Learning Pose Invariant and Covariant Classifiers from Image Sequences

THÈSE N^{O} 4746 (2010)

PRÉSENTÉE LE 6 JUILLET 2010

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

LABORATOIRE DE VISION PAR ORDINATEUR

PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Mustafa ÖZUYSAL

acceptée sur proposition du jury:

Prof. P. Dillenbourg, président du jury
Prof. P. Fua, Dr V. Lepetit, directeurs de thèse
Prof. W. Gerstner, rapporteur
Prof. F. Jurie, rapporteur
Prof. J. Matas, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2010

Abstract

Object tracking and detection over a wide range of viewpoints is a long-standing problem in Computer Vision. Despite significant advance in wide-baseline sparse interest point matching and development of robust dense feature models, it remains a largely open problem. Moreover, abundance of low cost mobile platforms and novel application areas, such as real-time Augmented Reality, constantly push the performance limits of existing methods. There is a need to modify and adapt these to meet more stringent speed and capacity requirements.

In this thesis, we aim to overcome the difficulties due to the multi-view nature of the object detection task. We significantly improve upon existing statistical keypoint matching algorithms to perform fast and robust recognition of image patches independently of object pose. We demonstrate this on various 2D and 3D datasets.

The statistical keypoint matching approaches require massive amounts of training data covering a wide range of viewpoints. We have developed a weakly supervised algorithm to greatly simplify their training for 3D objects. We also integrate this algorithm in a 3D tracking-by-detection system to perform real-time Augmented Reality.

Finally, we extend the use of a large training set with smooth viewpoint variation to category-level object detection. We introduce a new dataset with continuous pose annotations which we use to train pose estimators for objects of a single category. By using these estimators' output to select pose specific classifiers, our framework can simultaneously localize objects in an image and recover their pose. These decoupled pose estimation and classification steps yield improved detection rates.

Overall, we rely on image and video sequences to train classifiers that can either operate independently of the object pose or recover the pose parameters explicitly. We show that in both cases our approaches mitigate the effects of viewpoint changes and improve the recognition performance.

Keywords: Computer vision, keypoint recognition, naive Bayes, tracking-by-detection, object detection, multi-view.

Résumé

La détection et le suivi d'objets à partir d'un grand nombre de points de vue représente un problème de longue date dans le domaine de la vision par ordinateur. Malgré les progrès réalisés en correspondance de points d'intérêt épars étant donné une large ligne de points principaux et les développements des modèles d'attributs denses et robustes, ce sujet demeure un problème ouvert. De plus, l'abondance de plateformes mobiles bon marché et les domaines d'application novateurs, tels que la réalité augmentée, repoussent constamment les limites des méthodes existantes. D'où le besoin de modifier ces méthodes afin de répondre à des exigences de vitesse et capacité toujours plus accrues.

Dans cette thèse, nous nous efforçