allows us to use relatively simple multi-view information coupled with online learning of appearance to efficiently produce reconstructions of a scene which are useful from a gaming perspective.

1.4 Publications

The work presented in Chapter 3 first appeared in:

- S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured Output Tracking with Kernels. In *IEEE International Conference on Computer Vision*, 2011.

The work presented in Chapter 4 first appeared in:

- S. Hare, A. Saffari, P. H. S. Torr. Efficient Online Structured Output Learning for Keypoint-Based Object Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

Both of these publications can be found as appendices to this thesis.

Chapter 2

Background and Related Work

In this chapter we provide an overview of the background material relevant to the work in this thesis. The first three sections focus on the computer vision application areas tackled in Chapters 3-5, while the fourth section focuses on structured learning, which features at the core of all the approaches developed in this thesis.

2.1 Object tracking

Object tracking aims to estimate the motion of a target object between successive frames of a video sequence. This is a fundamental problem in computer vision, and a great deal of prior research exists in the area. The interested reader is directed to [27] for a thorough survey of the field, while this section summarises those approaches which are most relevant to the work in this thesis.

All tracking algorithms require some kind of *representation* of the target object. Possible choices for this include points [76, 107], bounding box, contour [12, 56] and articulated structures [23, 94]. The choice of representation in turn determines the *state* of the tracker, which is what must be estimated in each video frame. For the work presented in Chapter 3 of this thesis, we consider only a bounding box representation (Figure 2.1). The advantage of such a representation is that the state is simple, consisting only of 2D translation and possibly rotation and scale. However, in the physical world we expect the target object to undergo deformations, out-of-plane motion and be affected by partial occlusions and lighting changes, none of which are handled explicitly by this representation. Consequently, these factors must be handled by an *appearance model*, which should encode the variability in the object as it appears within the

Figure 2.1: An example object tracking task using a bounding box representation. Notice the target object undergoes significant appearance changes, which must be handled by the tracking approach.

bounding box.

The appearance model is the primary differentiator between approaches which make use of a bounding box representation and can broadly be divided into two categories: generative and discriminative.

2.1.1 Generative appearance models

Generative approaches involve some kind of model which is able to capture the way the target object appears inside the bounding box during tracking.

The simplest approach for modelling the appearance of target object is with a single template image, for example the image inside the bounding box at the start of tracking. Tracking can then be performed by registering this template image with each subsequent video frame by maximising a similarity function, based on *e.g.* sum of squared differences (SSD) or normalised cross-correlation (NCC). To perform this maximisation, one approach is to use exhaustive local search around the previous tracker state. Although this is very straightforward, it is also computationally expensive. A more efficient approach is to assume that the similarity function is locally smooth and perform gradient-based optimisation [8,11,76]. This smoothness assumption may only be valid in a very local area, so in order to handle greater motion between frames, coarse-to-fine optimisation on an image pyramid can be used [18].

Tracking with a single template image suffers from robustness issues in practice, since it is does not provide sufficient tolerance to the changes in appearance which are expected during tracking. Various extensions to this approach have been proposed in order to improve robustness by incorporating illumination invariance [49], robustness to partial occlusion [57], and multiple appearance modalities [13].

A strength of template based approaches is that they are able to provide very accurate estimates of object state (Figure 2.2), and the mathematics they are based on makes it straightforward to extend the classes of transformations which are supported. But because of the way in which they model an object in terms of individual pixels which must be aligned exactly with pixels in a new frame, they

Figure 2.2: ... approach is able to track ... homography. Image court...

have little t... ther fragile.

An alter... reat it as a probability ... [30]. Tracking is then ... hm to iteratively max... the current tracker distribution. The strength of this approach is that it is far more tolerant to slight changes in object appearance, since it is essentially tracking a blob of colour (Figure 2.3). The associated weakness, however, is that discarding all spatial information reduces the discriminative power of the appearance model, meaning the tracker provides less accurate estimates of object state and may become confused by background regions with a similar colour distribution. Another issue with this approach is that it is less straightforward to extend tracking to include parameters such as scale and rotation, which are simple to include in a template tracking framework.

Figure 2.3: Mean-shift tracking using a colour distribution [30]. This approach is able to track under significant appearance change, since the colour distribution remains roughly the same. Image courtesy of D. Comaniciu [30].

572 IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 25, NO. 5, MAY 2003

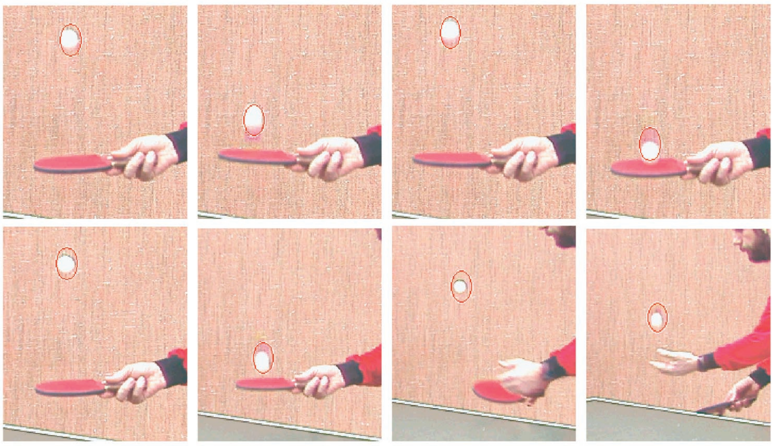


Fig. 8. *Ball* sequence. The frames 2, 12–16, 26, 32, 40, 48, and 51 are shown.

attraction, while the probabilities of the colors that are part of the background are considerably reduced. The ball is reliably tracked over the entire sequence of 60 frames.

The last example is also taken from the *Football* sequence. This time the head and shoulder of player number 59 is tracked (Fig. 9). Note the changes in target appearance along the entire sequence and the rapid movements of the target.

6.2 Kalman Prediction

It was already mentioned in Section 1 that the Kalman filter assumes that the noise sequences $\mathbf{v}_k$ and $\mathbf{n}_k$ are Gaussian and the functions $\mathbf{f}_k$ and $\mathbf{h}_k$ are linear. The dynamic equation becomes $\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{v}_k$, while the measurement equation is $\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{n}_k$. The matrix $\mathbf{F}$ is called the system matrix and $\mathbf{H}$ is the measurement matrix. As in the general case, the Kalman filter solves the state estimation problem in two steps: prediction and update. For more details, see [5, p. 56].

The kernel-based target localization method was integrated with the Kalman filtering framework. For a faster implementation, two independent trackers were defined for horizontal and vertical movement. A constant-velocity dynamic model with acceleration affected by white noise [5, p. 82] has been assumed. The uncertainty of the measurements has been estimated according to [55]. The idea is to normalize the similarity surface and represent it as a probability density function. Since the similarity surface is smooth, for each filter only three measurements are taken into account, one at the convergence point (peak of the surface) and the other two at a distance equal to half of the target dimension, measured from the peak. We fit a scaled Gaussian to the three points and compute the measurement uncertainty as the standard deviation of the fitted Gaussian.

A first set of tracking results incorporating the Kalman filter is presented in Fig. 10 for the 120 frames *Hand* sequence where the dynamic model is assumed to be affected by a noise

Fig. 9. *Football* sequence, tracking player number 59. The frames 70, 96, 108, 127, 140, and 147 are shown.

These issues have been addressed by various authors, and in particular the approach of Elgammal *et al.* [39] along with the related approach of Yang *et al.* [130] reintroduce spatial information into the mean-shift framework and perform tracking in so-called joint feature-spatial spaces. Another related approach for introducing spatial information into a histogram-based object model is that of Adam *et al.* [2], which divides the target object into multiple sub-regions, with tracking performed by robustly combining the results of tracking individual sub-regions.

2.1.1.1 Incorporating adaptability

The approaches described so far do not incorporate any notion of adaptability of the appearance model, as the template image or histogram remains fixed during tracking. Such adaptability is often essential in practice to handle changes in object appearance caused by object deformation and changing environmental conditions. One simple strategy for adapting the appearance model is to replace it each frame, discarding the previous model. This approach is very aggressive, however, as it does not maintain any history about the appearance of the object in previous frames. As a result, it is prone to *drift*, since small tracking errors will accumulate over time and ultimately result in tracking failure. One approach for dealing with this was proposed by Mattews *et al.* [79], which updates the template only when it is considered safe to do so and retains the original template to prevent drift (Figure 2.4). Other approaches maintain more sophisticated appearance models which summarise the appearance of the object over time and adapt gradually to changes, such as the incremental PCA approach of Ross *et al.* [95] and the WSL tracker of Jepson *et al.*. [57].

2.1.2 Discriminative appearance models

More recent tracking research has focused around appearance models which are discriminative, meaning that rather than capturing the appearance of the target object alone, they model the differences between the appearance of the target object and its surrounding background. Such approaches have benefited greatly

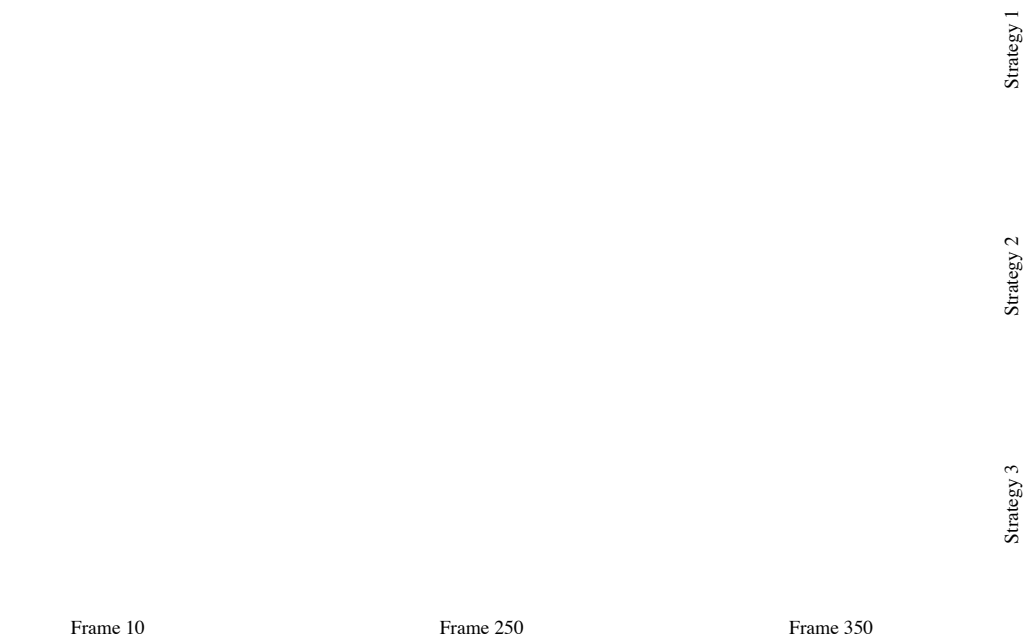


Figure 2.4: The template update problem. In the first row a fixed template is used, and tracking is lost as the target object changes appearance due to lighting. In the second row the template is updated every frame, which results in tracking drift because the template drifts. The final row uses an approach which combines both fixed and updating templates to result in successful tracking. Image courtesy of I. Matthews [79].

Next, we consider the more general case of template tracking with linear appearance variation. Specifically, we generalize our algorithm to AAMs [Cootes *et al*., 2001]. In this context, our appearance update algorithm can also be interpreted as a heuristic to avoid local minima and so we again quantitatively evaluate it as such. We also demonstrate how our algorithm can be applied to convert a generic person-independent AAM into a person specific AAM.

2 Single Template Tracking

We begin by considering the original template tracking problem [Lucas and Kanade, 1981] where the object is represented by a single template image. Suppose we are given a video sequence of images $I_n(\mathbf{x})$ where $\mathbf{x} = (x, y)^{\mathrm{T}}$ are the pixel coordinates and $n = 0, 1, 2, \ldots$ is the frame number. In template tracking, a subregion of the initial frame $I_0(\mathbf{x})$ that contains the object of interest is

2.1.2.1 Incorporating adaptability

2

Since discriminative approaches incorporate information about the target object and its background, providing a mechanism for adaptability becomes particularly important. While object detection research has shown that it is possible to use

large training sets containing 'typical' background examples to train object detectors, during tracking we will only be discriminating between the object and a particular background. It is therefore desirable to use an appearance model which is specific to this background. Since most discriminative appearance models are based around classifiers, online learning provides a natural mechanism for achieving this, providing the tracker with the ability to adapt both to changes in the object appearance, as well as changes in the surrounding background. After initialising the classifier at the start of tracking (*e.g.* with a user-specified bounding box, or the output from an object detector), approaches designed in this way operate in two stages. First the existing classifier is used to update the state of the tracker, and then the new state of the tracker is used in order to update the classifier (Figure 2.5).

Figure 2.5: Online learning for tracking. The classifier confidence function is used to update the state of the tracker, after which the classifier is updated by generating training samples from the new tracker state. Image courtesy of H. Grabner [46].

An influential approach based on these ideas was the Online Boosting method proposed by Grabner *et al.* [46]. In this method, a boosting-based [102] classifier similar to that proposed for object detection by Viola and Jones [123] is learned online [86]. To update the classifier each frame, a set of labelled training ex-

amples is generated using the current tracker state. The image patch inside the current tracker bounding box is treated as a positive training example, while image patches inside a number of randomly-selected bounding boxes from the local area (some of which overlap with the tracker bounding box) are treated as negative examples. The boosting-based classifier is able to select from a large pool of image features to best discriminate between the positive and negative examples.

This approach results in a powerful tracking framework, which under the right conditions is able to track arbitrary objects in complex backgrounds, whilst handling the various changes in appearance which cause problems for most tracking approaches. It still suffers from a number of drawbacks, however, which subsequent research has attempted to address.

The first problem is that of label noise. Because the tracker state will inevitably contain some errors, the training examples given to the classifier may not be labelled correctly. If the classifier cannot handle this noise, its ability to discriminate between the target object and its background will suffer, causing tracking quality to decline. Boosting is known to suffer from label noise [37], since it can overfit to samples which are not well predicted by the current classifier, so one approach is to make use of robust loss functions for boosting to provide more tolerance to label noise [70, 77]. Alternatively, different classifiers such as random forests [22, 101] which have better robustness to label noise may be employed. A particularly successful approach for handling labelling noise was proposed by Babenko *et al.* [7], who make use of Multiple Instance Learning (MIL) [36] to allow the classifier to select from a number of potential positive examples, according to its current state (Figure 2.6). This method was shown to provide significant improvements over the original Online Boosting tracker.

The second problem is the reliance of classification-based approaches on self-training, whereby the result of the tracker is always assumed to be correct and then used to update the classifier. Fundamentally, this is a problem with all adaptive tracking methods, since the only true supervision comes from the first frame of tracking (*i.e.* when the tracker is initialised). Given the framework of adaptive tracking using a discriminative classifier, however, a number of approaches have been proposed to try and mitigate the danger of self-training. One such approach

Figure 2.6: Multiple instance learning for handling label noise. The first column updates the classifier using a single positive and multiple negative examples, which may result in drift as the positive example is mis-aligned. The second column updates the classifier using multiple positive and negative examples, which may also result in drift as it is harder for the classifier to discriminate between the two classes. The final column uses multiple instance learning to allow the classifier to select for itself which example should be treated as positive based on its current state. Image courtesy of B. Babenko [7].

was proposed by Grabner *et al.* [47], which makes use of semi-supervised learning and treats all examples after the initial frame as unlabelled. This approach retains a classifier learned from the initial frame, which is used to anchor future updates during tracking so that significant drift cannot take place. In practice, however, this approach can suffer from a lack of adaptability to appearance change, which can also lead to poor tracking performance.

Fundamentally, there is a dilemma which must be faced when performing adaptive object tracking. On one hand, allowing too much adaptation of the appearance model can lead to drift and ultimately tracking failure. On the other hand, if the adaptation is constrained in order to prevent drift, the appearance model may not be able to handle the changes in object appearance, which will also lead to tracking failure. Recently, attempts have been made to resolve this dilemma by incorporating higher-level reasoning about the scene into the tracking framework [60, 99], which appears to be a promising research direction.

2.2 Keypoint-based object detection

Keypoint-based object detection is a widely-used approach for detecting instances of a specific textured target object in an image. Its robustness and efficiency mean that it forms the cornerstone of many computer vision applications such as augmented reality (AR) and simultaneous localisation and mapping (SLAM).

The target object is modelled as a collection of distinctive *keypoints*, each consisting of location and local appearance information. These keypoints are designed to be easy to identify when the object is observed in a given image. Detecting the object in an input image then follows a standard pipeline consisting of three stages: detecting keypoints in an input image; finding potential matches between image keypoints and model keypoints; and geometric verification of matches to determine overall object presence and geometric transformation (Figure 2.7).

The strengths of these approaches are twofold: firstly, they are able to detect an object under a large class of geometric and photometric transformations. This is possible because individual keypoints describe only local information about an object, allowing methods to be designed which are locally tolerant to such transformations. Secondly, they incorporate a great deal of redundancy, since

Figure 2.7: Keypoint-based object detection. A planar target object is shown on the left, and potential matches are found between object and image keypoints (brighter lines indicate higher matching scores). Geometric verification is then used to determine the homography transformation between object and image.

geometric verification provides a very strong cue for object detection. This means that detection requires only a subset of keypoints to be successfully matched, making these methods robust against partial occlusions and matching failures.

There exists a great deal of prior research related to each of the stages of the detection pipeline. In this section we provide a brief overview of some of the key literature.

2.2.1 Keypoint detection

In order to identify keypoints in an image, a detector is required. The aim is for a detector to have high *repeatability*, meaning it can reliably detect the same keypoint even as an image undergoes various geometric and photometric transformations. To achieve this repeatability, the detector is designed with invariances to certain transformations. In theory, if the local image data around the keypoint is transformed in a way which is handled by the detector, it will still be detected. In practice there are inevitably artefacts introduced by the imaging process such as aliasing and noise which can violate the assumptions made by the detector, but this is compensated by the redundancy which comes from modelling the object as a collection of such keypoints.

An early example of a keypoint detector was proposed by Harris [52], which uses the eigenvalues of a 2×2 matrix built from local image gradient information around each pixel in an image to identify stable *corners*. A related extension was proposed by Shi and Tomasi [107], which under certain assumptions results in more stable corners. These detectors are able to provide invariance to translation and rotation of the image.

To provide additional invariance to scaling of the image, a number of subsequent approaches have been proposed which make use of image *scale-space* [73]. This image representation adds a third dimension corresponding to the scale of a Gaussian kernel with which the image is convolved. *Blobs* can then be identified in scale-space by searching for extrema of the Laplacian of Gaussian (LoG) [74] or Difference of Gaussian (DoG) [75] operators. In particular, the DoG detector was introduced to accompany the well-known Scale-Invariant Feature Transform

(a) Input image **(b)** DoG keypoints **(c)** FAST keypoints

Figure 2.8: Examples of DoG [75] and FAST [96] keypoints. The DoG detector identifies multi-scale blobs, while the FAST detector identifies single-scale corners.

(SIFT) [75] descriptor and so is widely used in practice. A similar approach for blob detection uses the Determinant of Hessian (DoH) [74] operator, and a fast approximation of this method is used as the detector for the widely-used Speeded Up Robust Feature (SURF) [10] descriptor.

Detectors have also been proposed to handle general affine (translation, rotation, scaling and shearing) transformations of the image. Some of these are based on affine extensions to scale-space approaches [80, 121], while others identify different image features which are stable under affine transformations, such as Maximally Stable Extremal Regions (MSERs) [78].

Whilst the development of keypoint detectors with invariance to a large class of transformations is important, in practice an equally important consideration is computational cost, especially where we are interested in real-time applications. Consequently, one of the most commonly-used keypoint detectors in practice is the Features from Accelerated Segment Test (FAST) detector [96]. This approach aims to detect only single-scale corners, but focuses on doing so very efficiently. Corners are identified by scanning a ring of 16 pixels around a central pixel and checking whether a run of n consecutive pixels (with $n = 9$ most commonly used) which are brighter or darker than the central pixel exists. Furthermore, the ordering of tests is learned from training data to reject non-corner pixels as quickly as possible. Although this approach only offers invariance to rotation and translation, the fact that it is extremely fast, even on low-powered devices, means it is frequently used. Leutenegger *et al.* [72] propose an extension to FAST which also provides invariance to image scaling.

2.2.2 Keypoint matching

Once keypoints have been identified, the next stage in the object detection pipeline is to use local appearance information in order to match keypoints in an image against keypoints on the target object. Methods for achieving this can be divided into two categories: those based on descriptors and those based on classification.

2.2.2.1 Matching with descriptors

The traditional approach to matching has been to produce a *descriptor* for each keypoint, a signature based on local image information. Ideally these descriptors should be invariant to the same class of transformations as the detector, and the usual approach is to use statistics based on the image data around a keypoint to define a local coordinate frame in which to compute the descriptor. Given a descriptor type, matching then becomes a nearest-neighbour problem. For each keypoint in an image, the nearest-neighbour object keypoint is found using some distance metric (usually Euclidean). If this distance is sufficiently small, it is considered a candidate match. Additional heuristics are also employed in practice, such as rejecting matches which are not sufficiently unique, as defined by the ratio of distances between the nearest and second-nearest match [75].

Schmid and Mohr [103] introduced the concept of image descriptors by building Local Jets, vectors of local image derivative information, around image keypoints. Since then, many other descriptors have been proposed, by far the most well-known of which is the SIFT descriptor [75]. This descriptor is constructed from histograms of oriented gradient information collected from a 4x4 grid around each keypoint, along with normalisation to increase robustness to illumination changes (Figure 2.9). This carefully-designed descriptor and its associated DoG detector have become the gold standard in terms of performance for keypoint matching against which all other approaches are compared.

An issue with the SIFT descriptor is its computational cost. Construction of the descriptor involves relatively expensive image operations such as convolutions with Gaussian kernels and the resulting descriptor is a 128-dimensional real vec-

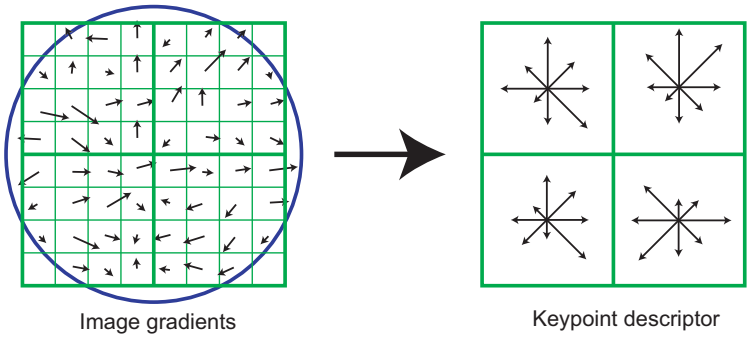


Figure 2.9: The SIFT descriptor. Local image gradients weighted by a Gaussian kernel around DoG keypoints are collected into a spatial histogram to produce the descriptor. This example shows 2x2 histograms, while the actual descriptor uses 4x4 histograms. Image courtesy of D. Lowe [75]

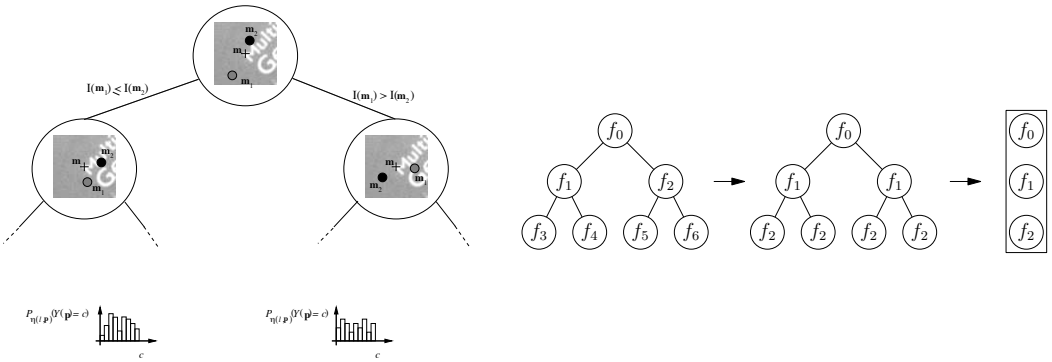
tor, making nearest-neighbour search expensive. Various approaches have been proposed for addressing these issues. The SURF descriptor [10] was designed to match the performance of SIFT, whilst replacing various stages in the pipeline with more efficient alternatives making use of integral images [123] and the resulting descriptor is only 64-dimensional. This approach has been shown to be suitable for real-time application on desktop computers, although it is still too expensive for low-powered devices. Approaches for accelerating the nearest-neighbour search include reducing the dimensionality of the descriptor through the use of principal component analysis (PCA) [61] or vector quantisation [112,120] and approximate search methods based on efficient tree structures [81,84,91]. However, these approaches all require the original descriptor to be computed first, which may itself be prohibitively expensive, particularly on low-powered devices.

2.2.2.2 Matching as classification

An alternative view of keypoint matching is to treat it as a classification problem. In this setting, each object keypoint defines a class, and a classifier is trained to identify which (if any) of these classes a given image keypoint corresponds to. This approach was first introduced by Lepetit and Fua [71], who trained a random forest [22] classifier based on simple tests between pairs of pixels around a keypoint. Subsequently, a related approach by Özuyal *et al.* [87] replaced the random forest classifier with a more discriminative and memory efficient random fern classifier (Figure 2.10). The key factor in both of these approaches is the

training stage of the classifier. This proceeds by generating a large number of synthetic views of the target object and then relying on the learning algorithm to choose tests which together discriminate between the object keypoints. There are two main strengths to such an approach. The first is that the classifier is tuned specifically for the object of interest and can focus its tests appropriately. This is in contrast to descriptor-based approaches, which require a universal descriptor which is suitable for all objects. The second strength is that because the tests are chosen to be very simple, the resulting classifier is efficient to evaluate at run-time, allowing for real-time object detection.

The weakness of classification-based approaches is that the training stage is typically time-consuming and computationally expensive, as a large number of examples of the object keypoints under various transformations must be generated to produce an accurate classifier. This is acceptable for certain applications, such as detecting a fixed image for an AR application, as the classifier can be fully trained offline and then will only ever be evaluated at runtime. There are other situations, however, where the classifier needs to be updated at runtime to include new keypoints. One such example is SLAM, where keypoint-based object detection can be used to perform relocalisation when tracking fails. Williams *et al.* [127] propose a modification of the random forest approach to allow learning of new keypoints for SLAM, but even with simplification of the classifier and



(a) Random forests **(b)** Random ferns

Figure 2.10: Random forests and ferns. Decision trees are constructed consisting of pairwise pixel tests around a keypoint. While random forests select different tests at each node, random ferns restrict all nodes at a given level to use the same test, resulting in a simpler linear structure and lower memory requirements. Images courtesy of V. Lepetit [71] and M. Özuysal [87]

the use of the GPU, training is still computationally expensive and can only handle a relatively small number of keypoints. Özuysal *et al.* [88] propose an alternative approach which uses an online version of random forest training, allowing the classifier to be updated incrementally as new training data arrives. Their approach is therefore suitable for situations in which keypoints are added or removed at runtime, but still requires examples of the keypoints under many different transformations, which must somehow be supplied.

2.2.2.3 Methods for low-powered devices

There has recently been significant interest in developing approaches for keypoint matching which are suitable for portable devices such as smartphones and tablets. These devices provide an excellent platform for AR and SLAM applications, but they also have far less computational power than a typical desktop computer.

As has already been discussed, descriptor-based approaches typically involve expensive image operations, followed by high-dimensional nearest-neighbour search. While classification-based approaches were designed to provide more efficient matching, the classifiers involved often have high memory usage, which also makes them unsuitable for low-powered devices. Wagner *et al.* [125] present a number of carefully-engineered modifications to both of these categories of approaches which allow them to run on low-powered devices. They propose an approximation of the SIFT descriptor and matching procedure which results in significantly lower computational cost compared with the original approach. They also present an approximation of the ferns approach which results in much lower memory usage compared with the original.

Rather than improving the efficiency of existing approaches, there have also been a number of recent methods proposed which are designed from the ground up to be suitable for low-powered devices.

Taylor *et al.* [116] propose Histogrammed Intensity Patches (HIPs), a classification based approach which builds independent histograms of pixel intensities for 64 sample locations around a model keypoint from training data. These histograms are each approximated very coarsely in binary form using 5 bits, resulting in a 320-bit representation of a model keypoint. Using a similar representation for

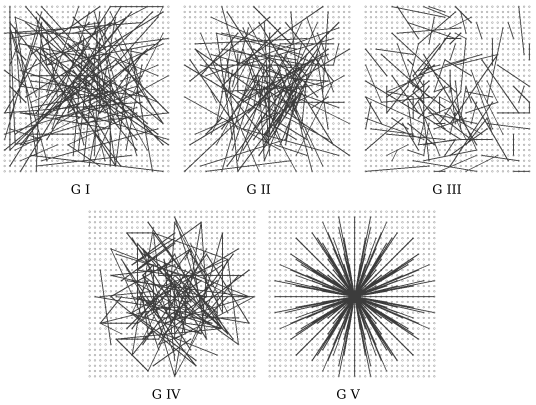


Figure 2.11: The BRIEF descriptor. Randomly-generated pairwise pixel tests are concatenated to give a binary descriptor. This figure shows different sampling strategies for selecting the tests. Image courtesy of M. Calonder [25]

an image keypoint, a dissimilarity measure between model and image keypoints can be computed using a bitwise XOR followed by a bit-count, both of which can be achieved very efficiently using bitwise operations on a CPU.

Calonder *et al.* [25] propose a descriptor-based approach called BRIEF, which uses simple binary tests on randomly-generated pairs of pixels around a keypoint, inspired by the random forest and fern classification-based approaches. The results of a number of independent tests are concatenated together to produce a binary descriptor (Figure 2.11). The distance between two such descriptors can then be computed using the Hamming distance, which can be computed very efficiently on a CPU using bitwise operations. Although this approach is extremely simple, it has experimentally been shown to produce results comparable to SIFT and SURF matching, whilst being around two orders of magnitude faster [24]. A number of variations on this approach have subsequently been proposed, which retain the core idea, but improve matching further by tuning the binary tests which are chosen for the descriptor [5, 72, 98].

2.2.3 Geometric verification

The final stage of the detection pipeline is geometric verification, which uses the set of independently-found keypoint matches to infer the overall presence and transformation of the target object. If the set of matches was known to be largely correct then this would be a simple task, and we could use *e.g.* least-squares

estimation to find the best-fitting transformation between model and image given the set of matches. However, because matches are generated independently by considering only local image information, the expectation is that there will be a significant number of outlier (incorrect) matches. For this reason, a *robust* estimation procedure must be used which is able to tolerate these outliers.

The majority of approaches for geometric verification are based on RANSAC [41]. Such approaches proceed by randomly sampling minimal subsets from the full set of matches to generate transformation hypotheses and then use the remaining matches to test these hypotheses. The number of matches which define a minimal subset depends on the class of transformation which is being considered [53]. To estimate a homography, for example, 4 matches are required, while to estimate 3D rotation and translation given known intrinsic camera parameters (the P3P problem), 3 matches are required. Depending on the ratio of inlier to outlier matches, with a sufficiently large number of random samples the probability of selecting a minimal set which is free from outliers is very high, allowing the procedure to robustly estimate an overall object transformation. Alternatively, if no transformation can be found with sufficient support from the set of matches, no detection is reported.

Subsequent research has further extended the underlying RANSAC approach to use a more principled maximum-likelihood estimation procedure [118], which is now more commonly used in practice. Another important improvement is the PROSAC algorithm [29], which does not sample matches uniformly at random, but rather assumes the matches can be ranked according to their quality (*e.g.* using their matching score) and biases the sampling to focus initially on the best matches. In practice this approach is able to estimate transformations with substantially fewer iterations than RANSAC, which brings great benefit for real-time applications.

2.3 Scene reconstruction

The task of reconstructing the underlying 3D scene which has been observed by a camera is a fundamental problem in computer vision, as it essentially aims

to invert the imaging process performed by a camera. Obtaining a 3D reconstruction of a scene is particularly useful for AR applications, as it allows virtual content to be introduced which interacts with its environment in a realistic way, resulting in a more compelling experience for the user. Inverting a 2D image is of course not possible in general, although approaches have been proposed which attempt to achieve this by incorporating additional prior assumptions about the scene [54, 132]. Given multiple views of the scene, however, the problem of scene reconstruction becomes well-posed, and there now exist a variety of sophisticated approaches able to produce high-quality dense 3D scene reconstructions.

Such multi-view reconstruction approaches take as their input multiple calibrated images of a scene, meaning the intrinsic camera parameters are known (focal length, principal point, distortion coefficients, etc), as well as the extrinsic camera parameters for each view (the 3D camera pose). This calibration information may either come from a carefully controlled capture environment in which the 3D pose of the camera is known in advance, or by using structure-from-motion techniques [53] to estimate the calibration information from the images themselves.

In order to estimate 3D information from multiple views, approaches typically make use of *photo-consistency* to establish dense correspondences between pixels in each view, which subsequently allows a 3D position for each pixel to be estimated by triangulation. This technique is referred to as multi-view stereo, since it generalises the principles used by stereo algorithms to more than two images. While the core principle for these methods remains the same, there are still a great variety of multi-view stero approaches [105] which differ in various factors such as how they measure photo-consistency, how they represent the scene, how they handle occlusion between views, and the optimisation strategy they use.

While photo-consistency provides a strong cue for multi-view reconstruction, there are situations in which it may not be able to provide useful information. Problems can occur with textureless regions, for example, where it becomes impossible to reliably establish correspondences between multiple views. Similar problems can also occur at occlusion boundaries. To tackle these problems, Campbell *et al.* [26] propose an approach to incorporate higher-level structural

constraints based around local continuity to help resolve such ambiguities.

2.3.1 Real-time approaches

Traditional multi-view reconstruction methods are designed to operate offline and are often computationally very expensive, taking minutes or hours on powerful desktop computers to produce reconstructions. Recently, however, there has been increased interest in developing approaches which are able to produce real-time reconstructions, and it is these approaches which have the most potential for AR applications.

One class of real-time method are those based around space-carving, which perform volumetric reconstruction using reasoning based on the visibility of features in the scene observed from multiple views. These approaches lend themselves well to the coarse reconstruction of individual closed objects, for example allowing a user to 'scan' an object to produce a 3D model. The ProFORMA method [89] achieves this by tracking points on the surface of an object as it is moved in front of a fixed camera and subsequently uses tetrahedral space-carving to produce a textured object model (Figure 2.12). A related approach proposed by Basitan *et al.* [9] tracks the silhouette of an object by performing colour-based segmentation in multiple viewpoints and uses space-carving on a voxel grid to generate a 3D reconstruction of the object.

Space-carving methods are typically not able to produce very accurate reconstructions and suffer from problems based on the topology of the scene. For

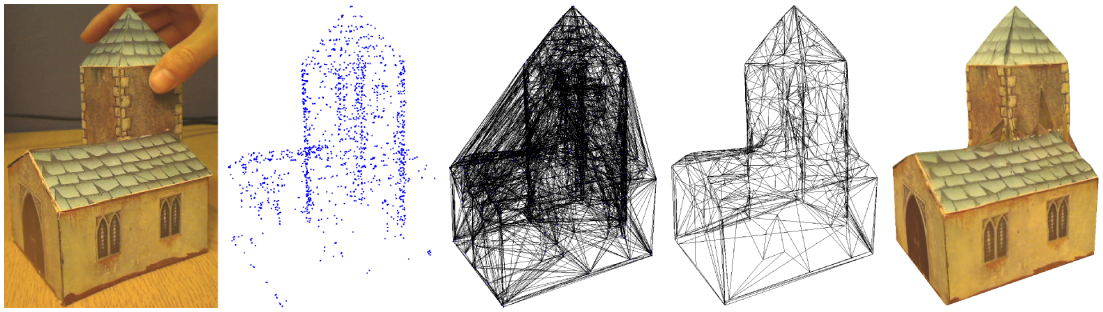


Figure 2.12: The ProFORMA reconstruction method. Points are tracked on the surface of an object to produce a 3D point cloud. Delaunay tetrahedralisation is applied and space-carving is used to remove empty tetrahedra. Image courtesy of Q. Pan [89]

example, space-carving based on silhouettes is not able to reconstruct concavities in an object, as these do not affect the silhouette. Another class of real-time method are those which use multi-view stereo, but have been engineered to be highly efficient so that they are capable of real-time reconstruction. One such example is the method of Vogiatzis and Hernández [124], which is able to estimate the 3D position of a large number of points in real-time to produce a dense point cloud for a scene. Methods proposed by Newcombe and Davison [82] and Stuehmer *et al.* [113] both use sophisticated optimisation algorithms which can be implemented on the GPU in order to produce real-time depth-maps for multiple views, which are then fused to give 3D scene reconstructions (Figure 2.13).

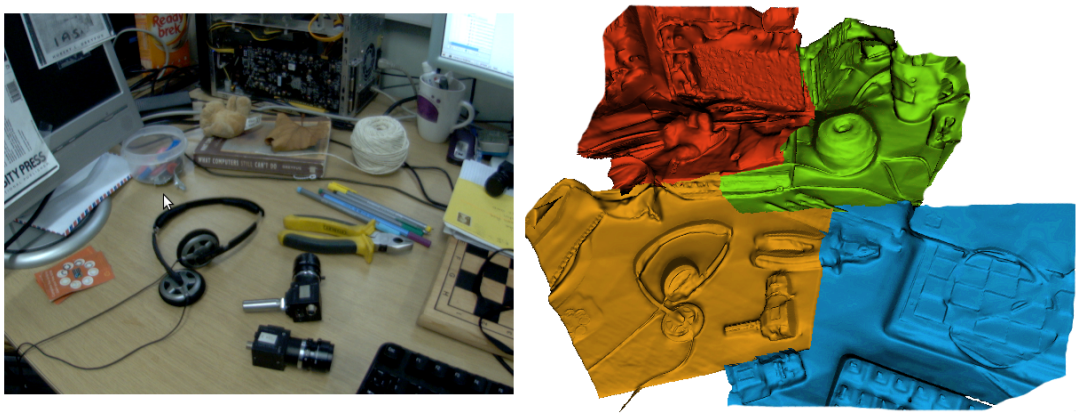


Figure 2.13: Real-time multi-view stereo using the method of Newcombe and Davison [82]. Here depth-maps have been computed and fused from 4 reference views to produce a 3D reconstruction. Image courtesy of R. Newcombe [82]

2.4 Structured learning

Computer vision as a field has benefited greatly from progress in machine learning, and powerful statistical models which can be learned efficiently from large quantities of data now form the core of most modern vision techniques.

The types of problems dealt with in computer vision often involve rich models with a large amount of *structure*. Such structure exists at various levels in the vision pipeline. At the low level, there is structure in terms of the local spatial relationships between pixels in an image. For higher-level scene understanding tasks, models are introduced which are structured, such as pictorial structures,

hidden markov models, etc.

Recent developments in machine learning have provided tools for learning with such structured models, and this section provides a summary. For further reading, the interested reader is directed to the survey by Nowozin and Lampert [85].

2.4.1 Structured prediction

Structured prediction provides a general framework for the task of finding solutions to structured problems. In this setting we have a prediction function $f : \mathcal{X} \to \mathcal{Y}$ from an input domain $\mathcal{X}$ to a structured output domain $\mathcal{Y}$. This prediction function is defined such that it makes use of an auxiliary function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, which can be seen as measuring the *compatibility* of an input-output pair. Predictions are then made according to

$$\hat{\mathbf{y}} = f(\mathbf{x}) \coloneqq \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} g(\mathbf{x}, \mathbf{y}), \tag{2.1}$$

meaning that $\hat{\mathbf{y}}$ is the output which has the highest compatibility with the input $\mathbf{x}$.

Such a framework encompasses many approaches commonly used in computer vision, and performing prediction amounts to solving an optimisation problem, with an objective function determined by $g(\mathbf{x}, \mathbf{y})$. Defining this objective function and finding efficient ways of solving it thus form the core of structured prediction problems.

2.4.1.1 Sliding-window object localisation

One example of a structured prediction approach used in computer vision is sliding-window object localisation, which is the most common method used for performing category-level object localisation. Here the task is to localise instances of a given category (*e.g.* face, person, car, etc.) in an image, typically by drawing a bounding box around them (Figure 2.14). Sliding-window approaches [33,40,122, 123] achieve this by training a classifier to predict whether a given bounding box in an image contains the category of interest or not. Localisation then proceeds by

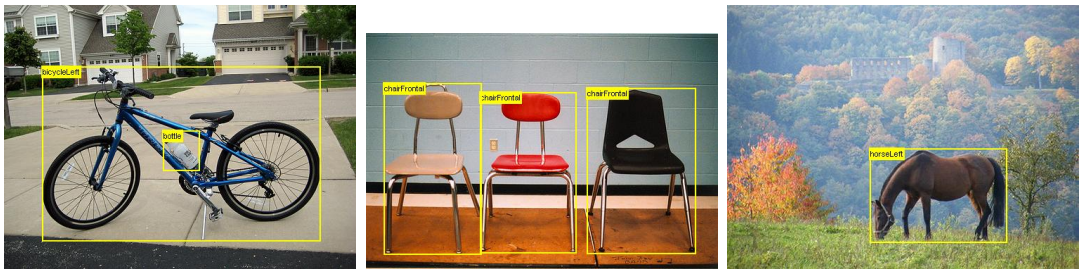


Figure 2.14: Sliding-window object localisation. The goal is to draw bounding boxes around instances of known object classes in an image.

searching over all possible bounding boxes in an image, with detections reported at local maxima of the classification confidence function. As such, this process can be viewed as an instance of structured prediction, in which the input is an image $\mathbf{x}$ and the output is a bounding box $\mathbf{y}$. The sliding-window search procedure is performing the maximisation (2.1), with $g(\mathbf{x}, \mathbf{y})$ being the classification confidence function for a given bounding box in the image.

Sliding-window methods must search and test a very large number of bounding boxes in an image, which is potentially too expensive for practical purposes. Consequently, researches have developed efficient ways of performing this prediction. One approach is to introduce a cascaded classifier [122, 123], which uses a simple and fast classifier in its early stages to reject windows which obviously do not contain the object, saving the full classifier for a smaller number of more promising windows. For certain types of classifier another method is to use branch-and-bound optimisation [67], which can make use of an upper bound on the classification score of a collection of windows in order to reject portions of an image which cannot possibly result in a detection.

2.4.1.2 Conditional random fields

Discrete labelling problems occur frequently in computer vision and are often modelled as conditional random fields (CRFs). A CRF consists of a set of random variables $\mathbf{Y} = \{Y_1, \ldots, Y_N\}$, each of which can be assigned a label from a set $\mathcal{L} = \{l_1, \ldots, l_K\}$. Often, the task is to label each pixel in an image, meaning there will be one random variable per image pixel. Examples of the types of labels include categories (*e.g.* road, building, tree, sky, etc.) in the case of semantic segmentation (Figure 2.15), greyscale intensity values in the case of image

denoising, or disparity values in the case of stereo matching.

Figure 2.15: Semantic segmentation. The goal is to label each pixel in an image with one of a number of known categories. Image courtesy of J. Shotton [110]

The random variables $\mathbf{Y}$ are not independent, but rather affect one another based on a neighbourhood $\mathcal{N}$, which most commonly consists of pairwise connections between random variables. Given some data $\mathbf{x}$ (*e.g.* the image data for semantic segmentation or denoising, or a pair of images for stereo matching), the posterior probability distribution of a particular labelling $\mathbf{y}$ of a pairwise CRF is defined by a *Gibbs* distribution [51]:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=1}^{N} \exp(-\psi_i(y_i)) \prod_{(i,j)\in\mathcal{N}} \exp(-\psi_{ij}(y_i, y_j)) \tag{2.2}$$

where Z is a constant normalisation factor, and the terms $\psi_i(y_i)$ and $\psi_{ij}(y_i, y_j)$ are referred to as unary and pairwise potentials, respectively. The way these potentials are defined is problem-specific. In the case of semantic segmentation, for example, the unary potential is generally determined by some type of classifier giving a per-pixel confidence of category membership, while the pairwise potential encourages neighbouring pixels with similar appearance to be assigned to the same category. Finding the maximum a-posteriori labelling of a CRF given some data $\mathbf{x}$ is thus an instance of structured prediction, since we wish to find:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}\in\mathcal{L}^N}{\operatorname{argmax}}\, P(\mathbf{y}|\mathbf{x}). \tag{2.3}$$

Figure 1: **Example results of our new simultaneous object class recognition and segmentation algorithm.** Up to 21 object classes (color-coded in the key) are recognized, and the corresponding object instances segmented in the images. For clarity, textual labels have been superimposed on the resulting segmentations. Note, for instance, how the airplane has been correctly recognized and separated from the building, the sky, and the grass lawn. In these experiments only one *single* learned multi-class model has been used to segment all the test images. Further results from this system are given in Figure 18.

mination, and to be robust to occlusion. Our focus is not only the accuracy of segmentation and recognition, but also the efficiency of the algorithm, which becomes particularly important when dealing with large image collections or video sequences.

At a local level, the *appearance* of an image patch i leads to ambiguities in its class label. For example, a window could be part of a car, a building or an airplane. To overcome these ambiguities, it is necessary to incorporate longer range information such as the spatial *layout* of an object and also *contextual* information from the surrounding image. To achieve this, we construct a discriminative model for labeling images which exploits all three types of information: texture, appearance, layout and context. Our technique can model very long-range contextual relationships extending over half the size of the image.

Additionally, our technique overcomes several problems typically associated with object recognition techniques that rely on sparse features (such as [33, 36]). These problems are mainly related to textureless or very highly textured image regions. Figure 2 shows some examples of images with which those techniques would very likely struggle. In contrast, our technique based on dense features is capable of coping with both textured and untextured objects, and with multiple objects which inter- or self-occlude, while retaining high efficiency.

The main contributions in this paper are threefold. The most significant is a novel type of feature, which we call the texture-layout filter. These features record patterns of textons, and exploit the textural context. Our second contribution is a new discriminative model that combines texture-layout filters with lower-level image features, in order to pro-

The distribution (2.2) is log-linear, which means performing this maximisation is equivalent to minimising the Gibbs energy, defined as:

$$E(\mathbf{y}) = \sum_{i=1}^{N} \psi_i(y_i) + \sum_{(i,j)\in\mathcal{N}} \psi_{ij}(y_i, y_j) \tag{2.4}$$

Minimising this energy is NP-hard in general, but in certain cases it can be performed exactly and in polynomial time. In the case of tree-structured CRFs, belief propagation [90] can be used. In the case of *submodular* energy functions [64], the energy minimisation is equivalent to a graph cut problem, for which several efficient algorithms exist [20, 64].

In other cases, minimising (2.4) exactly is not feasible, but there are still efficient approximate approaches. In particular, for the case of non-submodular multi-label problems the α-expansion and $\alpha\beta$-swap move-making algorithms [21] are widely used.

2.4.2 Learning the prediction function

The prediction function (2.1) may be designed by hand to capture the properties of the problem of interest, but in many cases it is desirable to *learn* the function based on training data. While structured prediction is often used in computer vision, typically the way learning has been introduced does not take into account the structure of the problem. For example, most approaches for sliding-window object detection (as discussed in Section 2.4.1.1) involve training a binary classifier from a training set of positive and negative examples. Therefore the learning is for making this binary decision. However, in practice this classifier will be used inside a sliding-window framework to perform structured prediction, which is not taken into account at all by the learning. Blaschko and Lampert [14] tackled this problem in their influential work and showed how this pipeline can be better embedded in a learning framework using a recently-proposed extension of the support vector machine (SVM) [31] to structured output spaces [119]. We now provide an overview of the classification SVM, and show how the principles behind it can be extended to structured learning problems.

2.4.2.1 Classification SVM

The classification SVM [31] is one of the most widely-used tools in machine learning and computer vision. The task is to take a set of training examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $y_i \in \{-1, 1\}$, and learn a classification function $h : \mathcal{X} \to \mathbb{R}$ which can be used to make predictions according to

$$\hat{y} := \text{sign}(h(\mathbf{x})). \tag{2.5}$$

The SVM defines $h(\mathbf{x})$ as a linear function of the input

$$h(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b, \tag{2.6}$$

where b is a constant bias, and $\mathbf{w}$ represents a hyperplane defining the *decision boundary* $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$. Assuming the training data are separable, that is, it is possible to find a decision boundary which correctly classifies all the positive and negative training examples, the SVM finds $\mathbf{w}$ which results in the largest possible separation between the positive and negative examples. To achieve this, two additional hyperplanes $\langle \mathbf{w}, \mathbf{x} \rangle + b = 1$ and $\langle \mathbf{w}, \mathbf{x} \rangle + b = -1$ are taken on either side of the decision boundary such that no training examples lie in the region in between. The region between these two hyperplanes is referred to as the *margin*, and the SVM finds the decision boundary which maximises the size of this margin for the training data (Figure 2.16). The size of the margin can be shown geometrically to be inversely proportional to $\|\mathbf{w}\|$, meaning the maximum-

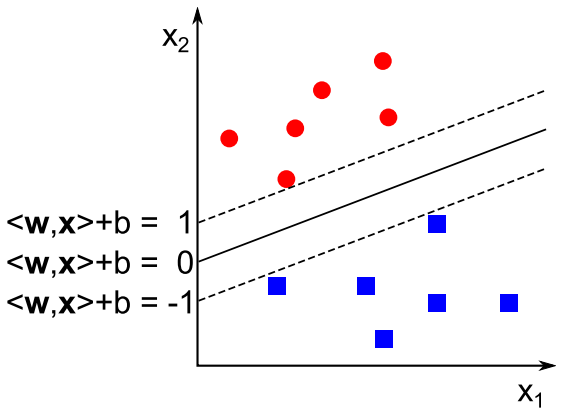


Figure 2.16: The classification SVM. Given linearly separable training data, the SVM finds the decision boundary with the largest margin between the two classes.

margin decision boundary can be found by solving the following convex quadratic optimisation problem [31]:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & \forall i : y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1. \end{aligned} \tag{2.7}$$

To handle the situation where the training examples are not linearly separable, it is possible to introduce *slack variables* which allow some of the training examples to violate the constraint that they must lie outside of the margin. The optimisation problem then becomes

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & \forall i : \xi_i \geq 0 \\ & \forall i : y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \end{aligned} \tag{2.8}$$

where C is a parameter which controls how strongly margin violations are penalised. When $C = \infty$, this optimisation problem is equivalent to (2.7), since it forces all $\xi_i = 0$.

The typical way in which (2.8) is solved is first by introducing Lagrange multipliers [31]:

$$\min_{\mathbf{w},\boldsymbol{\xi}} \max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 + \xi_i) - \sum_{i=1}^{N}\beta_i\xi_i \right\}, \tag{2.9}$$

with $\alpha_i, \beta_i \geq 0$. Applying the stationary Karush-Kuhn-Tucker (KKT) [65] condition and making the relevant substitutions into (2.9) results in the Lagrangian dual form

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ \text{s.t.} \quad & \forall i : 0 \leq \alpha_i \leq C. \\ & \sum_{i=1}^{N}\alpha_i y_i = 0 \end{aligned} \tag{2.10}$$

Another implication of the stationary KKT condition is an instance of the representer theorem [104], which states that the solution to this optimisation can

always be expressed as a linear combination of the training examples:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i. \tag{2.11}$$

Those training examples for which $\alpha_i > 0$ are referred to as *support vectors*, and in general this solution will be sparse, meaning only a small proportion of the training examples will have $\alpha_i > 0$.

It is also possible to extend the SVM to support non-linear classification. Since all input vectors $\mathbf{x}_i$ only ever appear inside scalar products (both during training and classification), it is possible to use the *kernel trick* and first apply a non-linear feature mapping $\boldsymbol{\phi}(\mathbf{x})$ to an input vector $\mathbf{x}$, before taking scalar products in this new feature space. A linear classifier learned in this mapped feature space will then correspond to a non-linear classifier in the original input space. This approach can be taken further by replacing scalar products with a *kernel function* $k(\mathbf{x}, \mathbf{x}')$. In this case, it is not necessary to perform an explicit feature-mapping of the input vectors. Provided that the kernel function satisfies certain properties [3], it can be shown that its evaluation is equivalent to a scalar product in a corresponding Hilbert space, which may even have infinite dimensionality. Since this mapping is never explicitly computed, evaluation of the kernel can remain efficient.

There are a number of mature, publicly-available SVM solvers which have been designed to efficiently solve the SVM optimisation problem (2.8) [28, 58]. Most of these in practice solve the dual problem (2.10), using the efficient sequential minimal optimisation (SMO) procedure proposed by Platt [92].

2.4.2.2 Structured SVM

Recently, the SVM has been extended beyond classification so that it can also be used for structured prediction problems [115, 119]. The task in this setting is to learn the prediction function (2.1) given a set of training examples $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, where now $\mathbf{y}_i \in \mathcal{Y}$ is a structured label. The way this problem is approached with the structured SVM is to define the auxiliary function $g(\mathbf{x}, \mathbf{y})$ as a linear

function

$$g(\mathbf{x}, \mathbf{y}) = \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) \rangle, \tag{2.12}$$

where $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ is a *joint feature mapping* of the input-output pair. Learning $g(\mathbf{x}, \mathbf{y})$ can thus be achieved by learning $\mathbf{w}$. Given the prediction function (2.1), the goal of learning is to satisfy the constraints:

$$\forall i : \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i) \rangle \geq \max_{\mathbf{y} \in \mathcal{Y} \setminus y_i} \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}) \rangle. \tag{2.13}$$

These constraints are non-linear, however they can equivalently be replaced by a larger set of linear constraints:

$$\forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus y_i : \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i) \rangle \geq \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}) \rangle. \tag{2.14}$$

As in the case of the classification SVM, there may be many $\mathbf{w}$ which satisfy all of these constraints, so it is necessary to define additional criteria for selecting the 'best' $\mathbf{w}$. This is achieved by generalising the concept of the margin, such that it now refers to the minimal difference between the score of a correct label and the closest runner-up over the entire training set [119]:

$$\gamma = \min_i \max_{\mathbf{y} \in \mathcal{Y} \setminus y_i} \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i) \rangle - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}) \rangle. \tag{2.15}$$

Analogously to the classification SVM, the structured SVM finds the $\mathbf{w}$ which maximises γ for the training data, which can be shown to be achieved with the following convex quadratic optimisation problem [119]:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & \forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus y_i : \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i) \rangle - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}) \rangle \geq 1. \end{aligned} \tag{2.16}$$

To handle the situation where it is not possible to satisfy the constraints (2.13), slack variables are introduced which allow some of the training examples to violate

them. The optimisation problem then becomes:

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & \forall i : \xi_i \geq 0 \\ & \forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus y_i : \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i)\rangle - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y})\rangle \geq 1 - \xi_i, \end{aligned} \tag{2.17}$$

where C is a parameter controlling how strongly margin violations are penalised. An issue with this formulation is that all margin violations are treated equally. In the case of the classification SVM this is appropriate, since the problem is binary. For structured prediction, however, it is desirable for prediction errors which are close to the correct label to be penalised less than those which are significantly different. This can be achieved by defining a problem-specific *loss function* $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$. This loss function should satisfy $\Delta(\mathbf{y}, \hat{\mathbf{y}}) = 0$ iff $\hat{\mathbf{y}} = \mathbf{y}$ and increase as $\hat{\mathbf{y}}$ and $\mathbf{y}$ become more dissimilar. This loss function can be incorporated into (2.17) by *margin rescaling*, which defines the size of the required margin between outputs according to the loss function[1] [115, 119]:

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & \forall i : \xi_i \geq 0 \\ & \forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus y_i : \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i)\rangle - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y})\rangle \geq \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i, \end{aligned} \tag{2.18}$$

As can be seen, the structured SVM optimisation problem is very similar in form to that of the classification SVM (2.7). The major difference is that now instead of N constraints, there are $N(|\mathcal{Y}| - 1)$. Depending on the size of the output space, this is potentially a very large or even infinite (if the output space is continuous) number. Nevertheless, practical approaches exist for performing this optimisation. Tsochantaridis *et al.* [119], who first introduced the structured SVM as presented here, also proposed a *cutting plane* [62] scheme for solving (2.18). The key observation is that although there are a very large number of constraints, only a small fraction of them will ever be active, with the remaining

[1] An alternative approach is *slack rescaling* [119], in which the loss function is incorporated by replacing the slack variables in (2.17) with $\xi_i \leftarrow \xi_i / \Delta(\mathbf{y}_i, \mathbf{y})$; however, this has seen less use in the computer vision literature.

ones satisfied automatically. The optimisation procedure maintains an active set of constraints, which define a reduced optimisation problem which can be solved to find $\mathbf{w}$. Given this solution, any constraints which are violated from the full set are identified and added into the active set, and the procedure is iterated until no further violated constraints exist. This method provably converges to the solution of (2.18) and has the additional benefit that the core optimisation procedure is able to use the same efficient methods [92] as standard classification SVM solvers. Subsequent improvements have also been made to this approach which result in faster convergence guarantees [59].

As in the case of classification SVMs, it is also possible to extend structured SVMs to non-linear prediction functions by kernelisation. In the case of classification SVMs, such kernels $k(\mathbf{x}, \mathbf{x}')$ operate on two elements from the input domain only. Structured SVMs extend this concept and make use of *joint kernels*, which operate on two input-output pairs:

$$k(\mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}') = \langle \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}), \boldsymbol{\phi}(\mathbf{x}', \mathbf{y}') \rangle. \tag{2.19}$$

An example of such a joint kernel is the *restriction kernel* [14], used for object localisation. Here the inputs $\mathcal{X}$ are images, and the outputs $\mathcal{Y}$ are bounding boxes. The restriction kernel $k_r(\mathbf{x}|_{\mathbf{y}}, \mathbf{x}'|_{\mathbf{y}'})$ applies any standard image-based kernel to the regions in $\mathbf{x}$ and $\mathbf{x}'$ defined by the bounding boxes $\mathbf{y}$ and $\mathbf{y}'$.

2.4.2.3 Online learning

Both the classification and structured SVM as presented so far assume that all the training data are available at the time of learning. This scenario is referred to as *batch* learning. A different scenario is *online* learning, in which the training data arrive sequentially. In this setting, the learner must be incrementally updated each time a new training example arrives. The current state of the learner is used in order to predict the label for this new example, which is then compared to the true label, and adjustments are made to the learner as appropriate. Besides handling the situation where training data truly does arrive sequentially, online learning is a useful tool when there is a great deal of training data which cannot

practically be processed by a batch learning algorithm.

Recent research has resulted in a variety of methods for training SVMs in an online fashion. These methods can be separated into two classes: those which operate in the primal, and those which operate in the dual.

Primal approaches. Methods for training SVMs online in the primal are generally based on *stochastic sub-gradient descent*. As an illustrative example, we present an overview of the Pegasos [106] algorithm for online training of a classification SVM. This approach maintains a hyperplane $\mathbf{w}_t$ which summarises the result of learning from all examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{t-1}$. The objective function of the primal SVM optimisation problem (2.8) can be rewritten in unconstrained form (with a constant scaling that does not affect the solution) by eliminating the slack variables $\boldsymbol{\xi}$:

$$f(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N}(1 - y_i\langle \mathbf{w}, \mathbf{x}_i\rangle)_+, \tag{2.20}$$

where $\lambda = \frac{1}{NC}$, and $(z)_+ = \max\{0, z\}$ is the *hinge* function. In order to optimise this given a single training example $(\mathbf{x}_t, y_t)$, Pegasos considers an approximate objective function based on just this example:

$$f(\mathbf{w}; t) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + (1 - y_t\langle \mathbf{w}, \mathbf{x}_t\rangle)_+. \tag{2.21}$$

This approximation is justified probabilistically because, considering the training examples as random variables, the expectation of its gradient is equivalent to the actual gradient of (2.20). The function (2.21) is convex in $\mathbf{w}$, but non-differentiable due to the discontinuity in the gradient of the hinge function $(z)_+$ at $z = 0$. Nevertheless, a *sub-gradient* [108] is given by

$$\boldsymbol{\nabla}_t = \lambda\mathbf{w}_t - \mathbb{I}(y_t\langle \mathbf{w}_t, \mathbf{x}_t\rangle < 1)y_t\mathbf{x}_t, \tag{2.22}$$

where $\mathbb{I}(\cdot)$ is an indicator function which takes a value of 1 if its argument is true and 0 otherwise. This sub-gradient is then used to perform a single gradient descent step according to

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t\boldsymbol{\nabla}_t, \tag{2.23}$$

where $\eta_t = \frac{1}{\lambda t}$ is the step size at time t. This approach provably converges to the batch SVM solution [106], whilst being extremely simple to implement. Further improvements have also been proposed which can accelerate convergence, such as performing updates to $\mathbf{w}$ which are averaged over time [93, 129].

A very similar approach for online learning can also be taken for the case of structured SVMs, where now the approximate objective function based on the structured training example $(\mathbf{x}_t, \mathbf{y}_t)$ is derived from (2.18) and given by:

$$f(\mathbf{w}; t) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + (\max_{\mathbf{y} \in \mathcal{Y} \setminus y_t} \{\Delta(\mathbf{y}_t, \mathbf{y}) + \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y})\rangle - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t)\rangle\})_+. \quad (2.24)$$

Let

$$\hat{\mathbf{y}}_t = \arg\max_{\mathbf{y} \in \mathcal{Y} \setminus y_t} \{\Delta(\mathbf{y}_t, \mathbf{y}) + \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y})\rangle\}, \quad (2.25)$$

then a sub-gradient is given by:

$$\boldsymbol{\nabla}_t = \lambda \mathbf{w}_t - \mathbb{I}(\Delta(\mathbf{y}_t, \hat{\mathbf{y}}_t) + \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_t, \hat{\mathbf{y}}_t)\rangle - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t)\rangle > 0)(\boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \hat{\mathbf{y}}_t)), \quad (2.26)$$

and $\mathbf{w}_t$ is updated in the same way as before. Notice that finding $\hat{\mathbf{y}}_t$ in(2.25) is closely related to the prediction function (2.1), except it now also includes the loss function Δ. This step is referred to as *loss-augmented prediction* and is a core consideration when learning with the structured SVM. Ideally, the loss function should be chosen in such a way that it decomposes over the output space, meaning the efficient prediction algorithms discussed in Section 2.4.1 can still be applied [14, 114].

Dual approaches. While primal approaches for online learning are efficient and simple to implement, they also rely on an explicit representation of the SVM weight vector $\mathbf{w}$. As has been discussed, one of the key strengths of SVMs is that non-linearity can be introduced through the use of kernels. However, once kernels are employed the weight vector is only represented implicitly based on the set of support vectors. In order to make use of kernels in an online setting, alternative algorithms have been proposed which perform online optimisation of the dual SVM optimisation problems. In the case of the classification SVM, the LASVM algorithm [16] performs online optimisation of (2.10), and the approach

has subsequently been extended to the structured SVM to perform online optimisation of the dual form of (2.18) with the LaRank algorithm [15, 17]. All of these methods are based on the fact that the standard approach for optimising the dual form of SVMs is to use sequential minimal optimisation (SMO) [92]. SMO involves repeatedly solving minimal sub-problems of the dual optimisation involving only pairs of Lagrange multipliers α_i and α_j, along with a strategy for choosing these pairs to encourage fast convergence. LASVM and LaRank both adapt SMO to an online setting, by alternating between optimising the Lagrange multipliers associated with new training examples as well as of existing support vectors.

When optimising in the dual, the solution is entirely defined by the set of support vectors. It is known that in general the number of support vectors increases with the size of the training set, meaning that in an online setting the number of support vectors grows without bound over time. This has the consequence that both prediction and learning become more expensive in terms of computation and memory usage over time, which is an undesirable property for an online learning algorithm. To tackle this issue, approaches have been proposed for incorporating a *budget* on the number of support vectors [32, 126]. These approaches set an upper limit on the number of support vectors which can be retained to describe the solution of the optimisation problem. Various strategies can then be employed for enforcing this budget. The simplest strategy is to remove support vectors, either based on their influence on the solution (*i.e.* remove the support vector with the smallest Lagrange multiplier) or based on their age. Other strategies [126] include projecting the support vector which will be removed onto the remaining support vectors, or merging pairs of support vectors.

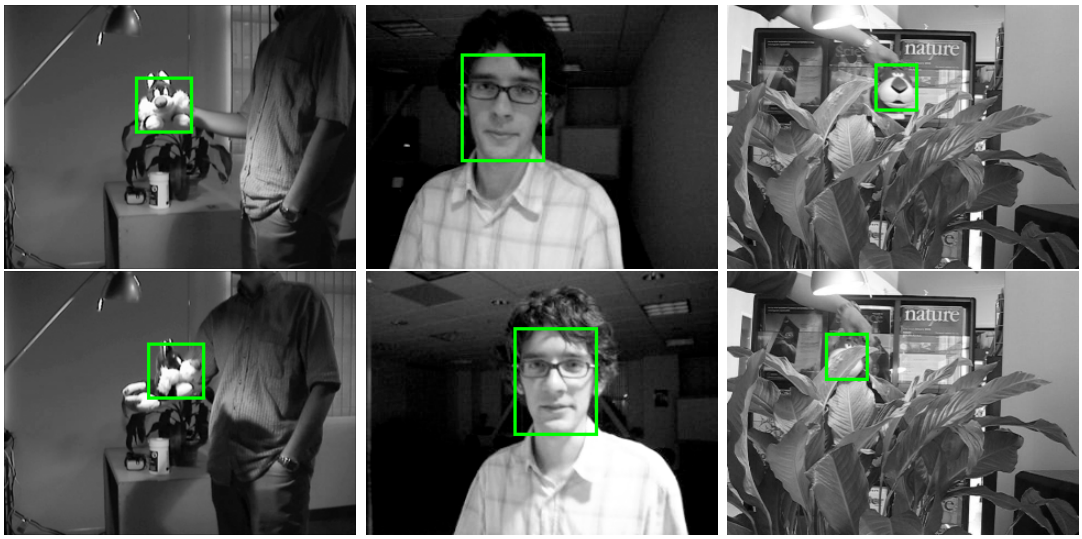
Chapter 3

Struck: Structured Output Tracking With Kernels

3.1 Introduction

Visual object tracking is one of the core problems of computer vision, with wide-ranging applications including human-computer interaction, surveillance and augmented reality, to name just a few. For other areas of computer vision which aim to perform higher-level tasks such as scene understanding and action recognition, object tracking provides an essential component.

For some applications, the object to be tracked is known in advance and it is possible to incorporate prior knowledge when designing the tracker. There are other cases, however, where it is desirable to be able to track arbitrary objects, which may only be specified at runtime. In these scenarios, the tracker must be able to model the appearance of the object on-the-fly and adapt this model during tracking to take into account changes caused by object motion, lighting conditions, and occlusion (as illustrated in Figure 3.1). Even when prior information about the object is known, having a framework with the flexibility to adapt to appearance changes and incorporate new information during tracking is attractive, and in real-world scenarios is often essential for successful tracking.



(a) Object motion **(b)** Lighting **(c)** Partial occlusion

Figure 3.1: Examples of different causes of appearance change of the target object. An adaptive tracking framework is needed in order to handle these appearance changes during tracking.

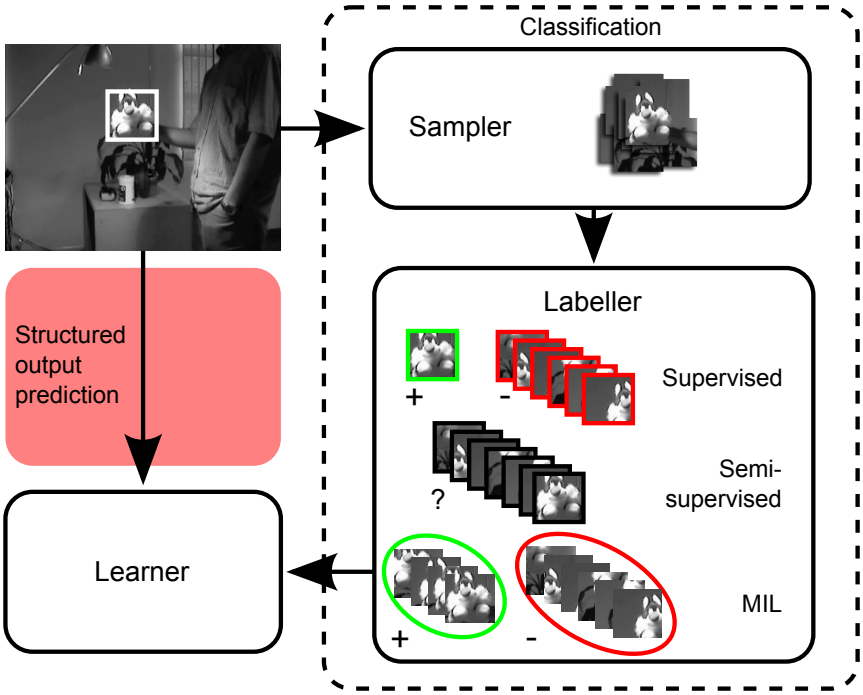


Figure 3.2: Different adaptive tracking-by-detection paradigms: given the current estimated object location, traditional approaches (shown on the right-hand side) generate a set of samples and, depending on the type of learner, produce training labels. Our approach (left-hand side) avoids these steps and operates directly on the tracking output.

An approach to tracking which has become particularly popular recently is tracking-by-detection [6], which treats the tracking problem as a detection task applied over time. This popularity is due in part to the great deal of progress made recently in object detection, with many of the ideas being directly transferable to tracking. Another key factor is the development of methods which allow the classifiers used by these approaches to be trained online, providing a natural mechanism for adaptive tracking [7, 46, 99].

Adaptive tracking-by-detection approaches maintain a classifier trained online to distinguish the target object from its surrounding background. During tracking, this classifier is used to estimate object location by searching for the maximum classification score in a local region around the estimate from the previous frame, typically using a sliding-window approach. Given the estimated object location, traditional algorithms generate a set of binary labelled training samples with which to update the classifier online. As such, these algorithms separate the

adaptation phase of the tracker into two distinct parts: (i) the generation and labelling of samples; and (ii) the updating of the classifier.

While widely used, this separation raises a number of issues. Firstly, it is necessary to design a strategy for generating and labelling samples, and it is not clear how this should be done in a principled manner. The usual approaches rely on predefined rules such as the distance of a sample from the estimated object location to decide whether a sample should be labelled positive or negative. Secondly, the objective for the classifier is to predict the binary label of a sample correctly, while the objective for the tracker is to estimate object location accurately. Because these two objectives are not explicitly coupled during learning, the assumption that the maximum classifier confidence corresponds to the best estimate of object location may not hold (a similar point was raised by Williams *et al.* [128]). State-of-the-art adaptive tracking-by-detection methods mainly focus on improving tracking performance by increasing the robustness of the classifier to poorly labelled samples resulting from this approach. Examples of this include using robust loss functions [70,77], semi-supervised learning [47,100], or multiple-instance learning [7,131].

In this chapter we take a different approach and frame the overall tracking problem as one of structured output prediction, in which the task is to directly predict the change in object location between frames. We present a novel and principled adaptive tracking-by-detection framework which integrates the learning and tracking, avoiding the need for ad-hoc update strategies (see Figure 3.2).

Most recent tracking by detection approaches have used variants of online boosting-based classifiers [7, 46, 99]. In object detection, boosting has proved to be very successful for particular tasks, most notably face detection using the approach of Viola and Jones [123]. Elements of this approach, in particular the Haar-like feature representation, have become almost standard in tracking-by-detection research. The most successful research in object detection, however, has tended to make use of SVMs rather than boosting, due to their good generalisation ability, robustness to label noise, and flexibility in object representation through the use of kernels [14, 40, 122]. Because of this flexibility of SVMs and their natural generalisation to structured output spaces, we make use of the

structured output SVM framework of Tsochantaridis *et al.* [119]. In particular, we extend the online structured output SVM learning method proposed by Bordes *et al.* [15, 17] and adapt it to the task of adaptive object tracking. We find experimentally that the use of our framework results in large performance gains over state-of-the-art tracking by detection approaches.

A structured output SVM framework has previously been applied to the task of object detection by Blaschko and Lampert [14]. In contrast to their work, in our setting there is no offline labelled data available for training (except the first frame which is assumed to be annotated) and instead online learning is used. However, online learning with kernels suffers from the *curse of kernelisation*, whereby the number of support vectors increases with the amount of training data. Therefore, in order to allow for real-time operation, there is a need to control the number of support vectors. Recently, approaches have been proposed for online learning of classification SVMs on a fixed budget [32, 126], meaning that the number of support vectors is constrained to remain within a specified limit. We apply similar ideas in this chapter and introduce a novel approach for budgeting which is suitable for use in an online structured output SVM framework. We find empirically that the introduction of a budget brings large gains in terms of computational efficiency, without impacting significantly on the tracking performance of our system.

3.2 Online structured output tracking

3.2.1 Tracking by detection

In this section we provide an overview of traditional adaptive tracking-by-detection algorithms, which attempt to learn a classifier to distinguish a target object from its local background.

Typically, the tracker maintains an estimate of the position $\mathbf{p} \in \mathcal{P}$ of a 2D bounding box containing the target object within a frame of a video sequence $\mathbf{f}_t \in \mathcal{F}$, where $t = 1, \ldots, T$ is the time. Given a bounding box position $\mathbf{p}$, a classifier is applied to features extracted from an image patch within the bounding

box $\mathbf{x}_t^{\mathbf{p}} \in \mathcal{X}$. The classifier is trained with example pairs $(\mathbf{x}, z)$, where $z = \pm 1$ is a binary label, and makes its predictions according to $\hat{z} = \text{sign}(h(\mathbf{x}))$, where $h : \mathcal{X} \to \mathbb{R}$ is the classification confidence function.

During tracking, it is assumed that a change in position of the target can be estimated by maximising h in a local region around the position in the previous frame. Let $\mathbf{p}_{t-1}$ be the estimated bounding box at time $t-1$. The objective for the tracker is to estimate a transformation (*e.g.* translation) $\mathbf{y}_t \in \mathcal{Y}$ such that the new position of the object is approximated by the composition $\mathbf{p}_t = \mathbf{p}_{t-1} \circ \mathbf{y}_t$. $\mathcal{Y}$ denotes our *search space* and its form depends on the type of motion to be tracked. For most tracking-by-detection approaches this is 2D translation, in which case $\mathcal{Y} = \{(\Delta u, \Delta v) \mid \Delta u^2 + \Delta v^2 < r^2\}$, where r is a search radius. In this case the composition $\mathbf{p}_t = \mathbf{p}_{t-1} \circ \mathbf{y}_t$ is given by $(u_t, v_t) = (u_{t-1}, v_{t-1}) + (\Delta u, \Delta v)$. Mathematically, an estimate is found for the change in position relative to the previous frame according to

$$\mathbf{y}_t = \underset{\mathbf{y} \in \mathcal{Y}}{\text{argmax}} \; h(\mathbf{x}_t^{\mathbf{p}_{t-1} \circ \mathbf{y}}), \tag{3.1}$$

and the tracker position is updated as $\mathbf{p}_t = \mathbf{p}_{t-1} \circ \mathbf{y}_t$.

After estimating the new object position, a set of training examples from the current frame is generated. We separate this process into two components: the *sampler* and the *labeller*. The sampler generates a set of n different transformations $\{\mathbf{y}_t^1, \ldots, \mathbf{y}_t^n\}$, resulting in a set of training examples $\{\mathbf{x}_t^{\mathbf{p}_t \circ \mathbf{y}_t^1}, \ldots, \mathbf{x}_t^{\mathbf{p}_t \circ \mathbf{y}_t^n}\}$. After this process, depending on the classifier type, the labeller chooses labels $\{z_t^1, \ldots, z_t^n\}$ for these training examples. Finally, the classifier is updated using these training examples and labels.

There are a number of issues which are raised by this approach to tracking. Firstly, the assumption made in (3.1) that the classification confidence function provides an accurate estimate of object position is not explicitly incorporated into the learning algorithm, since the classifier is trained only with binary examples and has no information about transformations. Secondly, examples used for training the classifier are all equally weighted, meaning that a negative example which overlaps significantly with the tracker bounding box is treated the same as one which overlaps very little. One implication of this is that slight inaccuracy dur-

ing tracking can lead to poorly labelled examples, which are likely to reduce the accuracy of the classifier, in turn leading to further tracking inaccuracy. Thirdly, the labeller is usually chosen based on intuitions and heuristics, rather than having a tight coupling with the classifier. Mistakes made by the labeller manifest themselves as *label noise*, and many current state-of-the-art approaches try to mitigate this problem by using robust loss functions [70, 77], semi-supervised learning [47, 100], or multiple-instance learning [7, 131]. We argue that all of these techniques, though justified in increasing the robustness of the classifier to label noise, are not addressing the real problem which stems from separating the labeller from the learner. The algorithm which we present does not depend on a labeller and tries to overcome all these problems within a coherent framework by directly linking the learning to tracking and avoiding an artificial binarisation step. Sample selection is fully controlled by the learner itself, and relationships between samples such as their relative similarity are taken into account during learning.

To conclude this section, we describe how a conventional labeller works, as this provides further insight into our algorithm. Traditional labellers use a *transformation similarity function* to determine the label of a sample positioned at $\mathbf{p}_t \circ \mathbf{y}_t^i$. This function can be expressed as $s_{\mathbf{p}_t}(\mathbf{y}_t^i, \mathbf{y}_t^j) \in \mathbb{R}$ which, given a reference position $\mathbf{p}_t$ and two transformations $\mathbf{y}_t^i$ and $\mathbf{y}_t^j$, determines how similar the resulting samples are. For example, the overlap function defined by

$$s^o_{\mathbf{p}_t}(\mathbf{y}_t^i, \mathbf{y}_t^j) = \frac{(\mathbf{p}_t \circ \mathbf{y}_t^i) \cap (\mathbf{p}_t \circ \mathbf{y}_t^j)}{(\mathbf{p}_t \circ \mathbf{y}_t^i) \cup (\mathbf{p}_t \circ \mathbf{y}_t^j)} \tag{3.2}$$

measures the degree of overlap between two bounding boxes. Another example of such a function is based on the distance of two transformations $s^d_{\mathbf{p}_t}(\mathbf{y}_t^i, \mathbf{y}_t^j) = -d(\mathbf{y}_t^i, \mathbf{y}_t^j)$.

Let $\mathbf{y}^0$ denote the identity (or null) transformation, *i.e.* $\mathbf{p} = \mathbf{p} \circ \mathbf{y}^0$. Given a transformation similarity function, the labeller determines the label z_t^i of a sample generated by transformation $\mathbf{y}_t^i$ by applying a *labelling function* $z_t^i = \ell(s_{\mathbf{p}_t}(\mathbf{y}^0, \mathbf{y}_t^i))$.

Most commonly, this can be expressed as

$$\ell(s_{\mathbf{p}_t}(\mathbf{y}^0, \mathbf{y}_t^i)) = \begin{cases} +1 & \text{for} \quad s_{\mathbf{p}_t}(\mathbf{y}^0, \mathbf{y}_t^i) \geq \theta_u \\ -1 & \text{for} \quad s_{\mathbf{p}_t}(\mathbf{y}^0, \mathbf{y}_t^i) < \theta_l \\ 0 & \text{for} \quad \text{otherwise} \end{cases} \tag{3.3}$$

where θ_u and θ_l are upper and lower thresholds, respectively. A binary classifier generally ignores the unlabelled examples [46], while those based on semi-supervised learning use them in their update phase [47,100]. In approaches based on multiple-instance learning [7,131], the labeller collects all the positive examples in a *bag* and assigns a positive label to the bag instead. Most, if not all, variants of adaptive tracking-by-detection algorithms use a labeller which can be expressed in a similar fashion. However, it is not clear how the labelling parameters (*e.g.* the thresholds θ_u and θ_l in the previous example) should be estimated in an online learning framework. Additionally, such heuristic approaches are often prone to noise and it is not clear why such a function is in fact suitable for tracking. In the subsequent section, we will derive our algorithm based on a structured output approach which fundamentally addresses these issues and can be thought of as a generalisation of these heuristic methods.

3.2.2 Structured output SVM

Rather than learning a classifier, we propose learning a prediction function $f : \mathcal{X} \to \mathcal{Y}$ to directly estimate the object transformation between frames. Our output space is thus the space of all transformations $\mathcal{Y}$ instead of the binary labels ± 1. In our approach, a labelled example is a pair $(\mathbf{x}, \mathbf{y})$ where $\mathbf{y}$ is the desired transformation of the target. We learn f in a structured output SVM framework [14,119], which introduces a discriminant function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that can be used for prediction according to

$$\mathbf{y}_t = f(\mathbf{x}_t^{\mathbf{p}_{t-1}}) = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}} \; g(\mathbf{x}_t^{\mathbf{p}_{t-1}}, \mathbf{y}). \tag{3.4}$$

Note the similarity between (3.4) and (3.1): we are performing a maximisation step in order to predict the object transformation, however now the discriminant

function g includes the label $\mathbf{y}$ explicitly, meaning it can be incorporated into the learning algorithm. In our framework, rather than using the tracker position to generate binary examples for training a classifier, we instead provide the single labelled example $(\mathbf{x}_t^{\mathbf{p}_t}, \mathbf{y}^0)$, which is then used to update the learner.

g measures the compatibility between $(\mathbf{x}, \mathbf{y})$ pairs and gives a high score to those which are well matched. By restricting this to be a linear function $g(\mathbf{x}, \mathbf{y}) = \langle \mathbf{w}, \mathbf{\Phi}(\mathbf{x}, \mathbf{y}) \rangle$, where $\mathbf{\Phi}(\mathbf{x}, \mathbf{y})$ is a joint kernel map (to be defined later), it can be learned in a large-margin framework from a set of examples $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$ by minimising the convex objective function

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & \forall i : \ \xi_i \geq 0 \\ & \forall i, \forall \mathbf{y} \neq \mathbf{y}_i : \ \langle \mathbf{w}, \delta\mathbf{\Phi}_i(\mathbf{y}) \rangle \geq \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i \end{aligned} \tag{3.5}$$

where $\delta\mathbf{\Phi}_i(\mathbf{y}) = \mathbf{\Phi}(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{\Phi}(\mathbf{x}_i, \mathbf{y})$. This optimisation aims to ensure that the value of $g(\mathbf{x}_i, \mathbf{y}_i)$ for the training example $(\mathbf{x}_i, \mathbf{y}_i)$ is greater than $g(\mathbf{x}_i, \mathbf{y})$ for $\mathbf{y} \neq \mathbf{y}_i$, by a margin which depends on a loss function Δ. This loss function should satisfy $\Delta(\mathbf{y}, \bar{\mathbf{y}}) = 0$ iff $\mathbf{y} = \bar{\mathbf{y}}$ and increase as $\mathbf{y}$ and $\bar{\mathbf{y}}$ become more dissimilar. The loss function plays an important role in our approach, as it allows us to address the issue raised previously of all samples being treated equally. This can be achieved by making use of the transformation similarity function introduced in Section 3.2.1. For example, as suggested by Blaschko and Lampert [14], we choose to base the loss function on bounding box overlap according to

$$\Delta(\mathbf{y}, \bar{\mathbf{y}}) = 1 - s^o_{\mathbf{p}_t}(\mathbf{y}, \bar{\mathbf{y}}), \tag{3.6}$$

where $s^o_{\mathbf{p}_t}(\mathbf{y}, \bar{\mathbf{y}})$ is the overlap function (3.2).

3.2.3 Online optimisation

To optimise (3.5) in an online setting, we use the approach of Bordes *et al.* [15,17]. Using standard Lagrangian duality techniques, (3.5) can be converted into its equivalent dual form

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i,\mathbf{y}\neq\mathbf{y}_i} \Delta(\mathbf{y},\mathbf{y}_i)\alpha_i^{\mathbf{y}} - \frac{1}{2}\sum_{\substack{i,\mathbf{y}\neq\mathbf{y}_i \\ j,\bar{\mathbf{y}}\neq\mathbf{y}_j}} \alpha_i^{\mathbf{y}}\alpha_j^{\bar{\mathbf{y}}}\langle\delta\boldsymbol{\Phi}_i(\mathbf{y}),\delta\boldsymbol{\Phi}_j(\bar{\mathbf{y}})\rangle \\ \text{s.t.} \quad & \forall i,\forall\mathbf{y}\neq\mathbf{y}_i:\ \alpha_i^{\mathbf{y}} \geq 0 \\ & \forall i:\ \sum_{\mathbf{y}\neq\mathbf{y}_i}\alpha_i^{\mathbf{y}} \leq C \end{aligned} \tag{3.7}$$

and the discriminant function expressed as $g(\mathbf{x},\mathbf{y}) = \sum_{i,\bar{\mathbf{y}}\neq\mathbf{y}_i}\alpha_i^{\bar{\mathbf{y}}}\langle\delta\boldsymbol{\Phi}_i(\bar{\mathbf{y}}),\boldsymbol{\Phi}(\mathbf{x},\mathbf{y})\rangle$. As in the case of classification SVMs, a benefit of this dual representation is that because the joint kernel map $\boldsymbol{\Phi}(\mathbf{x},\mathbf{y})$ only ever occurs inside scalar products, it can be defined implicitly in terms of an appropriate joint kernel function $k(\mathbf{x},\mathbf{y},\bar{\mathbf{x}},\bar{\mathbf{y}}) = \langle\boldsymbol{\Phi}(\mathbf{x},\mathbf{y}),\boldsymbol{\Phi}(\bar{\mathbf{x}},\bar{\mathbf{y}})\rangle$. The kernel functions we use during tracking are discussed in Section 3.2.5.

By reparametrising (3.7) [15] according to

$$\beta_i^{\mathbf{y}} = \begin{cases} -\alpha_i^{\mathbf{y}} & \text{if } \mathbf{y}\neq\mathbf{y}_i \\ \displaystyle\sum_{\bar{\mathbf{y}}\neq\mathbf{y}_i}\alpha_i^{\bar{\mathbf{y}}} & \text{otherwise,} \end{cases} \tag{3.8}$$

the dual can be considerably simplified to

$$\begin{aligned} \max_{\boldsymbol{\beta}} \quad & -\sum_{i,\mathbf{y}}\Delta(\mathbf{y},\mathbf{y}_i)\beta_i^{\mathbf{y}} - \frac{1}{2}\sum_{i,\mathbf{y},j,\bar{\mathbf{y}}}\beta_i^{\mathbf{y}}\beta_j^{\bar{\mathbf{y}}}\langle\boldsymbol{\Phi}(\mathbf{x}_i,\mathbf{y}),\boldsymbol{\Phi}(\mathbf{x}_j,\bar{\mathbf{y}})\rangle \\ \text{s.t.} \quad & \forall i,\forall\mathbf{y}:\ \beta_i^{\mathbf{y}} \leq \delta(\mathbf{y},\mathbf{y}_i)C \\ & \forall i:\ \sum_{\mathbf{y}}\beta_i^{\mathbf{y}} = 0 \end{aligned} \tag{3.9}$$

where $\delta(\mathbf{y},\bar{\mathbf{y}}) = 1$ if $\mathbf{y}=\bar{\mathbf{y}}$ and 0 otherwise. This also simplifies the discriminant function to $g(\mathbf{x},\mathbf{y}) = \sum_{i,\bar{\mathbf{y}}}\beta_i^{\bar{\mathbf{y}}}\langle\boldsymbol{\Phi}(\mathbf{x}_i,\bar{\mathbf{y}}),\boldsymbol{\Phi}(\mathbf{x},\mathbf{y})\rangle$. In this form we refer to those pairs $(\mathbf{x}_i,\mathbf{y})$ for which $\beta_i^{\mathbf{y}}\neq 0$ as *support vectors* and those $\mathbf{x}_i$ included in at least one support vector as *support patterns*. Note that for a given support pattern $\mathbf{x}_i$, only the support vector $(\mathbf{x}_i,\mathbf{y}_i)$ will have $\beta_i^{\mathbf{y}_i} > 0$, while any other support vectors $(\mathbf{x}_i,\mathbf{y})$, $\mathbf{y}\neq\mathbf{y}_i$, will have $\beta_i^{\mathbf{y}} < 0$. We refer to these as positive and negative support vectors respectively.

The core step in the optimisation algorithm of Bordes *et al.* [15, 17] is an SMO-style step [92] which monotonically improves (3.9) with respect to a pair of coefficients $\beta_i^{\mathbf{y}^+}$ and $\beta_i^{\mathbf{y}^-}$. Because of the constraint $\sum_{\mathbf{y}}\beta_i^{\mathbf{y}} = 0$, the coefficients

Require: i, $\mathbf{y}_+$, $\mathbf{y}_-$
1: $k_{00} = \langle \mathbf{\Phi}(\mathbf{x}_i, \mathbf{y}_+), \mathbf{\Phi}(\mathbf{x}_i, \mathbf{y}_+) \rangle$
2: $k_{11} = \langle \mathbf{\Phi}(\mathbf{x}_i, \mathbf{y}_-), \mathbf{\Phi}(\mathbf{x}_i, \mathbf{y}_-) \rangle$
3: $k_{01} = \langle \mathbf{\Phi}(\mathbf{x}_i, \mathbf{y}_+), \mathbf{\Phi}(\mathbf{x}_i, \mathbf{y}_-) \rangle$
4: $\lambda^u = \frac{g_i(\mathbf{y}_+) - g_i(\mathbf{y}_-)}{k_{00} + k_{11} - 2k_{01}}$
5: $\lambda = \max(0, \min(\lambda^u, C\delta(\mathbf{y}_+, \mathbf{y}_i) - \beta_i^{\mathbf{y}_+}))$
6: *Update coefficients*
7: $\beta_i^{\mathbf{y}_+} \leftarrow \beta_i^{\mathbf{y}_+} + \lambda$
8: $\beta_i^{\mathbf{y}_-} \leftarrow \beta_i^{\mathbf{y}_-} - \lambda$
9: *Update gradients*
10: **for** $(\mathbf{x}_j, \mathbf{y}) \in \mathcal{S}$ **do**
11: $\quad k_0 = \langle \mathbf{\Phi}(\mathbf{x}_j, \mathbf{y}), \mathbf{\Phi}(\mathbf{x}_i, \mathbf{y}_+) \rangle$
12: $\quad k_1 = \langle \mathbf{\Phi}(\mathbf{x}_j, \mathbf{y}), \mathbf{\Phi}(\mathbf{x}_i, \mathbf{y}_-) \rangle$
13: $\quad \nabla_j(\mathbf{y}) \leftarrow \nabla_j(\mathbf{y}) - \lambda(k_0 - k_1)$
14: **end for**

Algorithm 3.1: SMOStep

must be modified by opposite amounts, $\beta_i^{\mathbf{y}_+} \leftarrow \beta_i^{\mathbf{y}_+} + \lambda$, $\beta_i^{\mathbf{y}_-} \leftarrow \beta_i^{\mathbf{y}_-} - \lambda$, leading to a one-dimensional maximisation in λ which can be solved in closed form (Algorithm 3.1).

The remainder of the online learning algorithm centres around how to choose the triplet $(i, \mathbf{y}_+, \mathbf{y}_-)$ which should be optimised by this SMO step. For a given i, $\mathbf{y}_+$ and $\mathbf{y}_-$ are chosen to define the feasible search direction with the highest gradient, where the gradient of (3.9) with respect to a single coefficient $\beta_i^{\mathbf{y}}$ is given by

$$\begin{aligned} \nabla_i(\mathbf{y}) &= -\Delta(\mathbf{y}, \mathbf{y}_i) - \sum_{j,\bar{\mathbf{y}}} \beta_j^{\bar{\mathbf{y}}} \langle \mathbf{\Phi}(\mathbf{x}_i, \mathbf{y}), \mathbf{\Phi}(\mathbf{x}_j, \bar{\mathbf{y}}) \rangle \\ &= -\Delta(\mathbf{y}, \mathbf{y}_i) - g(\mathbf{x}_i, \mathbf{y}). \end{aligned} \tag{3.10}$$

Three different update steps are considered, which map very naturally onto a tracking framework:

- ProcessNew Processes a new example $(\mathbf{x}_i, \mathbf{y}_i)$. Because all the $\beta_i^{\mathbf{y}}$ are initially 0, and only $\beta_i^{\mathbf{y}_i} \geq 0$, $\mathbf{y}_+ = \mathbf{y}_i$. $\mathbf{y}_-$ is found according to $\mathbf{y}_- = \operatorname{argmin}_{\mathbf{y} \in \mathcal{Y}} \nabla_i(\mathbf{y})$. During tracking, this corresponds to adding the true label $\mathbf{y}_i$ as a positive support vector and searching for the most important sample to become a negative support vector according to the current state of the learner, taking into account the loss function. Note, however, that

this step does not necessarily add new support vectors, since the SMO step may not need to adjust the $\beta_i^{\mathbf{y}}$ away from 0.

- ProcessOld Processes an existing support pattern $\mathbf{x}_i$ chosen at random. $\mathbf{y}_+ = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}} \nabla_i(\mathbf{y})$, but a feasible search direction requires $\beta_i^{\mathbf{y}} < \delta(\mathbf{y}, \mathbf{y}_i)C$, meaning this maximisation will only involve existing support vectors. As for ProcessNew, $\mathbf{y}_- = \operatorname{argmin}_{\mathbf{y}\in\mathcal{Y}} \nabla_i(\mathbf{y})$. During tracking, this corresponds to revisiting a frame for which we have retained some support vectors and potentially adding another sample as a negative support vector, as well as adjusting the associated coefficients. Again, this new sample is chosen to take into account the current learner state and loss function.

- Optimize Processes an existing support pattern $\mathbf{x}_i$ chosen at random, but only modifies coefficients of existing support vectors. $\mathbf{y}_+$ is chosen as for ProcessOld, and $\mathbf{y}_- = \operatorname{argmin}_{\mathbf{y}\in\mathcal{Y}_i} \nabla_i(\mathbf{y})$, where $\mathcal{Y}_i = \{\mathbf{y} \in \mathcal{Y} \mid \beta_i^{\mathbf{y}} \neq 0\}$.

Of these cases, ProcessNew and ProcessOld are both able to add new support vectors, which gives the learner the ability to perform sample selection during tracking and discover important background elements. This selection involves searching over $\mathcal{Y}$ to minimise $\nabla_i(\mathbf{y})$, which may be a relatively expensive operation. In practice, we found for the 2D translation case it was sufficient to sample from $\mathcal{Y}$ on a polar grid, rather than considering every pixel offset. The Optimize case only considers existing support vectors, so is a much less expensive operation.

As suggested by Bordes *et al.* [17], we schedule these update steps as follows. A Reprocess step is defined as a single ProcessOld step followed by n_O Optimize steps. Given a new training example $(\mathbf{x}_i, \mathbf{y}_i)$ we call a single ProcessNew step followed by n_R Reprocess steps. In practice we typically use $n_O = n_R = 10$.

During tracking, we maintain a set of support vectors $\mathcal{S}$. For each $(\mathbf{x}_i, \mathbf{y}) \in \mathcal{S}$ we store the coefficients $\beta_i^{\mathbf{y}}$ and gradients $\nabla_i(\mathbf{y})$, which are both incrementally updated during an SMO step. If the SMO step results in a $\beta_i^{\mathbf{y}}$ becoming 0, the corresponding support vector is removed from $\mathcal{S}$.

3.2.4 Incorporating a budget

An issue with the approach described thus far is that the number of support vectors is not bounded and in general will increase over time. Evaluating $g(\mathbf{x}, \mathbf{y})$ requires evaluating scalar products (or kernel functions) between $(\mathbf{x}, \mathbf{y})$ and each support vector, which means that both the computational and storage costs grow linearly with the number of support vectors. Additionally, since (3.10) involves evaluating g, both the ProcessNew and ProcessOld update steps will become more expensive as the number of support vectors increases. This issue is particularly important in the case of tracking, as in principle we could be presented with an infinite number of training examples.

Recently a number of approaches have been proposed for online learning of classification SVMs on a fixed budget [32, 126], meaning the number of support vectors cannot exceed a specified limit. If the budget is already full and a new support vector needs to be added, these approaches identify a suitable support vector to remove and potentially adjust the coefficients of the remaining support vectors to compensate for the removal.

We now propose an approach for incorporating a budget into the algorithm presented in Section 3.2.3. Similar to Wang *et al.* [126], we choose to remove the support vector which results in the smallest change to the weight vector $\mathbf{w}$, as measured by $\|\Delta\mathbf{w}\|^2$. However, as with the SMO step used during optimisation, we must also ensure that the constraint $\sum_{\mathbf{y}} \beta_i^{\mathbf{y}} = 0$ remains satisfied. Because of the fact that there only exists one positive support vector for each support pattern, it is sufficient to only consider the removal of negative support vectors during budget maintenance. In the case that a support pattern has only two support vectors, then this will result in them both being removed. Removing the negative support vector $(\mathbf{x}_r, \mathbf{y})$ results in the weight vector changing according to

$$\bar{\mathbf{w}} = \mathbf{w} - \beta_r^{\mathbf{y}} \mathbf{\Phi}(\mathbf{x}_r, \mathbf{y}) + \beta_r^{\mathbf{y}} \mathbf{\Phi}(\mathbf{x}_r, \mathbf{y}_r), \tag{3.11}$$

meaning

$$\begin{aligned}\|\Delta\mathbf{w}\|^2 = \beta_r^{\mathbf{y}^2}\big\{&\langle\mathbf{\Phi}(\mathbf{x}_r,\mathbf{y}),\mathbf{\Phi}(\mathbf{x}_r,\mathbf{y})\rangle + \\ &\langle\mathbf{\Phi}(\mathbf{x}_r,\mathbf{y}_r),\mathbf{\Phi}(\mathbf{x}_r,\mathbf{y}_r)\rangle - 2\langle\mathbf{\Phi}(\mathbf{x}_r,\mathbf{y}),\mathbf{\Phi}(\mathbf{x}_r,\mathbf{y}_r)\rangle\big\}.\end{aligned} \tag{3.12}$$

Each time the budget is exceeded we remove the support vector resulting in the minimum $\|\Delta\mathbf{w}\|^2$. We show in the experimental section that this does not impact significantly on tracking performance, even with modest budget sizes, and improves the efficiency. We name the proposed algorithm *Struck* and show the overall tracking loop in Algorithm 3.2. Our C++ implementation of Struck is publicly available[1].

Require: $\mathbf{f}_t$, $\mathbf{p}_{t-1}$, $\mathcal{S}_{t-1}$

Estimate change in object location
$\mathbf{y}_t = \text{argmax}_{\mathbf{y}\in\mathcal{Y}}\ g(\mathbf{x}_t^{\mathbf{p}_{t-1}},\mathbf{y})$
$\mathbf{p}_t = \mathbf{p}_{t-1}\circ\mathbf{y}_t$
Update discriminant function
$(i,\mathbf{y}_+,\mathbf{y}_-) \leftarrow$ ProcessNew$(\mathbf{x}_t^{\mathbf{p}_t},\mathbf{y}^0)$
SMOStep$(i,\mathbf{y}_+,\mathbf{y}_-)$
BudgetMaintenance()
for $j = 1$ to n_R **do**
  $(i,\mathbf{y}_+,\mathbf{y}_-) \leftarrow$ ProcessOld()
  SMOStep$(i,\mathbf{y}_+,\mathbf{y}_-)$
  BudgetMaintenance()
  for $k = 1$ to n_O **do**
    $(i,\mathbf{y}_+,\mathbf{y}_-) \leftarrow$ Optimize()
    SMOStep$(i,\mathbf{y}_+,\mathbf{y}_-)$
  end for
end for
return $\mathbf{p}_t$, $\mathcal{S}_t$

Algorithm 3.2: Struck tracking loop.

[1] `http://www.samhare.net/research`

3.2.5 Kernel functions and image features

The use of a structured output SVM framework provides great flexibility in how images are actually represented. In practice we choose to use a restriction kernel [14] which uses the relative bounding box location $\mathbf{y}$ to crop a patch from a frame $\mathbf{x}_t^{\mathbf{p}\circ\mathbf{y}}$, allowing a standard image kernel to be applied between pairs of such patches

$$k_{xy}(\mathbf{x}, \mathbf{y}, \bar{\mathbf{x}}, \bar{\mathbf{y}}) = k(\mathbf{x}^{\mathbf{p}\circ\mathbf{y}}, \bar{\mathbf{x}}^{\bar{\mathbf{p}}\circ\bar{\mathbf{y}}}). \tag{3.13}$$

The use of kernels makes it straightforward to incorporate different image features into our approach, and in our experiments we consider a number of examples. We also investigate using multiple kernels in order to combine different image features together.

3.3 Experiments

3.3.1 Tracking-by-detection benchmarks

Our first set of experiments aims to compare the results of the proposed approach with existing tracking-by-detection approaches. The majority of these are based around boosting or random forests and use simple Haar-like features as their image representation. We use similar features for our evaluation in order to provide a fair comparison and isolate the effect of the learning framework, but note that these features were specifically designed to work with the feature-selection capability of boosting, having been originally introduced by Viola and Jones [123]. Even so, we find that with our framework we are able to significantly outperform the existing state-of-the-art results.

We use 6 different types of Haar-like feature arranged on a grid at 2 scales on a 4×4 grid, resulting in 192 features, with each feature normalised to give a value in the range $[-1, 1]$. The reason for using a grid, as opposed to random locations, is partly to limit the number of random factors in the tracking algorithm, since the learner itself has a random element, and partly to compensate for the fact that we do not perform feature selection. Note, however, that the number of

features we use is lower than systems against which we compare, which use at least 250. We concatenate the feature responses into a feature vector $\mathbf{x}$ and apply a Gaussian kernel $k(\mathbf{x}, \bar{\mathbf{x}}) = \exp(-\sigma\|\mathbf{x}-\bar{\mathbf{x}}\|^2)$, with $\sigma = 0.2$ and $C = 100$ which is fixed for all sequences. Like the systems against which we compare, we track 2D translation $\mathcal{Y} = \{(\Delta u, \Delta v) \mid \Delta u^2 + \Delta v^2 < r^2\}$. During tracking we use a search radius $r = 30$ pixels, though when updating the classifier we take a larger radius $r = 60$ to ensure stability. As mentioned in Section 3.2.3, we found empirically that searching $\mathcal{Y}$ exhaustively when performing online learning was unnecessary, and it is sufficient to sample from $\mathcal{Y}$ on a polar grid (we use 5 radial and 16 angular divisions, giving 81 locations).

To assess tracking performance, we use the Pascal VOC overlap criterion as suggested by Saffari *et al.* [99] and report the average overlap between estimated and ground truth throughout each sequence. Because of the randomness involved in our learning algorithm, we repeat each sequence 5 times with different random seeds and report the median result.

Table 3.1 shows the results obtained by our tracking framework for various budget sizes B, along with published results from existing state-of-the-art approaches [2, 7, 46, 69, 99], and example frames can be seen in Figure 3.3. It can be seen from these results that Struck outperforms the current state-of-the-art on almost every sequence, often by a considerable margin. These results also demonstrate that the proposed budgeting mechanism does not impact significantly on tracking results. Even when the budget is reduced as low as $B = 20$ we outperform the state-of-the-art on 4 out of 8 sequences.

In Figure 3.4 we show some examples of the support vector set $\mathcal{S}$ at the end of tracking. An interesting property which can be observed is that the positive support vectors (shown with green borders) provide a compact summary of the change in object appearance observed during tracking. In other words, our tracker is able to identify distinct appearances of the object over time. Additionally, it is clear that the algorithm automatically chooses more negative support vectors than positive. This is mainly because the foreground can be expressed more compactly than the background, which has higher diversity. We also see from these figures that the budgeting mechanism we use maintains support vectors from the

Sequence	Struck_{∞}	Struck_{100}	Struck_{50}	Struck_{20}	MIForest	OMCLP	MIL	Frag	OAB
coke	**0.57**	**0.57**	0.56	<u>0.52</u>	0.35	0.24	0.33	0.08	0.17
david	0.80	0.80	**0.81**	0.35	0.72	0.61	0.57	0.43	0.26
face1	0.86	0.86	0.86	0.81	0.77	0.80	0.60	**0.88**	0.48
face2	**0.86**	**0.86**	**0.86**	<u>0.83</u>	0.77	0.78	0.68	0.44	0.68
girl	**0.80**	**0.80**	**0.80**	<u>0.79</u>	0.71	0.64	0.53	0.60	0.40
sylvester	**0.68**	**0.68**	0.67	0.58	0.59	0.67	0.60	0.62	0.52
tiger1	**0.70**	**0.70**	0.69	<u>0.68</u>	0.55	0.53	0.52	0.19	0.23
tiger2	0.56	**0.57**	0.55	0.39	0.53	0.44	0.53	0.15	0.28
Average FPS	12.1	13.2	16.2	21.4					

Table 3.1: Average bounding box overlap on benchmark sequences. The first four columns correspond to our method with different budget size indicated by the subscript, and the rest of the columns show published results from state-of-the-art approaches. The best performing method is shown in bold. We also show underlined the cases when Struck with the smallest budget size ($B = 20$) outperforms the state-of-the-art. The last row gives the average number of frames per second using our C++ implementation.

(a) *coke*

(b) *david*

(c) *face1*

(d) *face2*

(e) *girl*

(f) *sylvester*

(g) *tiger1*

(h) *tiger2*

Figure 3.3: Example frames from benchmark tracking sequences, showing the results of Struck compared with MILTrack [7], OMCLP [99] and OAB [46]. Videos of these results can be found at `http://www.samhare.net/research`.

(a) *girl* **(b)** *david* **(c)** *sylvester*

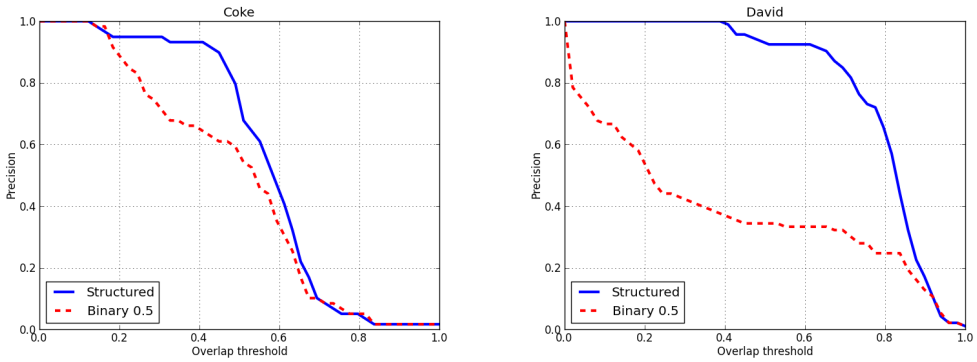
Figure 3.4: Visualisation of the support vector set $\mathcal{S}$ at the end of tracking with $B = 64$ (chosen for illustrative purposes). Each patch shows $\mathbf{x}_t^{\mathbf{p}\circ\mathbf{y}}$, and positive and negative support vectors have green and red borders respectively. Notice that the positive support vectors capture the change in appearance of the target object during tracking.

entire tracking sequence and does not discard old appearance information. We believe that this contributes to the strong performance of our tracker, as it helps prevent drift during tracking which could occur if old information was discarded.

3.3.2 Effect of structured learning

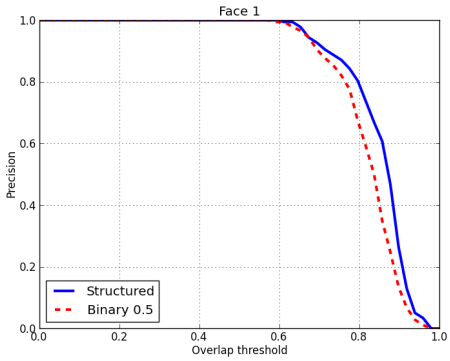
To investigate the importance of structured learning on our results, we next perform a set of experiments against a baseline classification SVM. To achieve this we modify our tracking framework such that the learner is no longer trained using structured examples, but rather using a set of binary examples. Each frame a single positive example is generated using the current tracker state, and negative examples are generated by sampling from $\mathcal{Y}$ as in Section 3.3.1 and taking those which have an overlap of less than 0.5 with the tracker state (*i.e.* $\theta_u = 1$ and $\theta_l = 0.5$ using the labelling function (3.3)). All other factors are kept the same, meaning both approaches use the same image features as in Section 3.3.1 and both use a budget size $B = 100$.

Figure 3.5 shows precision plots for these two tracking approaches on each of the benchmark test sequences from Section 3.3.1. These plots show the percentage of frames for which the overlap between the ground truth bounding box and tracker bounding box is greater then a particular threshold, which provides a

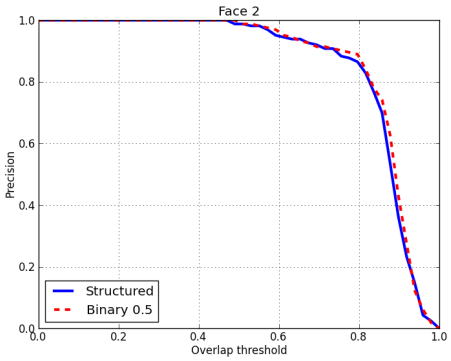


(a) *coke*

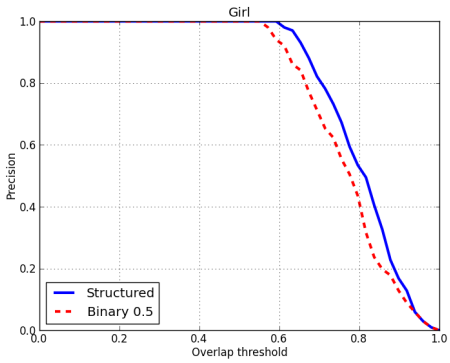
(b) *david*



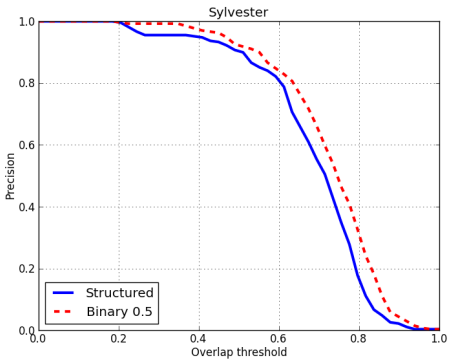
(c) *face1*



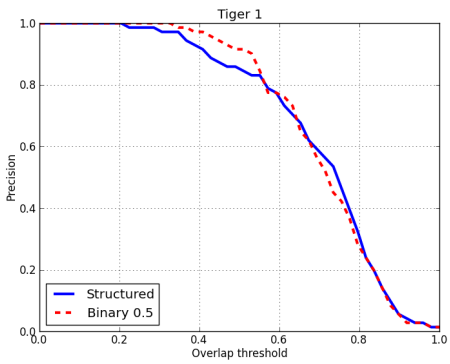
(d) *face2*



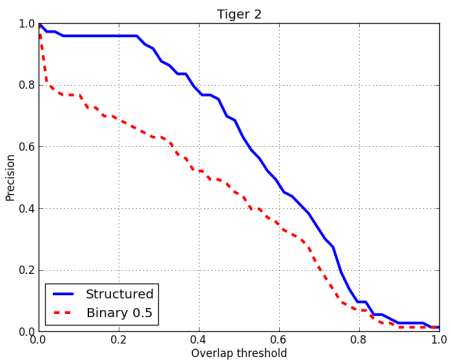
(e) *girl*



(f) *sylv*



(g) *tiger1*



(h) *tiger2*

Figure 3.5: Precision plots comparing the results of tracking using our structured SVM framework with a baseline classification SVM. These plots show the percentage of frames for which the overlap between the ground truth bounding box and tracker bounding box is greater then a particular threshold

more detailed view of the tracker performance than the average overlap used in the previous section. As before, we run each tracker 5 times on the sequence and compute the median precision for a given overlap threshold to produce these plots.

We can see from these results that overall the precision curves for the structured SVM are better than or roughly equivalent to those for the classification SVM, which demonstrate that the structured learning framework we use is able to produce gains in accuracy over a traditional classification-based approach. These gains are most notable on the more challenging sequences such as *coke*, *david* and *tiger2*, for which the classification SVM does not perform particularly well.

In many cases, however, we see that the performance of the two tracking approaches are quite similar. This indicates that a large part of the performance gains observed in Section 3.3.1 can be attributed to our use of a kernelised SVM rather than a boosting-based classifier. Nevertheless, we can still observe that structured learning is able to bring additional performance gains, and importantly it removes the need for introducing a binary labelling strategy, providing a more tightly integrated approach to learning in a tracking context.

3.3.3 Combining kernels

A benefit of the framework we have presented is that it is straightforward to use different image features by modifying the kernel function used for evaluating patch similarity. In addition, different features can be combined by averaging multiple kernels: $k(\mathbf{x}, \bar{\mathbf{x}}) = \frac{1}{N_k} \sum_{i=1}^{N_k} k^{(i)}(\mathbf{x}^{(i)}, \bar{\mathbf{x}}^{(i)})$. Such an approach can be considered a basic form of multiple kernel learning (MKL), and indeed it has been shown [44] that in terms of performance full MKL (in which the relative weighting of the different kernels is learned from training data) does not provide a great deal of improvement over this simple approach.

In addition to the Haar-like features and Gaussian kernel used in Section 3.3.1, we also consider the following features:

- Raw pixel features obtained by scaling a patch to 16×16 pixels and taking the greyscale value (in the range $[0, 1]$). This gives a 256-D feature vector,

Sequence	A	B	C	A+B	A+C	B+C	A+B+C
coke	0.57	0.67	0.69	0.62	0.65	0.68	0.63
david	0.80	0.83	0.67	0.84	0.68	**0.87**	**0.87**
face1	0.86	0.82	0.86	0.82	**0.87**	0.82	0.83
face2	0.86	0.79	0.79	0.83	**0.86**	0.78	0.84
girl	0.80	0.77	0.68	0.79	**0.80**	0.79	0.79
sylvester	0.68	0.75	0.72	0.73	0.72	**0.77**	0.73
tiger1	0.70	0.69	0.77	0.69	0.74	0.74	0.72
tiger2	0.57	0.50	0.61	0.53	**0.63**	0.57	0.56
Average	0.73	0.73	0.72	0.73	0.74	**0.75**	**0.75**

Table 3.2: Combining kernels. A: Haar features with Gaussian kernel ($\sigma = 0.2$); B: Raw features with Gaussian kernel ($\sigma = 0.1$); C: Histogram features with intersection kernel. The bold shows when multiple kernels improve over the best performance of individual kernels, while the underline shows the best performance within the individual kernels. The last row shows the average of each column.

which is combined with a Gaussian kernel with $\sigma = 0.1$.

- Histogram features obtained by concatenating 16-bin intensity histograms from a spatial pyramid of 4 levels. At each level L, the patch is divided into $L \times L$ cells, resulting in a 480-D feature vector. This is combined with an intersection kernel: $k(\mathbf{x}, \bar{\mathbf{x}}) = \frac{1}{D}\sum_{i=1}^{D} \min(x_i, \bar{x}_i)$.

Table 3.2 shows tracking results on the same benchmark videos, with $B = 100$ and all other parameters as specified in Section 3.3.1. It can be seen that the behaviour of the individual features are somewhat complementary. In many cases, combining multiple kernels seems to improve results. However, it is also noticeable that the performance gains are not significant for some sequences. This could be because of our naïve kernel combination strategy and as has been shown by other researchers, *e.g.* [46], feature selection plays a major role in online tracking. Therefore, further investigation into full MKL could potentially result in further improvements.

3.4 Summary

In this chapter, we have presented a new adaptive tracking-by-detection framework based on structured output prediction. Unlike existing methods based on

classification, our algorithm does not rely on a heuristic intermediate step for producing labelled binary samples with which to update the classifier, which is often a source of error during tracking. Our approach uses an online structured output SVM learning framework, making it easy to incorporate image features and kernels. From a learning point of view, we take advantage of the well-studied large-margin theory of SVMs, which brings benefits in terms of generalisation and robustness to noise (both in the input and output spaces). To prevent unbounded growth in the number of support vectors, and allow real-time performance, we also introduced a budget maintenance mechanism for online structured output SVMs. We showed experimentally that our algorithm gives superior performance compared to state-of-the-art trackers.

We believe that the structured output framework we presented provides a very rich platform for incorporating advanced concepts into tracking. For example, it would be relatively straightforward to extend the output space to include rotation and scale transformations. It would also be possible to incorporate object dynamics into this model. While these extensions focus on the output space, the input space could also be enriched through the use of alternative image features and multiple kernel learning.

The framework we have presented does have some limitations. One issue is that if errors are made during tracking, the self-training approach we employ means that it is possible for bad information to be incorporated into the appearance model during learning. We believe that our method is quite robust to this situation as the set of support vectors allows the appearance model to capture multiple modalities of the target object, however there is no explicit means for identifying and discarding such erroneous information. Fundamentally, as discussed in Section 2.1, this is a difficulty which is faced by all adaptive tracking approaches, since true supervision only occurs at the point of initialisation. Another issue is that at present we use exhaustive search in order to predict the change in tracker location according to (3.4), meaning our approach is relatively computationally expensive. It should be possible, however, to incorporate a gradient-based approach for prediction into our framework, which would result in significant performance gains.

Chapter 4

Efficient Online Structured Output Learning for Keypoint-Based Object Tracking

4.1 Introduction

Keypoint-based object detection has become a cornerstone of modern computer vision, enabling great advances in areas such as augmented reality (AR) and simultaneous localisation and mapping (SLAM). These object detection approaches model an object as a set of keypoints, which are matched independently in an input image. Robust estimation procedures based on RANSAC [29, 41, 118] are then used to determine geometrically consistent sets of matches which can be used to infer the presence and transformation of the object.

There has been a great deal of progress in making these approaches suitable for real-time applications and there are now a range of methods available for use on a desktop PC [10,71,87]. Recently, there has been significant interest in developing approaches suitable for low-powered mobile devices such as smartphones and tablets, which are becoming increasingly popular platforms for computer vision applications [25, 72, 96, 116]. These approaches focus on making the matching stage as efficient as possible, since this is generally the most time-consuming part of the detection pipeline. To achieve this they design image descriptors which can be represented as binary vectors, allowing matching to be performed very efficiently by measuring Hamming distance between descriptors, which can be implemented using binary CPU instructions.

The object models built by traditional approaches are static, usually constructed offline for a particular object. For certain applications like AR and SLAM, however, we want to detect the object repeatedly in a dynamic environment. Additionally, some applications require on-the-fly learning and detection to build an instantaneous model from only a single snapshot of the object. Therefore it is desirable to be able to learn an object model efficiently online and adapt it to a particular environment, which is not typically addressed by traditional approaches. This process of adapting or learning the model should not add significant overhead to the detection pipeline and should still be suitable for real-time detection on low-powered devices. These requirements create a very challenging problem for a learning algorithm.

The approach we propose in this chapter frames the entire object detection procedure as a structured learning problem, such that overall detection performance can be optimised given a set of training images. Our formulation combines feature learning, matching, and pose estimation into a single unified framework. Furthermore, because we use a linear structured SVM to perform learning, we are able to perform training online, which allows us to quickly adapt our model to a given environment. Additionally, we show that we can accurately approximate our model during evaluation in such a way that we can take advantage of binary descriptors and the efficiency they provide. As a result, our algorithm adds a relatively small amount of computational overhead compared to static models, while improving the detection rate significantly.

4.2 Motivation and related work

Keypoint-based methods for geometric object detection generally follow a two stage approach:

1. Finding a set of 2D correspondences between an object model and an input image.

2. Estimating the transformation of the object in the image using a robust geometric verification method based on hypotheses generated from the correspondences (*e.g.* RANSAC and its variants).

Generally these two stages are considered as separate problems, and many algorithms focus on improving the object detection quality by employing robust methods for each of these steps individually.

To find the appearance-based 2D correspondences, there are two approaches: matching and classification. Matching-based approaches [10, 25, 72, 75] use descriptors to store a signature for each model keypoint in a database. These descriptors are designed to be invariant to various geometric and photometric transformations and can then be matched given a suitable distance metric to keypoints in an image in a nearest-neighbour fashion.

Classification-based approaches [71,87,116] treat matching as multi-class classification, in which the task is to classify each image keypoint as either background or a particular keypoint from the model. These classifiers are learned offline from training examples of the object observed under various geometric and photometric transformations (usually generated synthetically) and are therefore tuned to the specific object and how individual keypoints might appear in an image. The training algorithm and the number of training examples determine the computational complexity of the learning stage.

Since classification-based approaches rely on an expensive training stage as well as the availability of a 2D/3D object model at training time, these approaches cannot easily be used for on-the-fly detection and tracking of arbitrary objects. This particular problem of the classification-based approaches limit their applicability in practice.

Özuysal *et al.* [88] propose an approach for learning a classification-based model at runtime, by using online random forests to reduce training time. However, this approach is still too computationally expensive to be useful on low-powered devices and also does not continue to adapt the model after the initial training phase. The method most related to our own work is that proposed by Grabner *et al.* [48], in which keypoint classifiers are learned online by using Haar features and an online boosting algorithm. This approach relies on the fact that the geometric verification step can be used in order to provide labels for updating the classifiers in an online manner, allowing for adaptive tracking-by-detection.

To the best of our knowledge, all previous methods involving learning treat the generation of correspondences and estimation of object transformation separately. In this chapter, we propose a novel approach which combines these two steps into a coherent structured learning framework. In this formulation, correspondence generation, learning, and transformation estimation all work together in a unified optimisation formulation with the goal of performing object detection robustly. Our approach proposes an alternative view on keypoint-based object detection where the transformation estimation algorithm operates as the maximisation step of a structured prediction framework. Unlike the online boosting approach of Grabner *et al.* [48], our formulation is also capable of incorporating any kind of

keypoint descriptor into its learning process and is specifically targeted towards low-powered devices.

Structured output prediction was introduced to the computer vision community by Blaschko and Lampert [14] for the task of 2D sliding-window object localisation. In Chapter 3 we have seen how a similar approach can be taken which uses online learning to perform adaptive 2D tracking-by-detection. The work in this chapter is different from these approaches because we are now interested in object detection and tracking under a much larger class of transformations such as 3D pose or homography, and as a result we propose using RANSAC in order to perform structured prediction.

There has recently been significant research interest focusing on object detection for low-powered portable platforms such as smartphones. In particular, highly efficient methods such as BRIEF [25] and BRISK [72] have been developed for descriptor matching. Both of these methods perform simple binary pixel-based tests on keypoints in order to build binary descriptors. By representing these descriptors as bitsets and measuring similarity using the Hamming distance, matching can be performed extremely efficiently using bitwise operations which are well-supported by modern CPUs. We show how the internal representation of our algorithm can be approximated to take advantage of these binary descriptors, making our approach also suitable for low-powered devices.

4.3 Structured learning formulation

In this section, we describe our formulation of keypoint-based object detection as a structured learning problem.

4.3.1 RANSAC for structured prediction

Given an object model M and an input image I, the goal of object detection is to compute a transformation $T \in \mathcal{T}$ which maps M to I. A 3D pose or 2D homography are examples of such a transformation.

We can think of this process as one of structured prediction, with the output

space consisting of all valid transformations, along with a null transformation indicating the absence of the object. We therefore assume that there exists a function $T = f(M, I)$ and that this function can be expressed as

$$T = \underset{T' \in \mathcal{T}}{\operatorname{argmax}}\, g(M, I, T'), \tag{4.1}$$

where g is a compatibility function, scoring all possible transformations of the object given an image.

In practice, finding a solution for the prediction function (4.1) under a specific model definition is generally unfeasible because the output space is very large, and evaluating image observations under different transformations of the model will be expensive. The way that this issue is usually handled is by applying an iterative robust parameter estimation algorithm such as RANSAC [41] or PROSAC [29] to approximately solve (4.1). These algorithms rely on a sparse representation for the model and image and use a set of correspondences between model and image points as their input.

Consider an object model M which is based on a sparse set of keypoints $\mathcal{M} = \{u_1, \ldots, u_J\}$, with each keypoint defined by a location (2D or 3D). Similarly, let the image I be represented as a sparse set of keypoints $\mathcal{I} = \{v_1, \ldots, v_K\}$. A set of correspondences $\mathcal{C} = \{(u_j, v_k, s_{jk}) | u_j \in \mathcal{M}, v_k \in \mathcal{I}, s_{jk} \in \mathbb{R}\}$ is found between model keypoints and image keypoints, where s_{jk} is a correspondence score derived from appearance information. Traditional RANSAC defines a score for a given transformation in terms of the number of inliers

$$g(\mathcal{C}, T) = \sum_{(u_j, v_k) \in \mathcal{C}} \mathbb{I}(\|v_k - T(u_j)\|_2 < \tau), \tag{4.2}$$

where $T(u_j)$ is the location of model keypoint u_j under the transformation T, τ is a spatial mis-alignment threshold and $\mathbb{I}(.)$ is an indicator function. This score is then used as the compatibility function in (4.1) and maximised approximately by randomly sampling transformations which are compatible with minimal subsets of correspondences in $\mathcal{C}$. Variants such as PROSAC use the correspondence scores s_{jk} to bias this sampling in order to reach a solution in fewer iterations.

Existing approaches have applied learning in an offline setting [71, 87, 116]

as well as in an online setting [48, 88] to encourage reliable appearance-based correspondences to be found in $\mathcal{C}$. However, in these approaches the generation and scoring of correspondences and the maximisation of (4.2) are decoupled from each other. These approaches therefore do not perform learning which takes into account the entire transformation prediction process.

To allow learning for the entire prediction process, we propose introducing a weight vector $\mathbf{w}_j$ for each model keypoint u_j. This weight vector is used to score correspondences according to $s_{jk} = \langle \mathbf{w}_j, \mathbf{d}_k \rangle$, where $\mathbf{d}_k$ is a descriptor extracted around image keypoint v_k, normalised such that $\|\mathbf{d}_k\|_2 = 1$. We then propose modifying the compatibility function (4.2) to include correspondence scores, such that it can be written as a linear operator

$$\begin{aligned} g_{\mathbf{w}}(\mathcal{C}, T) &= \sum_{(u_j, v_k) \in \mathcal{C}} s_{jk} \, \mathbb{I}(\|v_k - T(u_j)\|_2 < \tau) \\ &= \langle \mathbf{w}, \mathbf{\Phi}(\mathcal{C}, T) \rangle, \end{aligned} \tag{4.3}$$

where $\mathbf{w} = [\mathbf{w}_1, \ldots, \mathbf{w}_J]^T$ is the concatenation of model weight vectors and $\mathbf{\Phi}(\mathcal{C}, T) = [\boldsymbol{\phi}_1(\mathcal{C}, T), \ldots, \boldsymbol{\phi}_J(\mathcal{C}, T)]^T$ is a joint feature mapping. Each $\boldsymbol{\phi}_j$ is defined as

$$\boldsymbol{\phi}_j(\mathcal{C}, T) = \begin{cases} \mathbf{d}_k & \exists (u_j, v_k) \in \mathcal{C} : \|v_k - T(u_j)\|_2 < \tau \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{4.4}$$

Our goal is to learn the compatibility function (4.3) parameterised by $\mathbf{w}$ such that the behaviour of this function in the output space is close to the actual behaviour of RANSAC, but, because it includes information about appearance, in the process of learning we will discover which model points are the most discriminative and how best we can utilise them to predict transformations.

4.3.2 Structured SVM learning

Now, given a set of training examples $\{(I_i, T_i)\}_{i=1}^N$, $\mathbf{w}$ can be learned in a maximum-margin structured learning framework [119]. For each training example i, this formulation tries to maximise the margin between the score of the true transformation T_i and all alternative transformations. This can be expressed by the

following optimisation problem

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi}} \quad & \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \xi_i \\ \text{s.t.} \quad & \forall i:\ \xi_i \geq 0 \\ & \forall i, \forall T \neq T_i:\ \langle \mathbf{w}, \delta\boldsymbol{\Phi}_i(T) \rangle \geq \Delta(T_i, T) - \xi_i \end{aligned} \tag{4.5}$$

where $\delta\boldsymbol{\Phi}_i(T) = \boldsymbol{\Phi}(\mathcal{C}_i, T_i) - \boldsymbol{\Phi}(\mathcal{C}_i, T)$, and λ is a parameter determining the trade-off between training set accuracy and regularisation. $\Delta(T_i, T)$ is a loss function which measures the penalty for choosing T instead of the true transformation T_i. The loss function $\Delta(T_i, T)$ should measure the dissimilarity of two competing transformation hypotheses and will be discussed in Section 4.3.3.

Because we are using RANSAC to perform structured prediction and this relies on an accurate set of correspondences, we modify this formulation to also encourage each inlier correspondence to score higher than any other image correspondence. This can be realised as an additional set of ranking constraints and the formulation then becomes

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi},\boldsymbol{\gamma}} \quad & \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \xi_i + \nu \sum_{i=1}^{N} \sum_{(u_j, v_k) \in \mathcal{C}_i^*} \gamma_{ij} \\ \text{s.t.} \quad & \forall i:\ \xi_i \geq 0 \\ & \forall i, \forall T \neq T_i:\ \langle \mathbf{w}, \delta\boldsymbol{\Phi}_i(T) \rangle \geq \Delta(T_i, T) - \xi_i \\ & \forall i, \forall j:\ \gamma_{ij} \geq 0 \\ & \forall i, \forall (u_j, v_k), \forall k' \neq k : \langle \mathbf{w}_j, \mathbf{d}_k - \mathbf{d}_{k'} \rangle \geq 1 - \gamma_{ij} \end{aligned} \tag{4.6}$$

where $\mathcal{C}_i^* \subset \mathcal{C}_i$ is the set of inlier correspondences under T_i, and ν is a weighting parameter.

The learning problem presented in (4.6) allows us to train a discriminative model in a unified way in which learning the representation of model points and performing pose estimation are combined in a single structured learning framework.

4.3.3 Loss functions

The optimisation problem (4.6) requires a loss function Δ to be defined between two transformations. We consider a number of possible loss functions, which we compare experimentally in Section 4.4.1.

The first loss function we consider is designed specifically for the case where the transformations are projective homographies. Given two homographies T and T', we define a distance

$$d_{homography}(T, T') = \frac{1}{4} \sum_{i=1}^{4} \|\mathbf{c}_i - (TT'^{-1})(\mathbf{c}_i)\|_2, \tag{4.7}$$

where $\{\mathbf{c}_i\}_{i=1}^4 = \{(-1,-1)^\mathsf{T}, (1,-1)^\mathsf{T}, (-1,1)^\mathsf{T}, (1,1)^\mathsf{T}\}$ are the corners of a square. This distance can become arbitrarily large, so we define a loss function using a truncated version:

$$\Delta_{homography}(T, T') = \min(d_{homography}(T, T'),\ 20). \tag{4.8}$$

A potential issue with this loss function is that since the compatibility function $g_{\mathbf{w}}(\mathcal{C}, T)$ sums over those correspondences in $\mathcal{C}$ which are inliers under T, transformations with more inliers are likely to score higher than those with a smaller number of inliers. For this reason we also consider loss functions which take into account the fact that transformations will have different numbers of inliers. We define two such loss functions, which are applicable for all classes of transformations (*i.e.* not only homographies):

1. Hamming distance on inliers:

$$\Delta_{hamming}(T, T') = \sum_{(u_j, v_k) \in \mathcal{C}} \mathbb{I}\big(z(u_j, v_k, T) \neq z(u_j, v_k, T')\big), \tag{4.9}$$

   where $z(u_j, v_k, T) = \mathbb{I}(\|v_k - T(u_j)\|_2 < \tau)$. This loss function aims to penalise transformations having different inlier sets.

2. Difference in number of inliers:

$$\Delta_{inliers}(T, T') = |g(\mathcal{C}, T) - g(\mathcal{C}, T')|, \tag{4.10}$$

where g is the RANSAC scoring function (4.2). This loss function aims to penalise transformations with different numbers of inliers, similar in spirit to the traditional RANSAC approach.

4.3.4 Online learning

While (4.6) can be solved offline as a batch problem, we are interested in applying our approach for adaptive tracking-by-detection, and therefore need a means for updating $\mathbf{w}$ online. Because we are using a linear structured SVM, this can be readily achieved using stochastic gradient descent. We first rewrite the optimisation problem (4.6) in unconstrained form as

$$\begin{aligned}\min_{\mathbf{w}}\Big\{\frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N}\big(\max_{T\neq T_i}\{\Delta(T_i,T) - \langle \mathbf{w}, \delta\mathbf{\Phi}_i(T)\rangle\}\big)_+ + \\ \nu\sum_{i=1}^{N}\sum_{(u_j,v_k)\in\mathcal{C}_i^*}\big(\max_{k'\neq k}\{1 - \langle \mathbf{w}_j, \mathbf{d}_k - \mathbf{d}_{k'}\rangle\}\big)_+\Big\}\end{aligned} \tag{4.11}$$

where $(.)_+ = \max\{0, .\}$ is the hinge function. Given a training example (I_t, T_t) at time t, a subgradient of (4.11) is found with respect to $\mathbf{w}$, and a gradient descent step is then performed according to

$$\begin{aligned}\mathbf{w}_j^{t+1} \leftarrow &(1-\eta_t\lambda)\mathbf{w}_j^t + \\ &\mathbb{I}(\max_{T\neq T_t}\{\Delta(T_t,T) - \langle \mathbf{w}^t, \delta\mathbf{\Phi}_t(T)\rangle\} > 0)\eta_t\boldsymbol{\alpha}_j^t + \\ &\mathbb{I}(u_j \in \mathcal{C}_t^*)\,\mathbb{I}(\max_{k'\neq k}\{1 - \langle \mathbf{w}_j^t, \mathbf{d}_k - \mathbf{d}_{k'}\rangle\} > 0)\eta_t\nu\boldsymbol{\beta}_j^t,\end{aligned} \tag{4.12}$$

where $\eta_t = 1/\lambda t$ is the step size. Let $\hat{T} = \text{argmax}_{T\neq T_t}\{\Delta(T_t,T) - \langle \mathbf{w}^t, \delta\mathbf{\Phi}_t(T)\rangle\}$ and $\hat{k} = \text{argmax}_{k'\neq k}\{1 - \langle \mathbf{w}_j^t, \mathbf{d}_k - \mathbf{d}_{k'}\rangle\}$. Then $\boldsymbol{\alpha}_j^t$ and $\boldsymbol{\beta}_j^t$ are defined as

$$\boldsymbol{\alpha}_j^t = \boldsymbol{\phi}_j(\mathcal{C}_t, T_t) - \boldsymbol{\phi}_j(\mathcal{C}_t, \hat{T}), \tag{4.13}$$

and

$$\boldsymbol{\beta}_j^t = \mathbf{d}_k - \mathbf{d}_{\hat{k}}. \tag{4.14}$$

To estimate T_t for the current image, we use the prediction of (4.1) given

the old model representation $\mathbf{w}_{t-1}$, and we then update the model representation by performing a single stoachastic gradient descent step according to (4.12), as shown in Figure 4.1. Furthermore, when performing RANSAC in order to optimise the prediction function (4.1) we will also be exploring and scoring other transformations, which gives us a mechanism for identifying any margin violations which have occurred, the largest of which will contribute to the gradient descent step (4.12). In this way, our online learning approach can re-use the intermediate results of estimating T_t and thus adds only a small amount of overhead compared to detection alone.

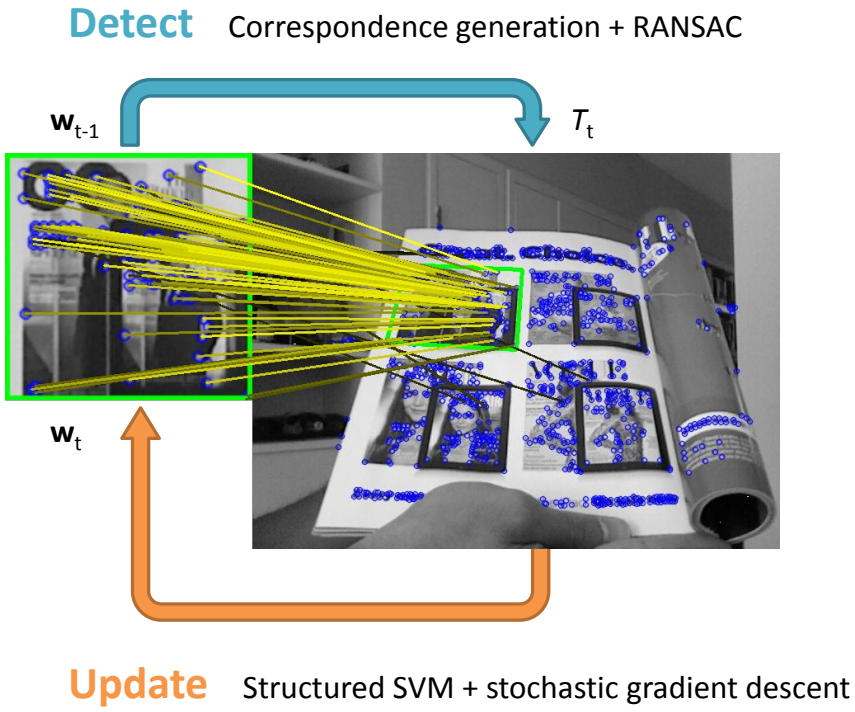


Figure 4.1: Adaptive tracking-by-detection loop. At time t, the model $\mathbf{w}_{t-1}$ from the previous frame is used in order to estimate the transformation T_t, which is subsequently used as a training example to give an updated model $\mathbf{w}_t$.

4.3.5 Binary approximation of model

An important goal of our method is to be real-time and suitable for low-powered devices, and we would therefore like to take advantage of binary descriptors. Although these descriptors are very compact when represented as bitsets, to use a linear SVM requires converting them into high-dimensional real vectors. While

this is acceptable when updating the learner, it would be very computationally expensive at the matching stage, which requires exhaustive evaluation of every model classifier with every image keypoint. To avoid this, we propose approximating each $\mathbf{w}_j$ in terms of a set of basis vectors

$$\mathbf{w}_j \approx \sum_{i=1}^{N_b} \beta_i \mathbf{b}_i \tag{4.15}$$

where $\mathbf{b}_i \in \{-1, 1\}^D$, and D is the dimensionality of the descriptor. This approximation must be updated each time $\mathbf{w}_j$ changes, so we choose to use a simple greedy method as described in Algorithm 4.1.

Require: $\mathbf{w}_j$, N_b
  $\mathbf{r} = \mathbf{w}_j$ *(initialise residual)*
  for $i = 1$ to N_b **do**
    $\mathbf{b}_i = \text{sign}(\mathbf{r})$
    $\beta_i = \langle \mathbf{b}_i, \mathbf{r} \rangle / \|\mathbf{b}_i\|^2$ *(project* $\mathbf{r}$ *onto* $\mathbf{b}_i$*)*
    $\mathbf{r} \leftarrow \mathbf{r} - \beta_i \mathbf{b}_i$ *(update residual)*
  end for
  return $\{\beta_i\}_{i=1}^{N_b}$, $\{\mathbf{b}_i\}_{i=1}^{N_b}$

Algorithm 4.1: Binary approximation of $\mathbf{w}_j$.

Using this approximation, we can efficiently compute the scalar product $\langle \mathbf{w}_j, \mathbf{d} \rangle$ using only bitwise operations. To do so, we represent each $\mathbf{b}_i$ using a binary vector and its complement: $\mathbf{b}_i = \mathbf{b}_i^+ - \overline{\mathbf{b}_i^+}$, where $\mathbf{b}_i^+ \in \{0, 1\}^D$. We then rewrite

$$\langle \mathbf{w}_j, \mathbf{d} \rangle \approx \sum_{i=1}^{N_b} \beta_i (\langle \mathbf{b}_i^+, \mathbf{d} \rangle - \langle \overline{\mathbf{b}_i^+}, \mathbf{d} \rangle), \tag{4.16}$$

and note that each scalar product inside the summation can be computed very efficiently using a bitwise AND followed by a bit-count. This can be computed even more efficiently if we have precomputed the bit-count of $\mathbf{d}$, since $\langle \mathbf{b}_i^+, \mathbf{d} \rangle - \langle \overline{\mathbf{b}_i^+}, \mathbf{d} \rangle = 2\langle \mathbf{b}_i^+, \mathbf{d} \rangle - |\mathbf{d}|$. This means that by approximating $\mathbf{w}_j$ with N_b components, our correspondence score is roughly N_b times more expensive to evaluate than a binary Hamming distance. In practice, we find it sufficient to set $N_b = 2$, see Section 4.4.3 for experimental results.

4.4 Experiments

We performed a number of experiments in order to validate the approach described in this chapter. Our method is applicable to general object models and transformations, but for the purposes of our experiments we consider the case of a planar object model detected in an image under a homography transformation.

We recorded a number of video sequences of a static scene observed from a moving camera, using a SLAM system to track the 3D camera pose in each frame (example frames can be seen in Figure 4.2). Each sequence begins with a fronto-parallel view of a planar patch, which is used in our experiments to define the object model. Using the known camera pose, we computed a ground-truth homography for the object in each video frame, which is then used for evaluating the quality of the homography estimates produced during object detection in our experiments.

Our experiments all consider the task of tracking-by-detection, as described in Section 4.3.4, in which the target object should be detected in consecutive frames of a video sequence. For this task we do not use any information about the location of the object in the previous frame when detecting the object, but we use each successful detection in order to perform an online learning step to update our object model for subsequent frames. For each sequence, we initialise a model using the fronto-parallel planar patch in the first frame, by detecting the 100 strongest features to define the locations of model keypoints $\mathcal{M}$. The weight vector $\mathbf{w}_j$ for each model keypoint is initialised by setting it to the descriptor extracted for each model keypoint in the first frame.

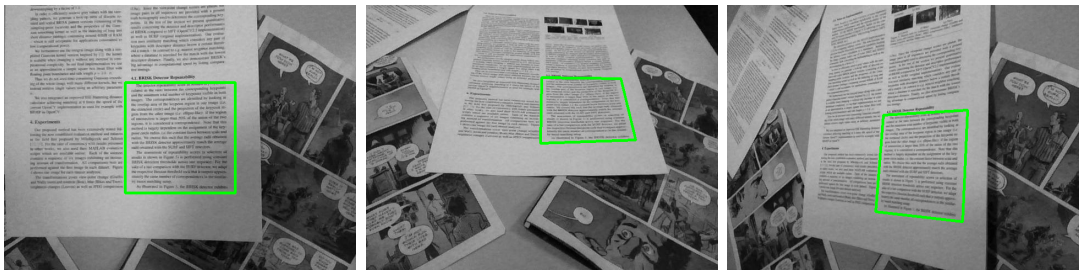
When learning with binary descriptors, we apply the feature transformation $\tilde{\mathbf{d}} = (\mathbf{d} - \mathbf{0.5})/0.5\sqrt{D}$, where D is the dimensionality of the descriptor, which centres and normalises the descriptors, as this is known to improve the performance of stochastic gradient descent algorithms [68]. During matching this transformation can easily be handled implicitly in the binary approximation without any overhead. We fix the SVM learning rate $\lambda = 0.1$ for all experiments. We also set $\nu = 1$ for the structured model.

In our experiments, we measure detection accuracy using the homography dis-

(a) *barbapapa*

(b) *comic*

(c) *map*



(d) *paper*

(e) *phone*

Figure 4.2: Example frames from our test sequences, which also show the ground-truth homography. These sequences are challenging for keypoint-based detection approaches due to the presence of many similar features in the scene.

tance $d_{homography}(T, T')$ (4.7) introduced in Section 4.3.3. Using this distance, we are able to quantitatively assess how the predicted object homography compares with the ground-truth homography in each frame of our test sequences.

A C++ implementation of our approach as well as the annotated videos used during our experiments are publicly available to download[1].

4.4.1 Loss functions

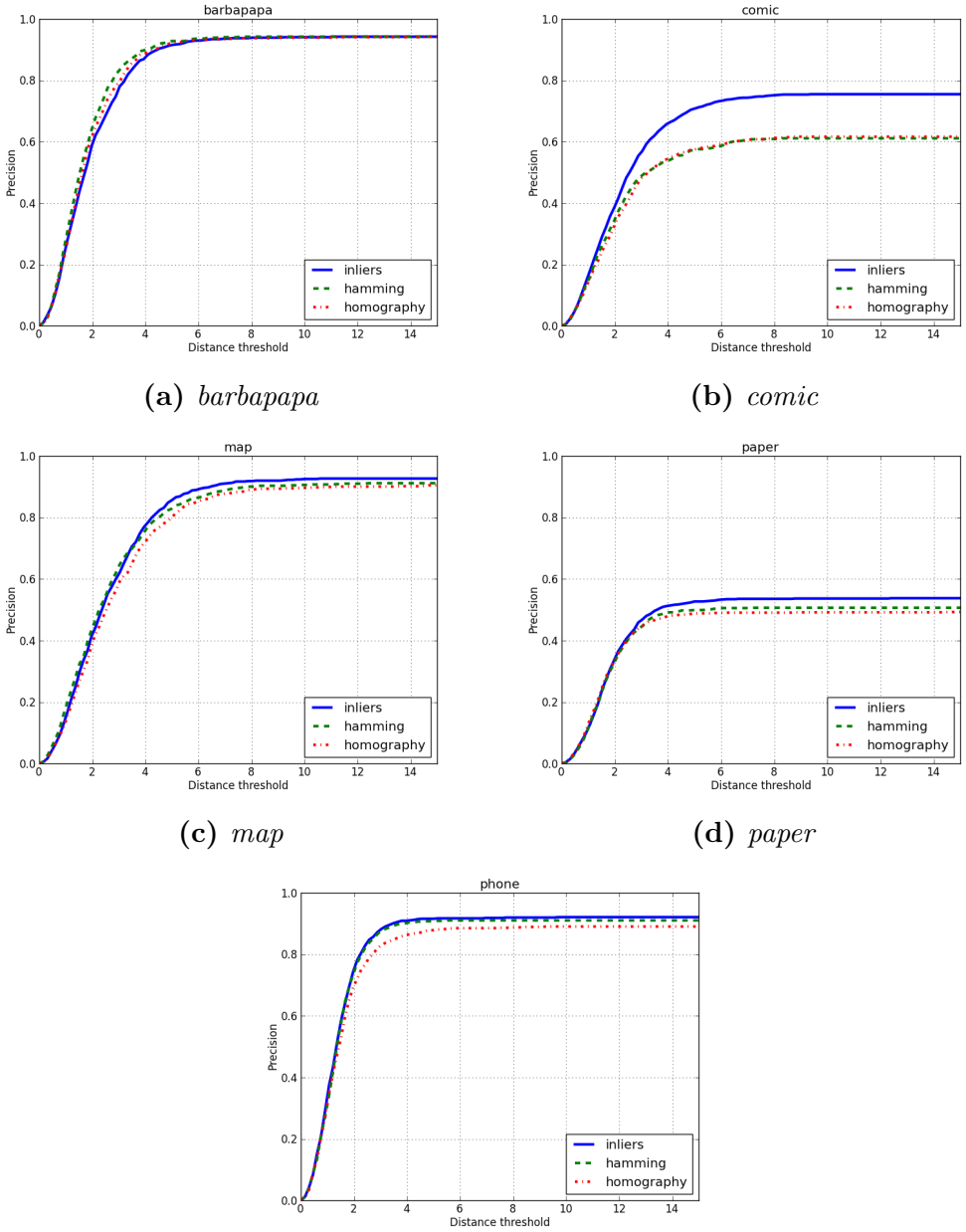
Our first set of experiments aim to investigate which of the loss functions proposed in Section 4.3.3 results in the best tracking-by-detection performance in our framework. For these experiment we use the BRISK detector with 512-bit BRISK descriptor, without using our binary approximation method.

Figure 4.3 shows precision plots obtained for each of our test sequences when using the three loss functions described in Section 4.3.3. These plots show the percentage of frames for which the homography distance $d_{homography}(T, T')$ between the detected homography and ground-truth homography is less than a particular threshold. Frames in which no detection is found are considered to have infinite distance, which is why these plots do not reach a precision of 1. From these plots we can see that overall the performance with all three loss functions is quite similar, but that the $\Delta_{inliers}$ loss function is able to consistently produce the highest detection precision on our test sequences. On the *comic* sequence, in particular, this loss function results in significantly improved performance. Another advantage of this loss function is that, unlike $\Delta_{homography}$, it is valid for all classes of transformations, since it is computed in terms of correspondences only. Therefore this can be considered a general-purpose loss function for our approach.

4.4.2 Effect of structured learning

Our next set of experiments investigate the applicability of our approach to various descriptor types, and explores the contribution of our structured learning framework, compared with independent classification for keypoint matching.

[1] `http://www.samhare.net/research`.



(a) *barbapapa* **(b)** *comic*

(c) *map* **(d)** *paper*

(e) *phone*

Figure 4.3: Precision plots comparing loss functions. These plots show the percentage of frames for which the homography distance $d_{homography}(T, T')$ defined in Section 4.3.3 between the detected homography and ground-truth homography is less than a particular threshold.

Based on the results of the previous section, for all these experiments we use the loss function $\Delta_{inliers}$.

To provide a baseline with which to compare our method, we implemented a modification of our framework consisting of independent online SVM classifiers for each model keypoint. This modification takes away the coupling between model points that comes from our model and trains each SVM classifier independently of one another. At run-time, this approach computes a matching score for the j-th model keypoint using the learned SVM classifier as $f_j(\mathbf{d}_k) = \langle \mathbf{w}_j, \mathbf{d}_k \rangle$ and uses this score to find the highest scoring match to construct the correspondence set for pose estimation. To update each classifier, each inlier returned from the geometric verification set is taken as a positive training example, and the next highest scoring match for the model keypoint is taken as a negative example. We then perform a stochastic gradient descent step to update the classifier.

We apply our approach using three different combinations of interest point detector and descriptor: FAST detector with 256-bit BRIEF descriptor, BRISK detector with 512-bit BRISK descriptor and SURF detector with SURF64 descriptor. These have been chosen to illustrate that our method works with a variety of feature point detectors and descriptors, but as they each have different invariances and dimensionality, our results should not be interpreted as a comparison between different descriptor types. Therefore, we are interested in relative performance figures for a particular feature point detector and descriptor combination.

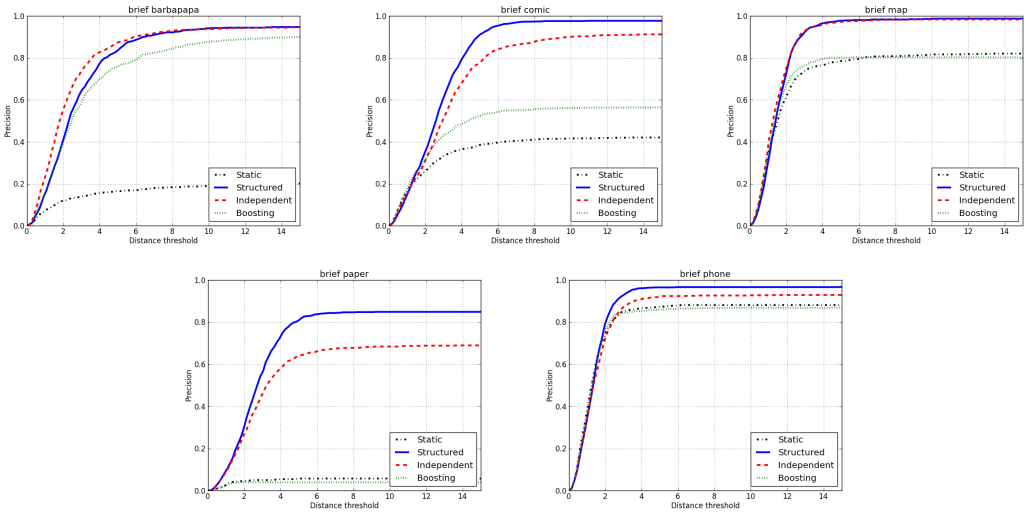
To provide an additional baseline, we implemented the boosting-based classification approach proposed by Grabner *et al.* [48], by making use of the publicly available online boosting code provided by the authors[2]. We train these classifiers in the same manner as our independent SVM baseline.

Figure 4.4 shows precision plots for each combination of keypoint detector and descriptor on our test sequences[3]. To summarise these plots, Table 4.1 shows the precision at a threshold of $d_{homography}(T, T') < 10$, which we consider to be correct detections.

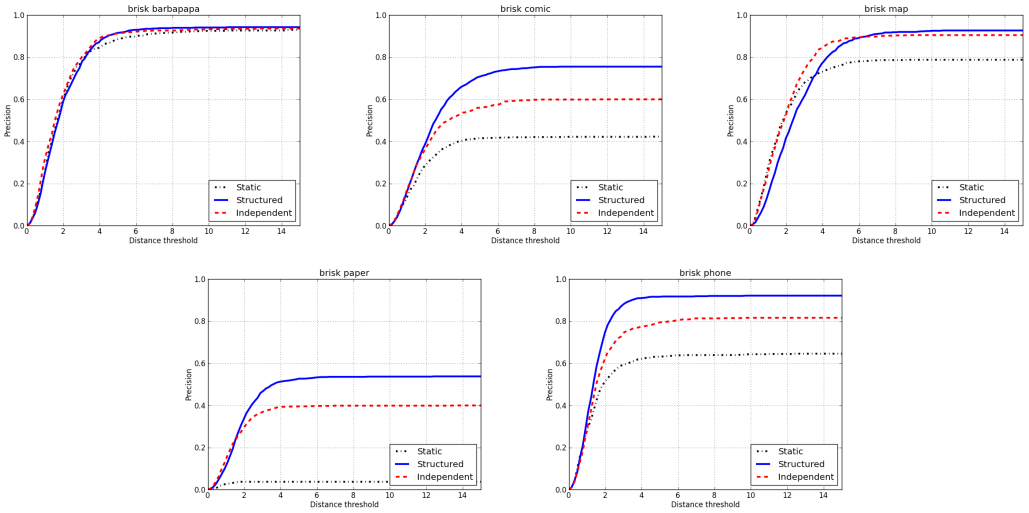
As can be seen from these results, the structured learning framework out-

[2] `http://www.vision.ee.ethz.ch/boostingTrackers/onlineBoosting.htm`.

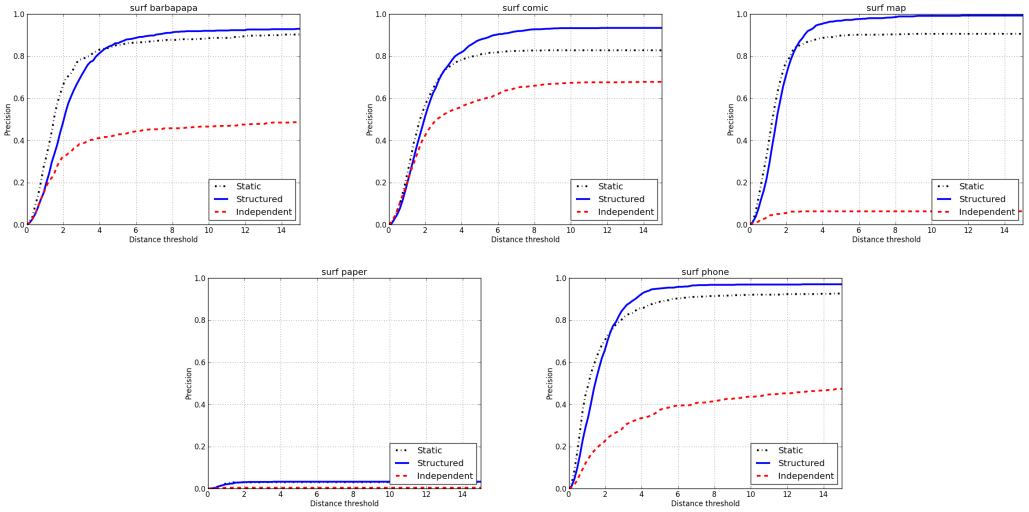
[3] Videos of these results can be found at `http://www.samhare.net/research`.



(a) FAST detector with 256-bit BRIEF descriptor



(b) BRISK detector with 512-bit BRISK descriptor



(c) SURF detector with SURF64 descriptor

Figure 4.4: Precision plots for different detector/descriptor combinations. For each combination we plot the results without learning (static), independently trained SVM classifiers, and our structured learning framework. Additionally, in (a) we plot the results of the boosting approach [48].

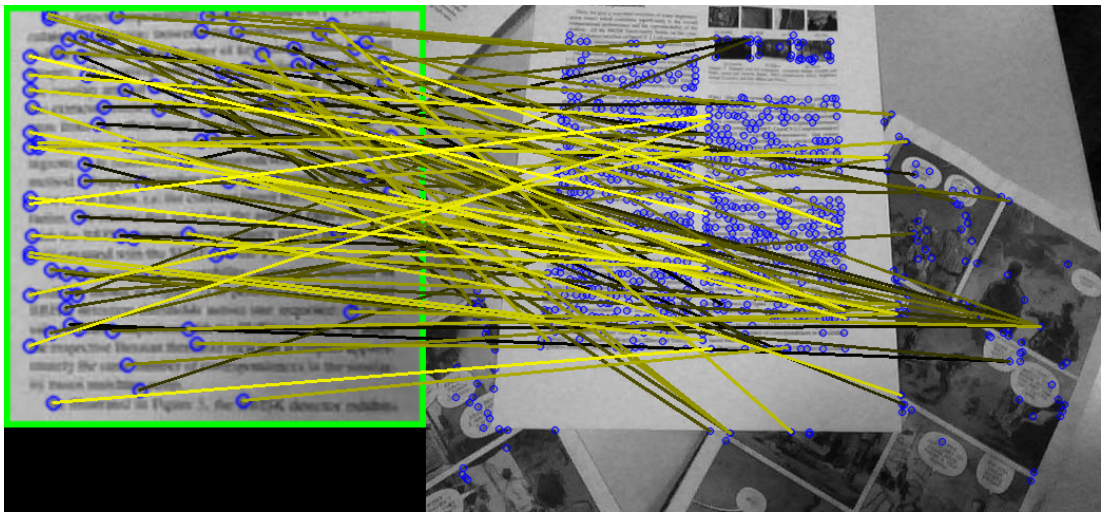
Sequence	BRIEF			BRISK			SURF			Boost. [48]
	Static	Indep.	Struct.	Static	Indep.	Struct.	Static	Indep.	Struct.	
barbapapa	0.19	**0.94**	**0.94**	0.93	0.93	**0.94**	0.89	0.47	**0.92**	0.88
comic	0.42	0.90	**0.98**	0.42	0.60	**0.76**	0.83	0.67	**0.93**	0.56
map	0.82	0.98	**0.99**	0.79	0.91	**0.93**	0.91	0.06	**0.99**	0.80
paper	0.06	0.68	**0.85**	0.04	0.40	**0.54**	**0.03**	0.01	**0.03**	0.04
phone	0.88	0.93	**0.97**	0.64	0.82	**0.92**	0.92	0.44	**0.97**	0.87

Table 4.1: Average detection rates for test sequences (the higher better). Each row represents a video sequence. Each set of columns shows a different combination of feature point detector and descriptor, while the last single column shows the results for the boosting approach. Within a feature detector/descriptor combination, we compare the results without learning (static), independently trained SVM classifiers, and our structured learning framework. The bold-face font highlights the best-performing method for a video sequence for a given detector/descriptor combination.

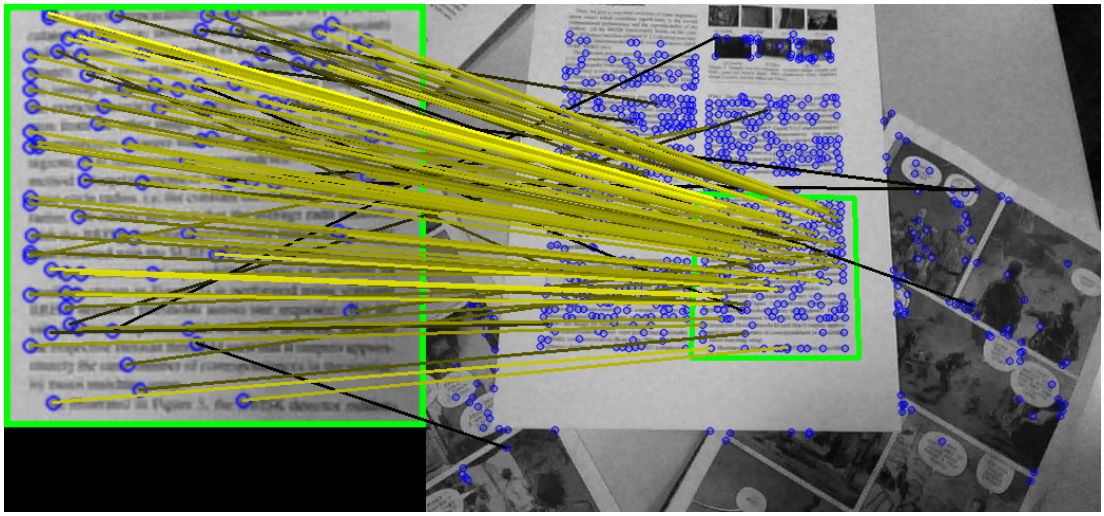
performs the static model (with no learning), as well as the model trained with independent SVM classifiers. Comparing the results of independent SVM classifiers and the static model highlights the fact that adapting an object model to a particular environment online helps a lot in practice. However, the highest detection rate is attained when we used our structured learning framework, in which the learning of the object model and geometric estimation are linked inside a unified formulation. It should be noted that for SURF descriptors the independent SVMs had difficulty learning an object model. We suspect that this is caused because of the continuous nature of the SURF descriptor and the fact that the number of generated keypoints is lower with the SURF keypoint detector. However, given the same settings, the structured learning approach is able to benefit fully from the adaptation process and improve upon the static model.

For the boosting-based learning approach, it is only fair to compare results against the models where we use the BRIEF descriptor (as both of these methods use the same FAST keypoint detector). Again, one can see by comparing the boosting method with the static method that learning provides an improvement. However, the boosting-based approach is not able to outperform the independent SVM baseline and therefore also performs worse than our structured learning framework.

The most difficult video in our set of experiments is the *paper* sequence. This video sequence features highly repetitive local appearance structures and a simple static model fails in all cases. The learning-based approaches (except the boosting method), however, are able to deliver a reasonable detection rate using binary descriptors. An example frame from this sequence is shown in Figure 4.5, where we also display the correspondences which have been found before geometric verification. As can be seen in the top image, because of the confusing appearance of the local image features, the static BRIEF model fails to match model keypoints reliably to the image. However, the structured learning framework which uses the same set of descriptors extracted from the input image has learned a more discriminative object model and is able to provide more correct correspondences, resulting in a successful detection. Another observation is that although the structured learning model produces some incorrect correspondences, they all have



(a) Static BRIEF model



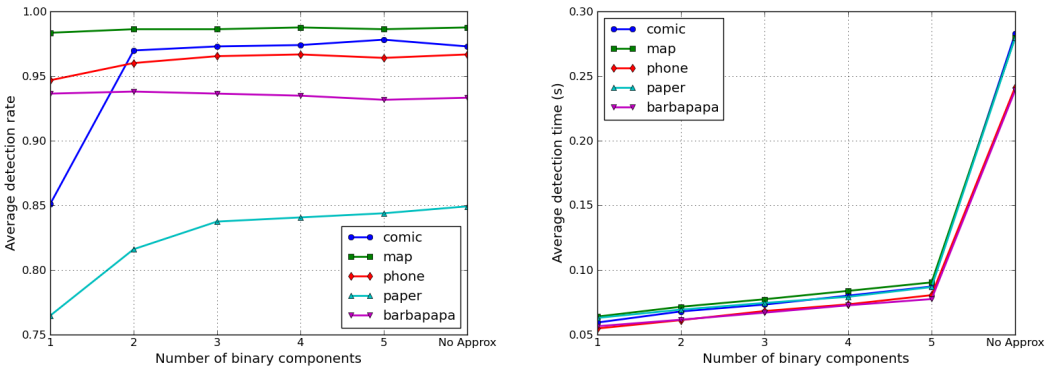
(b) Learned BRIEF model using our structured learning formulation

Figure 4.5: Example frame from the *paper* sequence showing the top correspondence for each model keypoint. The model is displayed in a green box on the left of these images. The brightness of each line indicates the correspondence score, before any geometric verification has taken place (the brighter the higher the score). The learned model has adapted to discriminate against the many confusing keypoints in the image, resulting in a successful detection, while no detection is found with the static model.

very low scores (as shown by their dark colour).

4.4.3 Binary approximation

To verify that the binary approximation proposed in Section 4.3.5 is reasonable when using binary descriptors such as BRIEF and BRISK, we repeat our experiments for the BRIEF descriptor model learned in our structured framework and approximate the model keypoint weight vectors $\mathbf{w}_j$ with varying numbers of binary bases N_b. As can be seen in Figure 4.6, in general the binary approximation produces detection performance comparable to the original results with $N_b \geq 2$ bases, and for the less challenging sequences even a single basis suffices. In terms of detection time, which includes the stages of generating correspondences between model and image, performing geometric verification, and updating the learner, we see that the binary approximation provides significant performance gains (using $N_b = 2$ we observe approximately 4 times faster detection with our implementation).



(a) Detection rate **(b)** Detection time

Figure 4.6: Behaviour of the learned BRIEF model using our structured formulation when employing a binary approximation of each $\mathbf{w}_j$ as described in Section 4.3.5. Considering $N_b = 2$, we see from (a) that detection performance is almost equivalent to the original model without approximation, whilst (b) shows that this results in approximately 4 times faster detection time.

4.4.4 Low-powered implementation

To demonstrate that our approach is indeed suitable of use on a low-powered device, we have ported our implementation to run on an Apple iPhone 4 (see Figure 4.7). On this device we observe a frame-rate of around 5fps for our approach using the proposed binary approximation with $N_b = 2$, compared with around 8fps for the static approach without learning. Note that with more device-specific optimisation both of these frame-rates could be improved, but we can already observe that our method does not add a significant overhead to the detection pipeline and is therefore suitable for real-time applications on low-powered devices.

Figure 4.7: Our method is able to perform real-time detection and learning on low-powered devices. Here it is shown running on an Apple iPhone 4.

4.5 Summary

In this chapter, we have presented a novel approach to learning for real-time keypoint-based object detection and tracking. Our formulation generalises previous methods by combining the feature matching, learning, and object pose estimation into a single structured learning framework. We showed how our framework allows an object model to be learned online, and presented an approximation to create an efficient way of using binary descriptors at runtime. During our experi-

ments we observed that structured learning plays an important role in improving the detection rate compared to state-of-the-art static and learning-based feature matching techniques.

While we did not perform feature selection explicitly, our formulation implicitly is able to down-weight the less discriminative model features and therefore provides a good starting platform for further research into automatic online feature selection. A limitation of the method as presented, however, is that all of the model keypoints must be defined at the start of learning. For applications like SLAM it would be important to perform feature selection which could also incorporate new keypoints over time, which would require extending our framework.

Chapter 5

Planar Scene Reconstruction for Portable SLAM

5.1 Introduction

The work presented in this chapter continues with the theme of augmented reality on low-powered devices from Chapter 4. We now turn our attention to the task of scene reconstruction for mobile AR gaming based upon simultaneous localisation and mapping (SLAM). Tackling this particular problem is motivated by the industrial collaboration with Sony Computer Entertainment Europe, as this is an area which has been identified as being of particular interest in the context of making vision-based computer games. The work in this chapter can therefore be seen in a slightly different light to those preceding it, since it is concerned with attempting to produce a practical solution to a specific real-world problem given certain constraints. Nevertheless, the approach which we present here is closely linked to the work in previous chapters as we treat the task as one of structured prediction and show that the addition of online learning into the resulting framework can help to improve the quality of the resulting reconstruction algorithm.

Most current AR applications make use of a known target object which can be detected and tracked in 3D using the keypoint-based approaches discussed previously in this thesis. As well as being used as a tracking target, the physical object typically then provides a 'stage' upon which virtual content can be displayed such that it appears realistically in the scene. Such an approach means that the AR experience requires the user to have this physical object in front of them, for example an image in a magazine or on product packaging.

Recently, a great deal of progress has been made in the field of vision-based SLAM, and there are now a number of robust approaches [34, 63] which can be employed to reliably track the 3D pose of a camera in real-time as it moves in a previously unknown physical environment. A lot of subsequent engineering effort has also gone in to allowing these approaches to run on low-powered devices such as smartphones and portable games consoles.

SLAM has the potential to provide a powerful platform for AR gaming, as it is able to map large physical areas and, importantly, does away with the requirement

that the user has a known object in front of them. This gives much greater flexibility in terms of when and where an AR experience can take place. However, it has the associated drawback that there is no longer a known stage on which to place virtual content. The goal of the work in this chapter is thus to develop a system able to provide a reconstruction of the underlying scene as it is explored, such that virtual content can be displayed in a realistic manner.

The majority of approaches to SLAM are based around sparse representations of the scene. The map which is built by these systems consists of a set of distinctive 3D keypoints which can be reliably tracked, which are then used in order to estimate the 3D pose of the camera in each frame. While this sparse representation is sufficient for the task of camera tracking, and has the benefit of being computationally efficient, it does not generally provide enough information for the higher-level task of displaying virtual content in the scene. For this purpose, a more complete reconstruction of the scene is required.

In this chapter, we develop an approach which uses a sparse SLAM system running on a low-powered portable games console as its starting point and aims to produce a simple reconstruction of the scene. Our target application is tabletop AR, in which we envisage the user having a playing surface along with some other objects such as boxes or books. Guided by this application area and the constraints we have in terms of computational power, we propose modelling the scene using a small number of planes, the boundaries of which we then attempt to estimate using cues from the input image stream. Besides the computational benefits, modelling the scene in this way has additional advantages for gaming applications, as the resulting reconstruction is more semantically meaningful than *e.g.* a mesh, since each planar region defines a distinct area on which game content can be displayed.

In common with the work presented in other chapters of this thesis, the approach we develop is framed as a task of structured prediction. We formulate scene reconstruction as a pixel-wise labelling problem and use a CRF to impose structure on the solution. We use relatively simple multi-view photo-consistency information in order to keep computational requirements low, but show how we can also incorporate online learning based on the appearance of each plane in

order to refine the initial solution. In this way, our approach results in an efficient reconstruction algorithm which we demonstrate to be suitable even for low-powered devices.

5.2 Motivation and related work

Multi-view reconstruction has a rich history in computer vision and many sophisticated approaches have been proposed. Algorithms are typically provided with a set of calibrated images of a scene from multiple viewpoints and then proceed to infer 3D information about the scene using multi-view stereo [105]. The calibration information can either come from a carefully-controlled capture environment in which the 3D pose of the camera is known in advance, or by using structure-from-motion techniques [53] to recover the calibration information from the images themselves. These approaches are typically designed to operate offline, with the goal of producing highly accurate reconstructions, without particular concern for computational constraints.

A SLAM system is itself performing real-time structure-from-motion and is therefore able to provide a set of calibrated images suitable for multi-view reconstruction. Because of this, there has recently been research interest in adapting multi-view reconstruction algorithms to a real-time SLAM setting. These methods are based on traditional reconstruction techniques, but take advantage of the fact that some of these algorithms lend themselves well to parallelisation. This means that these approaches can be implemented using general-purpose graphics processing unit (GPGPU) programming and make use of the extremely powerful graphics hardware in modern computers. In this way, the approaches developed by Newcombe and Davison [82, 83] and Stuehmer *et al.* [113] are both able to produce highly-detailed dense reconstructions of a scene as it is explored by a handheld camera.

The goal for our own work is also to produce a scene reconstruction in real-time using the result of a SLAM system, but we are specifically interested in doing so on low-powered portable gaming devices. In this setting, even performing SLAM using a sparse representation presents a significant computational challenge and

requires careful engineering. Furthermore, GPGPU programming is not an option, since the limited graphics hardware available is entirely used for displaying game content, meaning any solution must be suitable for a low-power CPU.

Given these constraints, the approach we take is to simplify the reconstruction task by modelling the scene using a small number of planes. While this assumption will not be suitable for all scenes, for the application of tabletop gaming which we are targeting we expect it to be able to capture the coarse structure of the scene adequately.

Similar piecewise-planar modelling approaches have been previously used for multi-view reconstruction, particularly in the case of urban street scene reconstruction, where the goal is to reconstruct building facades and roads [43, 111]. The motivation for using a planar assumption in these cases is that although it provides a simplified reconstruction of the scene, the complexity of the resulting model is constrained, meaning it provides a form of regularisation of the solution and can better handle challenges such as poorly textured or specular surfaces. These approaches are not designed for real-time operation on low-power devices, however, and still require significant computational resources.

The contribution of this chapter is a structured prediction framework for performing fully-automatic coarse scene reconstruction on a low-powered device in a few seconds, meaning it can be used in conjunction with a SLAM system running on this device to provide a platform for AR gaming. We also demonstrate that by introducing online learning of appearance information into our framework, we are able to refine the initial reconstruction solution obtained from photo-consistency information alone. In this way, our approach is able to adapt to a given scene and better handle situations in which photo-consistency is not informative or reliable.

5.3 Our approach

5.3.1 SLAM system

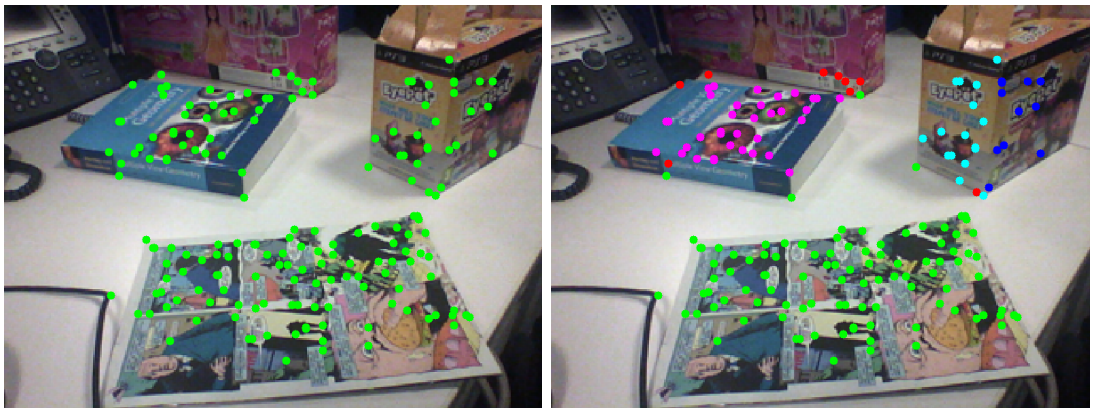
The starting point for our method is the *Magnet* SLAM system developed by Sony Computer Entertainment Europe, which has been designed to run on the

PlayStation Vita portable games console. This system is based on the PTAM method proposed by Klein and Murray [63], but has been carefully engineered to allow it to run on a low-powered device. As the camera explores a scene, the system constructs and maintains a *map* consisting of a set of 3D *landmarks* $\mathcal{L} = \{\mathbf{p}_1, \ldots, \mathbf{p}_{N_L}\}$, along with a set of *keyframes* $\mathcal{K} = \{K_1, \ldots, K_{N_K}\}$, where each keyframe is a tuple $K_i = (I_i, \mathcal{M}_i, T_i)$ consisting of a 320×240 pixel RGB image I_i, a set of 2D image-space *measurements* of a subset of landmarks (*i.e.* those landmarks which have been successfully tracked in I_i) $\mathcal{M}_i$, and the estimated 3D pose of the camera T_i. Over time, new keyframes are added to the map and a background thread periodically performs bundle-adjustment [53] in order to jointly refine the estimates of the landmark positions and keyframe camera poses. For performance reasons on a low-powered device, both the number of landmarks and the number of keyframes are kept relatively low, meaning the maps built by this system are particularly sparse. For a typical tabletop scene, we can expect something of the order of $N_K = 50$ and $N_L = 200$. Figure 5.1a shows an example of a typical tabletop scene, along with the landmarks which have been inserted into the map.

One factor which is difficult to handle in a SLAM system is scene scale, since this can not be directly estimated using visual information alone. While it is possible to estimate true scene scale given additional sensor information from accelerometers and gyroscopes, this functionality is not present in the *Magnet* SLAM system. Instead, the scene scale is arbitrary, with the initial landmarks inserted into the map at an average distance of 15 units from the first keyframe. The fact that scale is unknown does not cause problems in practice, particularly since for the tabletop scenes we are interested in, the true scale of the scene stays roughly constant.

5.3.2 Plane finding

The sparse landmarks tracked by the SLAM system correspond to locally planar surface patches in the scene. The first stage of our approach aims to automatically identify larger planes which are supported by clusters of these landmarks. Our



(a) Landmarks **(b)** Plane assignments

Figure 5.1: A typical tabletop scene. The left image shows the landmarks which have been inserted into the SLAM map, while the right image shows how these landmarks are automatically assigned to planes using the approach described in Section 5.3.2. Here 4 planes have been identified, each of which has a different colour, while red corresponds to the background class.

assumption is that each landmark can either be assigned to one of these larger planes, or otherwise can be labelled as part of the 'background' of the scene, meaning it does not belong to any plane. To achieve this, we make use of the energy-based model-fitting approach PEARL [55]. In essence, this approach offers a means for performing RANSAC [41] for fitting multiple models. However, by fitting these models simultaneously, rather than greedily fitting them individually, it has been shown to produce superior results [55]. Crucially, the method also offers a means for automatically estimating the appropriate number of models to fit, which is essential for our application.

We begin by generating an initial set of plane hypotheses $\mathcal{H}$, where each hypothesis is described by a parameter vector θ_h, consisting of a 3D normal vector and the distance from the origin. To generate $\mathcal{H}$ we perform a 2D Delaunay triangulation of the landmarks when projected into a single keyframe (the choice of keyframe is arbitrary, in practice we use the reference keyframe discussed in the following section). We use each resulting triangle to define a plane hypothesis, by computing the plane passing through all three landmarks in 3D. We also include an additional 'background' hypothesis $\emptyset$.

PEARL defines an energy function in terms of a set of labelling variables $f = \{f_p\}$, which specifies an index into $\mathcal{H}$ for each landmark $\mathbf{p}$, along with the

set of plane parameters $\theta = \{\theta_h\}$

$$E(f, \theta) = \sum_{\mathbf{p} \in \mathcal{L}} D_p(f_p, \theta_{f_p}) + \sum_{\mathbf{p},\mathbf{q} \in \mathcal{N}} V_{pq}(f_p, f_q) + \sum_{h \in \mathcal{H}} c_h \delta_h(f). \tag{5.1}$$

The first term in this energy is a data cost, which specifies the cost for assigning each landmark to a given plane. For this we use

$$D_p(f_p, \theta_{f_p}) = \begin{cases} \|\mathbf{p} - \theta_{f_p}\| & \text{if } f_p \neq \emptyset \\ d_\emptyset & \text{otherwise} \end{cases} \tag{5.2}$$

where $\|\mathbf{p} - \theta_{f_p}\|$ is the perpendicular distance between the landmark $\mathbf{p}$ and the plane with parameters θ_{f_p}. For the background hypothesis $\emptyset$, a constant cost $d_\emptyset$ is used. In all our experiments we fix $d_\emptyset = 0.3$.

The second term in (5.1) is a smoothness cost, which encourages neighbouring landmarks defined by a neighbourhood $\mathcal{N}$ to take the same label. In our case, we define a neighbourhood using the edges of the Delaunay triangulation which we originally computed for generating $\mathcal{H}$ and then use

$$V_{pq}(f_p, f_q) = w_{pq} \mathbb{I}(f_p \neq f_q), \tag{5.3}$$

where $\mathbb{I}$ is an indicator function, and

$$w_{pq} = \beta \exp\left(-\frac{\|\mathbf{u}_p - \mathbf{u}_q\|^2}{\sigma_w^2}\right). \tag{5.4}$$

Here $\mathbf{u}_p$ is the 2D position of landmark $\mathbf{p}$ when projected into the reference image (and likewise for $\mathbf{u}_q$), meaning that w_{pq} is larger for pairs of landmarks which are closer together in the reference image. σ_w is computed based on the mean value within the reference image $\sigma_w^2 = \frac{2}{|\mathcal{N}|} \sum_{p,q \in \mathcal{N}} \|\mathbf{u}_p - \mathbf{u}_q\|^2$ [19], and in all our experiments we fix the parameter $\beta = 0.05$.

The final term in (5.1) is a label cost, which plays the important role of controlling the number of planes which are active. Here

$$\delta_h(f) = \begin{cases} 1 & \text{if } \exists p : f_p = h \\ 0 & \text{otherwise,} \end{cases} \tag{5.5}$$

meaning every plane with non-zero support will incur a cost. The parameter c_h controls how much cost is paid for each active hypothesis, and therefore by setting this parameter appropriately we can encourage solutions using a small number of planes, which is what we desire for our application. In our experiments we fix $c_h = 4$, except for the background hypothesis for which we use $c_\emptyset = 0$, as it should always be active and not penalised.

PEARL then proceeds to minimise (5.1) in an EM fashion: it first fixes plane parameters θ and optimises over the landmark labelling f, and then fixes f and optimises over θ. Since only the first term in (5.1) is affected by θ, for a given labelling f we can simply perform a least-squares fit for each active plane independently, which is guaranteed to improve the solution. The original PEARL algorithm [55] proposed a heuristic means of optimising the labelling f given a fixed θ, but this was subsequently improved by Delong *et al.* [35], who proposed an extension to the α-expansion [21] algorithm capable of handling the label cost term in (5.1), which is the method we use. These two optimisation steps constitute a single iteration of PEARL, and in practice we find it sufficient to perform 3 iterations, as the method converges quickly.

At the end of this process, we are left with a set of active planes $\mathcal{P} = \{\pi_1, \ldots, \pi_{N_P}\}$, along with an assignment of each landmark to one of these planes, or to the background class. An example assignment can be seen in Figure 5.1b, in which the landmarks have been coloured to reflect their assignments to planes.

5.3.3 Boundary estimation

The set of planes $\mathcal{P}$ gives us some information about the geometry of the scene, but because these planes have infinite extent they are of limited use in practice, since any augmentation would not respect physical boundaries of these planes in the scene. The next stage of our approach therefore aims to estimate boundary information for each of these planes. While we can already derive some information about plane extents by considering which landmarks have been assigned to each plane, this information is very sparse and is not sufficient to obtain a full reconstruction of the scene. Furthermore, by definition landmarks correspond

to highly-textured points in the scene, meaning they do not generally provide information about regions with low texture.

The approach we propose for estimating plane boundaries is to treat the task as a structured prediction problem and perform pixel-wise labelling of a reference view of the scene. Our goal is to assign each pixel in this view to one of the planes in $\mathcal{P}$, or to the background if it does not lie on any of these planes. Given such a labelling, we can then back-project the labels in the reference view onto each plane in order to obtain their extents. This approach is viewpoint-dependent and is therefore only able to produce a 2.5D reconstruction of the scene. However, this formulation considerably simplifies the resulting labelling problem, since in a particular view each pixel can only be assigned to a single plane, and we can therefore avoid explicitly handling the complex dependencies between planes such as how they occlude one another. This type of 2.5D reconstruction is also how most other approaches for real-time multi-view reconstruction proceed [82, 83, 113].

We begin by selecting a reference view of the scene, which is taken from the set of keyframes $\mathcal{K}$. Given that landmarks provide us with useful information for performing labelling, we choose the keyframe K_r containing the largest number of landmark measurements to be our reference view. In order to keep computational requirements as low as possible, we first reduce the size of the labelling problem by over-segmenting the reference image I_r to produce a set of superpixels $\mathcal{S}$. The method we choose for generating superpixels is the SLIC [1] algorithm, which is particularly computationally efficient, whilst producing regular superpixels that respect image boundaries well. This step reduces the size of the labelling problem dramatically from $320 \times 240 = 76,800$ pixels to roughly 1000 superpixels. Example superpixel segmentations can be seen in Figure 5.2.

To find a labelling L of the superpixels we define a pairwise CRF over the graph $\mathcal{G} = (\mathcal{S}, \mathcal{N})$, where $\mathcal{N}$ is the neighbourhood defined by pairs of superpixels which share a boundary. In doing so, we introduce structure into the resulting labelling problem, since we are making the assumption that the labels of neighbouring superpixels should affect one another. A good labelling then corresponds

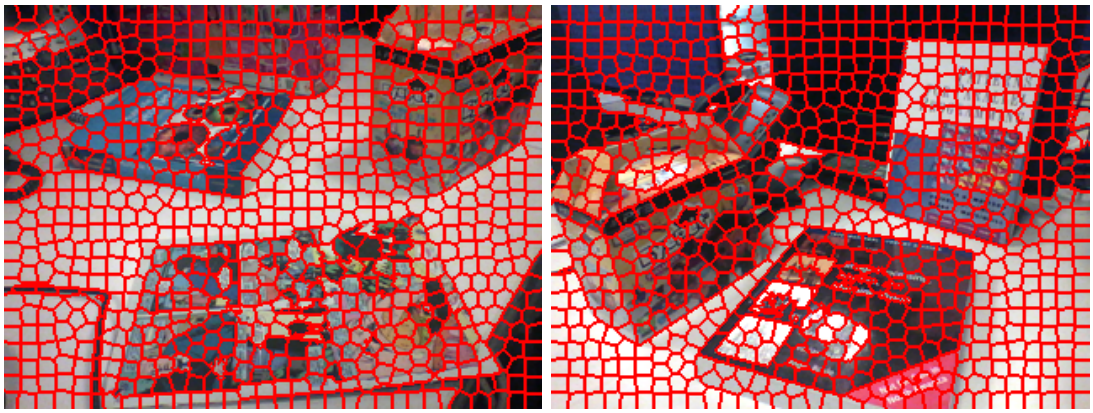


Figure 5.2: Examples of SLIC superpixels [1] for two reference images.

to the minimum of the energy function

$$E(L) = \sum_{s \in \mathcal{S}} D_s(L_s) + \sum_{s,t \in \mathcal{N}} V_{st}(L_s, L_t). \tag{5.6}$$

We define the data term D_s by considering the multi-view photo-consistency of pixels belonging to superpixel s. For each pixel $\mathbf{u}$ in I_r, given a particular plane π we can calculate a hypothesised 3D position $\mathbf{X}_\mathbf{u}^\pi$ by back-projecting the pixel onto the plane. We select a small number of other keyframes from nearby viewpoints[1] $\mathcal{K}^\rho = \{K_1^\rho, \ldots, K_{N_\rho}^\rho\}$ to use for measuring photo-consistency, where in practice we take $N_\rho = 4$. The per-pixel photo-consistency cost for the plane π is then defined as the average of the L_1-norm of the colour difference measured in Lab colour space in each keyframe[2]

$$\rho_\mathbf{u}(\pi) = \frac{1}{N_\rho} \sum_{K_o \in \mathcal{K}^\rho} \| I_r^{Lab}(\mathbf{u}) - I_o^{Lab}(\text{proj}(T_o^{-1} \mathbf{X}_\mathbf{u}^\pi)) \|_1, \tag{5.7}$$

where $\text{proj}(\cdot)$ projects a point from 3D camera space to 2D screen space. The L_1-norm is used in order to provide robustness against the situation where a pixel visible in the reference frame is occluded in another view, since we do not explicitly attempt to model these occlusions [83]. In order to add some tolerance for sensor noise and slight inaccuracy in the estimates of camera poses, we first apply a Gaussian blur with $\sigma = 1.0$ to all images before computing this cost. The

[1] In practice we choose the keyframes with the highest number of shared measurements with the reference keyframe, since this is a good indication that the viewpoints are nearby.

[2] The average is only taken over those keyframes for which $\mathbf{X}_\mathbf{u}^\pi$ projects inside the image I_o.

data term D_s for a superpixel is then defined by taking the median cost for all pixels belonging to this superpixel, which provides additional tolerance to error caused by our relatively simple photo-consistency cost

$$D_s(L_s) = \begin{cases} \underset{\mathbf{u} \in s}{\text{median}}(\rho_{\mathbf{u}}(L_s)) & \text{if } L_s \in \{\pi_1, \ldots, \pi_{N_P}\} \\ \rho_\emptyset & \text{otherwise.} \end{cases} \tag{5.8}$$

The background label $\emptyset$ presents a problem for this photo-consistency measure, since it is not possible to project a pixel into other views when it is assigned to the background, as the depth is unknown. The only option is therefore to use a constant cost in this case, which should be lower than the photo-consistency cost for typical incorrect plane assignments. This issue will be discussed more in Section 5.4, but for all our experiments we have empirically chosen the value $\rho_\emptyset = 5$. Example photo-consistency costs for the example scene in Figure 5.1 can be seen in Figure 5.3a.

The pairwise smoothness term V_{st} in (5.6) should encourage smooth labellings of the reference image and is defined in terms of a combination of colour similarity between neighbouring superpixels and 3D depth information obtained from the plane geometry, similar to the approach taken by Gallup *et al.* [43]

$$V_{st}(L_s, L_t) = \beta \; V^c(s,t) \; V^d_{st}(L_s, L_t). \tag{5.9}$$

Here β is a constant scaling factor to ensure that the data and smoothness terms are comparable, in our experiments we fix $\beta = 15$. The first term is influenced by colour similarity and is defined as

$$V^c(s,t) = 0.2 + 0.8 \exp\left(-\frac{\|\bar{I_r}^{Lab}(s) - \bar{I_r}^{Lab}(t)\|^2}{\sigma_c^2} \right), \tag{5.10}$$

where $\bar{I_r}^{Lab}(s)$ is the mean Lab colour over pixels $\mathbf{u} \in s$, and $\sigma_c^2 = \frac{2}{|\mathcal{N}|} \sum_{s,t \in \mathcal{N}} \|\bar{I_r}^{Lab}(s) - \bar{I_r}^{Lab}(t)\|^2$ [19]. The effect of this term is to encourage transitions between labels to take place at colour discontinuities in the reference image, since these often correspond to the boundaries of planes. The second term is influenced by depth

information between planes and defined as

$$V_{st}^{d}(L_s, L_t) = \begin{cases} 0 & \text{if } L_s = L_t \\ 1 & \text{if } L_s = \emptyset \text{ or } L_t = \emptyset \\ 0.3 + 0.7 \min(1, d/100) & \text{otherwise,} \end{cases} \tag{5.11}$$

where d is the 3D depth difference between the centre of superpixels s and t according to their labels. The effect of this term is to encourage transitions between labels to take place at locations in the reference image corresponding to the projections of plane intersections, which should help to ensure the labelling respects the planes which have been identified.

Finally, we make use of the labelling of landmarks provided by the initial plane-fitting stage of our approach in order to provide hard constraints to guide the superpixel labelling L. This step is important as it allows us to inject the sparse labelling information which has already been obtained for the 3D landmarks into the resulting dense reconstruction. Each 2D landmark measurement $\mathbf{u}_m \in \mathcal{M}_r$ in the reference frame corresponds to a 3D landmark $\mathbf{p}_m$ which has now been assigned to a particular plane π (or the background $\emptyset$). We therefore find the superpixel s which contains $\mathbf{u}_m$ and modify D_s such that it becomes

$$D_s(L_s) = \begin{cases} 0 & \text{if } L_s = \pi \\ \infty & \text{otherwise.} \end{cases} \tag{5.12}$$

In cases where multiple measurements fall within the same superpixel but belong to different planes, we leave D_s unchanged.

The resulting energy (5.6) defines a standard multi-label pairwise CRF, for which an approximate solution can be efficiently found using the α-expansion algorithm [21].

5.3.4 Online learning of plane appearance

Multi-view photo-consistency provides a strong cue for performing the labelling of the reference view, however there are still situations where it is not infor-

mative or reliable. This is particularly the case for textureless regions, where the photo-consistency cost (5.7) is generally unable to identify the correct plane assignment. Another issue is that the photo-consistency measure we use is computed in a rather simple manner to keep computational cost low and is therefore not very tolerant to slight inaccuracy in the keyframe poses estimated by the SLAM system, as well as different exposure settings or sensor noise in the camera images.

In order to handle these issues, we propose incorporating appearance information for each plane into the reconstruction pipeline. This is achieved by learning appearance models for each plane, which are initialised using the photo-consistency solution and subsequently used to refine the labelling in an iterative fashion. This approach is inspired by similar ideas which have been applied to interactive image segmentation, such as the GrabCut algorithm [97].

The approach we propose is to introduce a classifier which can be used to predict which plane a given superpixel belongs to, based on appearance information alone. For this purpose we use a multi-class linear SVM classifier learned in a one-vs-all manner [38], since this allows us to take advantage of efficient online SVM learning approaches [106]. For each superpixel s, we construct a feature vector $\mathbf{x}_s$ defined by the bins of a 3D colour histogram. This histogram uses 5 bins per colour channel, meaning $\mathbf{x}_s$ is a 125D vector with $\|\mathbf{x}_s\|_1 = 1$. For each plane π, as well as the background class $\emptyset$, we introduce a linear weight vector $\mathbf{w}_\pi$. Given a labelling L, we can generate positive and negative training examples $\mathcal{X}_\pi^+ = \{\mathbf{x}_s \mid L_s = \pi\}$ and $\mathcal{X}_\pi^- = \{\mathbf{x}_s \mid L_s \neq \pi\}$ for each plane, which are then used to update the associated weight vectors by performing online learning using the Pegasos algorithm [106] (which was also described in Section 2.4.2.3 of this thesis). For all our experiments, we use an SVM regularisation of $C = 0.1$.

In order to use such a classifier to produce a unary cost for the labelling energy function, we take the approach of Kumar and Hebert [66] and use a logistic function to produce a per-plane likelihood for each superpixel from the SVM classification score:

$$P(\mathbf{x}_s|L_s) = \frac{1}{1 + \exp(-\langle \mathbf{w}_{L_s}, \mathbf{x}_s \rangle)}. \tag{5.13}$$

This likelihood will be close to 1 when the classification score is large and positive and close to 0 when it is large and negative. We then take the negative log-likelihood to be the unary superpixel cost:

$$D_s^{app}(L_s) = -\log P(\mathbf{x}_s|L_s) = \log(1 + \exp(-\langle \mathbf{w}_{L_s}, \mathbf{x}_s \rangle)) \tag{5.14}$$

This appearance cost is combined with the original photo-consistency cost to give

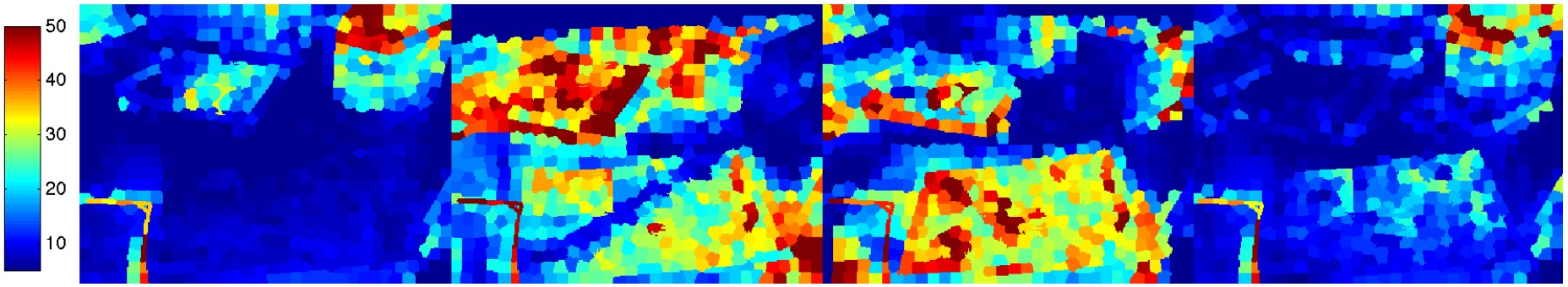
$$D_s(L_s) = \frac{1}{2}(D_s^{pc}(L_s) + \gamma D_s^{app}(L_s)) \tag{5.15}$$

where D_s^{pc} is the original cost defined in Section 5.3.3, and γ is a parameter to ensure the scales of the two costs are comparable. In all our experiments we fix $\gamma = 10$. Appearance costs produced after training the SVM classifier from the initial photo-consistency labelling for the example scene in Figure 5.1 can be seen in Figure 5.3b.

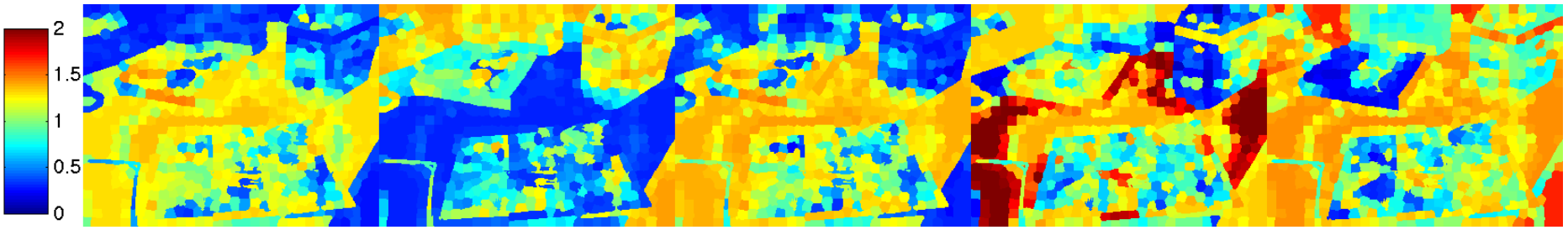
Our overall approach proceeds as follows: we first find an initial labelling L^0 using photo-consistency information alone (as described in Section 5.3.3). We then use this labelling in order to learn the weight vectors $\{\mathbf{w}_\pi^0\}$ for each plane. These weight vectors are subsequently used to define the combined data cost (5.15), resulting in a new labelling L^1. This new labelling is used to update the weight vectors per plane $\{\mathbf{w}_\pi^1\}$, and the process is repeated for a number of iterations, in a similar manner to GrabCut [97].

In common with the other approaches which have been presented in this thesis, we are therefore making use of online learning in order to provide an element of adaptability to a given environment with this approach. Our motivation is that the use of photo-consistency information provides a good starting point for a scene reconstruction and will succeed in many areas. However, there are other areas which will not be well reconstructed using photo-consistency information alone, and the hope is that plane appearance provides an orthogonal cue which will allow information to be transferred to the uncertain regions, resulting in a more consistent overall reconstruction.

Unlike the structured learning approaches presented in Chapters 3 and 4, in this approach the classifier does not explicitly take structure into account, as it



(a) Photo-consistency cost



(b) Appearance cost

Figure 5.3: Example superpixel unary costs for the tabletop scene in Figure 5.1. (a) shows the photo-consistency cost computed according to (5.8) for each of the 4 planes identified in Figure 5.1b. (b) shows the appearance cost computed according to (5.14) using the SVM classifier trained from the initial photo-consistency solution. Note that in (b) the left-most cost is for the background label, for which it is not possible to compute a photo-consistency cost.

is trained to classify superpixels independently. However, it is worth noting that the way in which this classifier is trained *does* take the structure into account. The samples used to train the classifier are generated based on the final CRF labelling, which has been found using the pairwise neighbourhood structure of the superpixels in the reference image. Thus our approach can still be seen as performing a form of structured learning, with the structure being taken into account implicitly by the learning procedure.

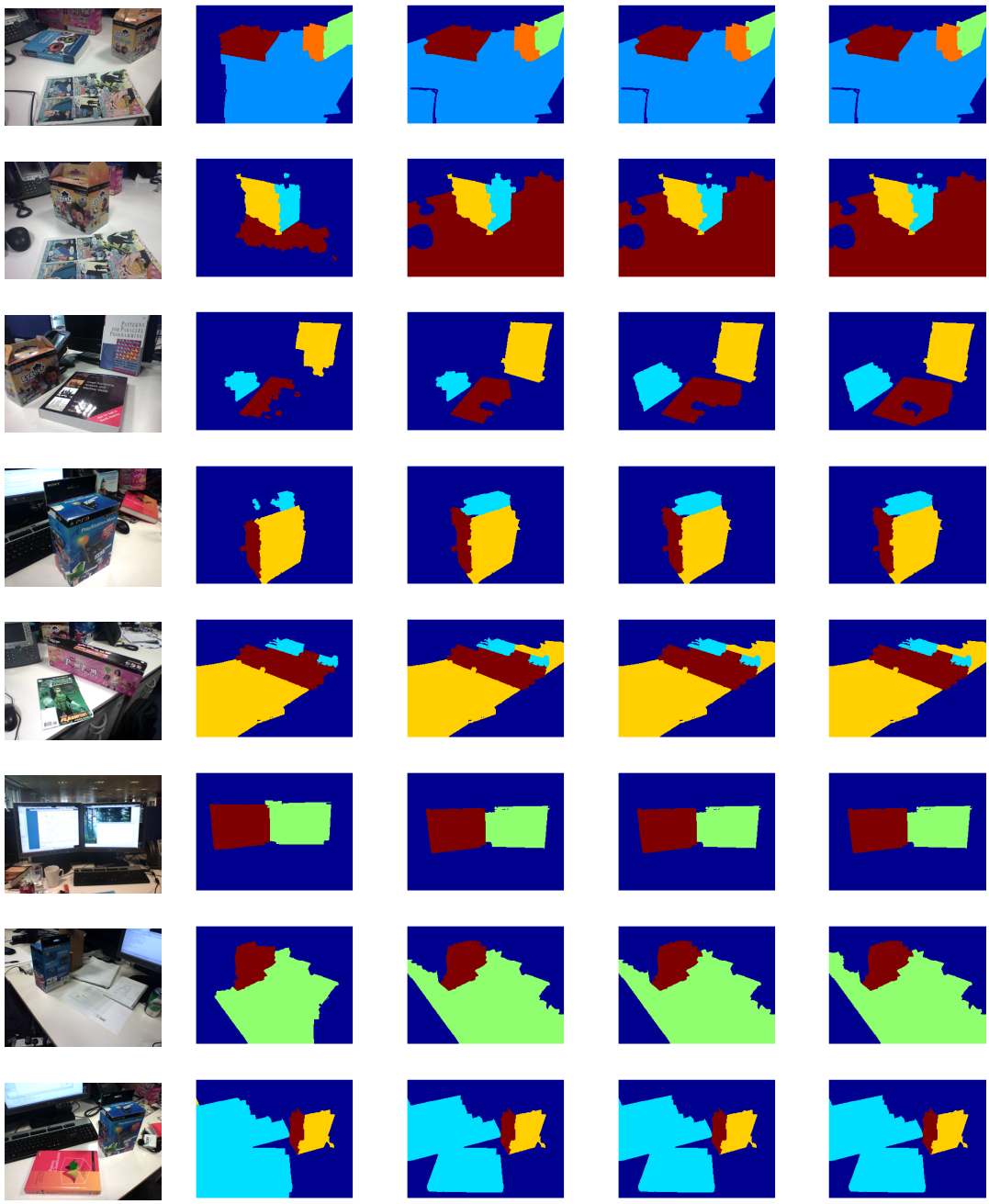
5.4 Results

Typical results produced by our approach on a number of example desktop scenes can be seen in Figure 5.4. For each scene this figure shows the reference image, along with the initial result found using photo-consistency information alone (Section 5.3.3). Subsequent columns show the result after each iteration of re-labelling using the appearance-based classifier (Section 5.3.4), which shows how the resulting labelling changes as appearance information is incorporated and updated.

As can be seen from these results, in many cases the proposed method is able to produce promising coarse scene reconstructions. For the applications we are interested in, namely providing basic scene reconstruction for gaming applications, these reconstructions would often be adequate. In most cases, the actual boundaries would not need to be displayed to the user, but rather they would be used internally by a game in order to allow virtual content to respect physical boundaries in the scene. The boundaries identified by our approach would be sufficient for defining collision geometry for a physics engine, or for performing occlusion of virtual objects as they move behind objects in the scene.

We see from these results that the plane-finding stage of our approach (Section 5.3.2) is rather robust and able to reliably find a small number of dominant planes in the scene using the landmarks from the SLAM map. We have found empirically that this stage is not very sensitive to parameter settings and also requires minimal computational cost.

The boundary estimation stage of our approach is less robust, however, and we see from the results that it does not reliably produce high-quality segmenta-



(a) (b) (c) (d) (e)

Figure 5.4: Result of the proposed method on a number of tabletop scenes. The reference image is shown in column (a), and column (b) shows the result of labelling using photo-consistency cost alone. Columns (c)-(e) show the result of labelling after each iteration of updating the appearance-based classifier. In all cases the dark blue colour corresponds to the background label.

tions. The primary issue is that the method aims to be fully automatic, and as a result the iterative online learning algorithm we use can fail to refine the solution if the initial labelling provided by the photo-consistency cost contains gross errors. Essentially, our method is performing an unsupervised clustering of the scene, and if the cluster initialisations are poor, the final labelling will also suffer. Another issue is that there are a relatively large number of parameters involved in defining the energy function which is minimised when performing labelling, and although these have all been fixed throughout our experiments, they have been set empirically by hand.

Perhaps the most serious difficulty with this approach is the requirement of having a background label. As has previously been mentioned, because pixels labelled as background have unknown depth, we must use a fixed value for the photo-consistency cost for this label. Choosing this value to work in all cases is difficult: if it is too high then regions which should be labelled as background are assigned to planes, which subsequently is reinforced when learning the appearance models for planes; conversely, if it is too low then regions which should be assigned to planes are assigned to the background, which also is reinforced once the appearance models are learned.

5.4.1 Implementation on a low-powered device

The work in this chapter was originally motivated by the desire to perform scene reconstruction on a low-powered device. To demonstrate that the approach we have developed is indeed suitable for this setting, we have produced an implementation for the PlayStation Vita portable games console.

After an initial period of exploring the scene in order to build up a map, we then trigger our reconstruction algorithm. The reconstruction algorithm only needs to be run once in order to produce 3D geometry which can subsequently be used by AR applications, so it is sufficient for it to be able to execute within a few seconds, which could then be run as a background task as part of a game. Table 5.1 shows timings for our method when running on this hardware. In this implementation, we first perform labelling using photo-consistency information

alone and then perform three iterations of appearance learning and re-labelling to produce the final reconstruction. The computational complexity of some of the stages involved are affected by the number of active planes, so we show timings for scenes in which 1-5 planes have been identified. In all cases the timings are produced by averaging the results over 3 different scenes with the given number of planes.

Stage	1 plane	2 planes	3 planes	4 planes	5 planes
Plane finding	62	62	62	62	62
Blurring	59	59	59	59	59
Photo-consistency	810	1388	1971	2557	3150
SLIC	391	391	391	391	391
Labelling	155	188	205	227	254
Learner update	5	10	14	19	24
Total	**1957**	**2687**	**3354**	**4048**	**4769**

Table 5.1: Timings (ms) of our approach running on a PlayStation Vita. We first perform labelling using photo-consistency information alone, followed by 3 iterations of appearance learning and re-labelling to produce the final reconstruction.

We see from these results that our approach is able to produce results for up to 5 planes within approximately 5 seconds. It should be noted that these timings are for a standard C++ implementation which does not include any device-specific optimisation, and we therefore would expect that all of these could be significantly improved. Nevertheless, the fact that the approach can run within a few seconds would already be acceptable for a background task during a game. The most expensive operation is currently the building of the per-pixel photo-consistency cost, since this involves warping the keyframe images with a homography defined by each of the planes which have been identified. As the number of planes increases, this stage dominates the overall time taken by the algorithm. However, we anticipate that this stage in particular could be significantly optimised for the target device, which would therefore make the overall algorithm much faster.

Figure 5.5 shows some examples of the reconstruction system running on this hardware, where the results of labelling have been back-projected to produce a 3D mesh defined by the centres of the superpixels. Although coarse, we can see that this geometry would be sufficient for the purpose of adding AR content realistically into the scene.

Figure 5.5: Examples results of the proposed reconstruction algorithm running on a PlayStation Vita portable games console. Here the labelling has been back-projected to produce a 3D mesh for each plane defined by the centres of superpixels.

5.5 Summary

In this chapter we have presented a method for performing coarse 3D reconstruction of tabletop scenes intended for AR applications based around SLAM. Our motivation was to produce an approach suitable for use on low-powered devices, and we have shown that this has been achieved with an implementation which operates within a few seconds on a PlayStation Vita portable games console.

Our approach makes use of simple photo-consistency information to obtain an initial reconstruction and subsequently uses online SVM learning of plane appearance in order to produce a more refined solution. We have demonstrated how this use of online learning allows us to transfer information from regions which are well-reconstructed using photo-consistency information to those which are not, such as textureless regions, resulting in a more complete overall reconstruction.

While the framework we have presented shows promising results, there are still a number of outstanding issues and avenues for future research which we believe would make for a more robust and practically useful solution.

The first issue is how best to handle the background class, which is required for labelling regions of the reference image which do not belong to any plane. This class requires a constant photo-consistency cost to be used and choosing this value to work across all scenes is difficult in our current framework. One approach for tackling this issue could be to make this value adaptive and attempt to estimate it online for a given scene in order to give a stable reconstruction.

Another avenue for future work would be to improve the accuracy of the labelling of the reference view. One issue at present is that our algorithm contains a relatively large number of parameters which have been set by hand, so it would most likely be beneficial to try and learn these parameters based on labelled training data, which has been shown to be beneficial for other pixel-wise labelling problems [4,114]. Accuracy could potentially also be improved by including additional features besides colour histograms when learning the per-plane appearance classifier. Texture, for example, or more sophisticated features [109, 110] could potentially provide stronger cues for classification.

Finally, the reconstruction produced by our algorithm is currently 2.5D, as we generate per-pixel depth information for the reference view. For a more complete 3D scene reconstruction, the reconstructions from multiple reference views could be fused together, in an approach similar to that used by other real-time reconstruction methods [82, 83].

Chapter 6

Conclusions

In this thesis we have tackled three real-time computer vision problems, all of which are motivated by their potential application to vision-based computer games. This motivation stems from an industrial collaboration with Sony Computer Entertainment Europe, who are interested in using computer vision to provide a platform for the development of modern and accessible computer games. The desire to tackle problems and produce techniques which have real-world applications has been an important factor throughout this thesis, and we hope that the approaches that have been presented can provide building blocks for future research and product development in this space.

A common theme throughout this thesis has been a focus on computational efficiency, since gaming applications typically demand real-time algorithms which can be run interactively as frames are received from a camera. This requirement has influenced many of the design choices which have been taken when developing solutions in this work. Furthermore, the work in Chapters 4 and 5 has focused on providing solutions which are suitable for low-powered devices, which are an increasingly important platform from a gaming perspective. While the power of these devices is increasing at a rapid pace, they still possess only a fraction of the power of a typical desktop computer, meaning designing real-time computer vision algorithms for them still presents a major challenge.

The other major theme throughout this thesis has been online structured learning, which has been incorporated into all of the solutions we have developed. We have used online learning in order to provide a principled and computationally efficient means for incorporating adaptability into our algorithms, which is essential for handling the wide variety of environments we expect to encounter when deploying vision-based games in the real world. Incorporating structure into the learning results in even greater gains, since the learner is more tightly integrated into the overall pipeline, meaning the adaptability is focused correctly for the target application.

6.1 Contributions

In Chapter 3 we considered the task of 2D arbitrary object tracking, which has many potential applications for human-computer interaction and AR. We presented a novel approach for this task which makes use of online kernelised structured output learning in order to model the appearance of the target object during tracking. Our method is able to adapt online to appearance changes of the target object and its surrounding background during tracking, and does so using a principled structured learning framework which takes the entire tracking pipeline into account, rather than artificially introducing an intermediate classification stage. The use of kernels provides great flexibility in terms of the image representation which can be used by our method, allowing different image features to be used and combined together. We also introduced a budgeting mechanism which ensures that the computational complexity of our approach remains bounded, meaning it is suitable for the real-time applications we are targeting. Experimentally, we observed that our framework results in a tracking algorithm which delivers state-of-the-art performance on standard tracking datasets.

Chapter 4 continued the theme of adaptive object tracking, this time focusing on keypoint-based object tracking, which is central to many AR applications. The approach we presented takes the traditional pipeline of keypoint matching and geometric verification, and embeds this within an online structured learning framework. In doing so, our approach is able to provide a principled mechanism for adapting the detection pipeline for a specific object and background environment. This allows our approach to provide significant improvements to detection performance compared with traditional methods and means we can handle challenges such as repetitive features and confusing background, which we demonstrated experimentally. Our approach adds only a small amount of overhead compared to a non-adaptive approach, and we further showed how we can make approximations which allow us to take advantage of recently proposed binary keypoint descriptors, allowing for real-time operation even on low-powered devices.

In Chapter 5 we tackled a different problem related to AR: scene reconstruc-

tion for SLAM on low-powered devices. In common with other work in this thesis, we framed the task as one of structured prediction and presented an approach which is able to automatically identify a small number of dominant planes in a scene, along with estimates of their boundaries. To perform this boundary estimation, our approach initially makes use of simple multi-view photo-consistency information, and subsequently incorporates online learning of the appearance of each plane to help refine the reconstruction. The resulting algorithm is computationally efficient, and we show that it is able to run on a typical scene in a few seconds on a low-powered device, making it suitable for mobile gaming applications based around SLAM.

6.2 Future work

When developing the approaches described in this thesis, there was a desire to produce principled frameworks which could be built upon by future research. The use of online learning, in particular, means there is a great deal of existing research from the machine learning and computer vision communities which could be incorporated into the approaches we have presented in this thesis.

For the 2D object tracking approach presented in Chapter 3, future work could include extending the output space which is used during tracking. One example would be to consider tracking which takes into account object deformation and articulation, while another would be to handle jointly tracking multiple target objects. Other potential avenues include exploring different types of image features, as well as incorporating online multiple kernel learning [45] to choose features which are well suited to a given object and environment. Another interesting direction would be to adapt this algorithm so that it is better-suited to low-powered devices, perhaps by using binary features such as those used for keypoint matching in Chapter 4.

For the keypoint-based object tracking approach presented in Chapter 4, our approach is already able to down-weight those keypoints which are less discriminative in order to better detect the target object. However, we do not perform feature selection explicitly, so future work might include adding a sparsity-inducing

norm [117] during learning to explicitly encourage feature selection. Another interesting avenue could be to model and learn keypoint deformation, which would allow the tracking of deformable and articulated objects. However, this would be difficult to achieve in an online learning framework which uses self-training, as measures would need to be taken to avoid drift.

For the scene reconstruction approach presented in Chapter 5, as has already been discussed in Section 5.5, there are a number of avenues for future work which would help to improve the reconstruction results. These include how best to handle the background class during labelling, improvements to labelling accuracy, and fusing multiple 2.5D reconstructions into a global 3D reconstruction.

One thing which is clear is that the increasing ubiquity of portable, powerful, devices containing cameras makes this an exciting time for the field of computer vision in general and presents many opportunities for vision-based gaming in particular. We hope that the work presented in this thesis will have contributed some building blocks which can be built upon by others both in academia and in industry.

Appendices

Bibliography

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC Superpixels. Technical report, Ecole Polytechnique Fédérale de Lausanne, 2010.

[2] A. Adam, E. Rivlin, and I. Shimshoni. Robust Fragments-Based Tracking using the Integral Histogram. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[3] M. A. Aizerman, E. M. Braverman, and L. I. Rozonér. Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning. *Automation and Remote Control*, 25:821–837, 1964.

[4] K. Alahari, C. Russell, and P. H. S. Torr. Efficient Piecewise Learning for Conditional Random Fields. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[5] A. Alahi, R. Ortiz, and P. Vandergheynst. FREAK: Fast Retina Keypoint. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[6] S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–72, 2004.

[7] B. Babenko, M.-H. Yang, and S. Belongie. Visual Tracking with Online Multiple Instance Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011.

[8] S. Baker and I. Matthews. Lucas-Kanade 20 Years On: A Unifying Framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[9] J. W. Bastian, B. Ward, R. Hill, A. van den Hengel, and A. R. Dick. Interactive Modelling for AR Applications. In *IEEE International Symposium on Mixed and Augmented Reality*, 2010.

[10] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[11] S. Benhimane and E. Malis. Real-Time Image-Based Tracking of Planes Using Efficient Second-Order Minimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[12] C. Bibby and I. D. Reid. Robust Real-Time Visual Tracking Using Pixel-Wise Posteriors. In *European Conference on Computer Vision*, 2008.

[13] M. J. Black and A. D. Jepson. EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[14] M. B. Blaschko and C. H. Lampert. Learning to Localize Objects with Structured Output Regression. In *European Conference on Computer Vision*, 2008.

[15] A. Bordes, L. Bottou, P. Gallinari, and J. Weston. Solving multiclass support vector machines with LaRank. In *International Conference on Machine Learning*, 2007.

[16] A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast Kernel Classifiers with Online and Active Learning. *The Journal of Machine Learning Research*, 6:1579–1619, 2005.

[17] A. Bordes, N. Usunier, and L. Bottou. Sequence Labelling SVMs Trained in One Pass. In *Proc. ECML-PKDD*, 2008.

[18] J.-Y. Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker. Technical report, Intel Corporation Microprocessor Research Labs, 1999.

[19] Y. Boykov and M.-P. Jolly. Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. In *IEEE International Conference on Computer Vision*, 2001.

[20] Y. Boykov and V. Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–37, 2004.

[21] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[22] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[23] P. Buehler, M. Everingham, D. Huttenlocher, and A. Zisserman. Long Term Arm and Hand Tracking for Continuous Sign Language TV Broadcasts. In *British Machine Vision Conference*, 2008.

[24] M. Calonder. *Robust, High-Speed Interest Point Matching for Real-Time Applications.* PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2010.

[25] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *European Conference on Computer Vision*, 2010.

[26] N. D. F. Campbell, G. Vogiatzis, C. Hernández, and R. Cipolla. Using Multiple Hypotheses to Improve Depth-Maps for Multi-View Stereo. In *European Conference on Computer Vision*, 2008.

[27] K. Cannons. A Review of Visual Tracking. Technical report, York University, 2008.

[28] C.-C. Chang and C.-J. Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

[29] O. Chum and J. Matas. Matching with PROSAC Progressive Sample Consensus. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[30] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[31] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[32] K. Crammer, J. Kandola, R. Holloway, and Y. Singer. Online Classification on a Budget. In *Neural Information Processing Systems*, 2003.

[33] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[34] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–67, 2007.

[35] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov. Fast Approximate Energy Minimization with Label Costs. *International Journal of Computer Vision*, 96(1):1–27, 2011.

[36] T. G. Dietterich. Solving the Multiple Instance Problem with Axis-Parallel Rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

[37] T. G. Dietterich. An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Machine Learning*, 40(2):139–157, 2000.

[38] K.-B. Duan and S. Keerthi. Which is the Best Multiclass SVM Method? An Empirical Study. In *International Conference on Multiple Classifier Systems*, 2005.

[39] A. Elgammal, R. Duraiswami, and L. S. Davis. Probabilistic Tracking in Joint Feature-Spatial Spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[40] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[41] M. A. Fischler and R. 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, 1981.

[42] K. Fukunaga and L. Hostetler. The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975.

[43] D. Gallup, J.-M. Frahm, and M. Pollefeys. Piecewise Planar and Non-Planar Stereo for Urban Scene Reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[44] P. Gehler and S. Nowozin. On Feature Combination for Multiclass Object Classification. In *IEEE International Conference on Computer Vision*, 2009.

[45] M. Gönen and E. Alpaydn. Multiple Kernel Learning Algorithms. *Journal of Machine Learning Research*, 12:2211–2268, 2011.

[46] H. Grabner, M. Grabner, and H. Bischof. Real-Time Tracking via On-line Boosting. In *British Machine Vision Conference*, 2006.

[47] H. Grabner, C. Leistner, and H. Bischof. Semi-Supervised On-Line Boosting for Robust Tracking. In *European Conference on Computer Vision*, 2008.

[48] M. Grabner, H. Grabner, and H. Bischof. Learning Features for Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[49] G. Hager and P. Belhumeur. Efficient Region Tracking with Parametric Models of Geometry and Illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[50] A. Halevy, P. Norvig, and F. Pereira. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[51] J. M. Hammersley and P. Clifford. Markov Fields on Finite Graphs and Lattices. Technical report, Unpublished, 1971.

[52] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Proc. 4th Alvey Vision Conference*, 1988.

[53] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[54] D. Hoiem, A. A. Efros, and M. Hebert. Recovering Surface Layout from an Image. *International Journal of Computer Vision*, 75(1):151–172, 2007.

[55] H. Isack and Y. Boykov. Energy-Based Geometric Multi-Model Fitting. *International Journal of Computer Vision*, 97(2):123–147, 2011.

[56] M. Isard and A. Blake. CONDENSATION - Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[57] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust Online Appearance Models for Visual Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1296–1311, 2003.

[58] T. Joachims. Making Large-Scale SVM Learning Practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods*. MIT Press, 1999.

[59] T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-Plane Training of Structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

[60] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-Learning-Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2011.

[61] Y. Ke and R. Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[62] J. E. Kelley Jr. The Cutting-Plane Method for Solving Convex Programs. *Journal of the Society for Industrial & Applied Mathematics*, 8(4):703–712, 1960.

[63] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *IEEE International Symposium on Mixed and Augmented Reality*, Nov. 2007.

[64] V. Kolmogorov and R. Zabih. What Energy Functions can be Minimized via Graph Cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–59, 2004.

[65] H. W. Kuhn and A. W. Tucker. Nonlinear Programming. In *Second Berkley Symposium on Mathematical Statistics and Probability*, 1951.

[66] S. Kumar and M. Hebert. Discriminative Random Fields. *International Journal of Computer Vision*, 68(2):179–201, 2006.

[67] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Efficient Subwindow Search: A Branch and Bound Framework for Object Localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2129–42, 2009.

[68] Y. LeCun, L. Bottou, G. Orr, and K. Müller. Efficient BackProp. In G. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, page 546. Springer Berlin / Heidelberg, 1998.

[69] C. Leistner, A. Saffari, and H. Bischof. MIForests: Multiple-Instance Learning with Randomized Trees. In *European Conference on Computer Vision*, 2010.

[70] C. Leistner, A. Saffari, P. M. Roth, and H. Bischof. On Robustness of On-line Boosting - A Competitive Study. In *Proc. ICCV-OLCV*, 2009.

[71] V. Lepetit and P. Fua. Keypoint Recognition Using Randomized Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–79, 2006.

[72] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary Robust Invariant Scalable Keypoints. In *IEEE International Conference on Computer Vision*, 2011.

[73] T. Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, 1994.

[74] T. Lindeberg. Feature Detection with Automatic Scale Selection. *International Journal of Computer Vision*, 30(2):79–116, 1998.

[75] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[76] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *International Joint Conference on Artificial Intelligence*, 1981.

[77] H. Masnadi-shirazi, V. Mahadevan, and N. Vasconcelos. On the design of robust classifiers for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[78] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In *British Machine Vision Conference*, 2002.

[79] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):810–5, 2004.

[80] K. Mikolajczyk and C. Schmid. Scale & Affine Invariant Interest Point Detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[81] M. Muja and D. G. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *International Conference on Computer Vision Theory and Applications*, 2009.

[82] R. Newcombe and A. J. Davison. Live Dense Reconstruction with a Single Moving Camera. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[83] R. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense Tracking and Mapping in Real-Time. In *IEEE International Conference on Computer Vision*, 2011.

[84] D. Nister and H. Stewenius. Scalable Recognition with a Vocabulary Tree. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[85] S. Nowozin and C. H. Lampert. Structured Learning and Prediction in Computer Vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3-4):185–365, Mar. 2010.

[86] N. C. Oza. Online Bagging and Boosting. In *IEEE International Conference on Systems, Man and Cybernetics*, 2005.

[87] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua. Fast Keypoint Recognition Using Random Ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):448–61, 2010.

[88] M. Özuysal, V. Lepetit, F. Fleuret, and P. Fua. Feature Harvesting for Tracking-by-Detection. In *European Conference on Computer Vision*, 2006.

[89] Q. Pan, G. Reitmayr, and T. Drummond. ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition. In *British Machine Vision Conference*, 2009.

[90] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[91] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object Retrieval with Large Vocabularies and Fast Spatial Matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[92] J. C. Platt. *Fast Training of Support Vector Machines Using Sequential Minimal Optimization*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[93] B. T. Polyak and A. B. Juditsky. Acceleration of Stochastic Approximation by Averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[94] D. Ramanan, D. A. Forsyth, and A. Zisserman. Tracking People by Learning Their Appearance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):65–81, 2007.

[95] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental Learning for Robust Visual Tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2007.

[96] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, May 2006.

[97] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts. *ACM Transactions on Graphics*, 23(3):309, 2004.

[98] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an Efcient Alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision*, 2011.

[99] A. Saffari, M. Godec, T. Pock, C. Leistner, and H. Bischof. Online Multi-Class LPBoost. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[100] A. Saffari, C. Leistner, M. Godec, and H. Bischof. Robust multi-view boosting with priors. In *European Conference on Computer Vision*, 2010.

[101] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line Random Forests. In *Proc. ICCV-OLCV*, 2009.

[102] R. E. Schapire. The Strength of Weak Learnability. *Machine Learning*, 5(2):197–227, 1990.

[103] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–535, 1997.

[104] B. Schölkopf, R. Herbrich, and A. J. Smola. A Generalized Representer Theorem. In *Conference on Computational Learning Theory*, 2001.

[105] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[106] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. *Mathematical Programming*, 127(1):3–30, 2010.

[107] J. Shi and C. Tomasi. Good Features to Track. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[108] N. Z. Shor. *Minimization Methods for Non-Differentiable Functions*. Springer, 1985.

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

[110] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context. *International Journal of Computer Vision*, 81(1):2–23, 2007.

[111] S. N. Sinha, D. Steedly, and R. Szeliski. Piecewise Planar Stereo for Image-Based Rendering. In *IEEE International Conference on Computer Vision*, 2009.

[112] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *IEEE International Conference on Computer Vision*, Oct. 2003.

[113] J. Stuehmer, S. Gumhold, and D. Cremers. Real-Time Dense Geometry from a Handheld Camera. In *DAGM Symposium on Pattern Recognition*, 2010.

[114] M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs Using Graph Cuts. In *European Conference on Computer Vision*, Lecture Notes in Computer Science, 2008.

[115] B. Taskar, C. Guestrin, and D. Koller. Max-Margin Markov Networks. In *Neural Information Processing Systems*, 2003.

[116] S. Taylor and T. Drummond. Multiple Target Localisation at over 100 FPS. In *British Machine Vision Conference*, 2009.

[117] R. Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[118] P. H. S. Torr and A. Zisserman. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[119] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large Margin Methods for Structured and Interdependent Output Variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[120] T. Tuytelaars and C. Schmid. Vector Quantizing Feature Space with a Regular Lattice. In *IEEE International Conference on Computer Vision*, 2007.

[121] T. Tuytelaars and L. Van Gool. Wide Baseline Stereo Matching based on Local, Afnely Invariant Regions. In *British Machine Vision Conference*, 2000.

[122] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *IEEE International Conference on Computer Vision*, 2009.

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

[124] G. Vogiatzis and C. Hernández. Video-based, Real-Time Multi-View Stereo. *Image and Vision Computing*, 29(7):434–441, 2011.

[125] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose Tracking from Natural Features on Mobile Phones. In *IEEE International Symposium on Mixed and Augmented Reality*, 2008.

[126] Z. Wang, K. Crammer, and S. Vucetic. Multi-Class Pegasos on a Budget. In *International Conference on Machine Learning*, 2010.

[127] B. Williams, G. Klein, and I. D. Reid. Real-Time SLAM Relocalisation. In *IEEE International Conference on Computer Vision*, 2007.

[128] O. Williams, A. Blake, and R. Cipolla. A sparse probabilistic learning algorithm for real-time tracking. In *IEEE International Conference on Computer Vision*, 2003.

[129] W. Xu. Towards Optimal One Pass Large Scale Learning with Averaged Stochastic Gradient Descent. Technical report, Stony Brook University, 2010.

[130] C. Yang, R. Duraiswami, and L. S. Davis. Efficient Mean-Shift Tracking via a New Similarity Measure. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[131] B. Zeisl, C. Leistner, A. Saffari, and H. Bischof. Online Semi-Supervised Multiple-Instance Boosting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[132] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. Shape from Shading: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999.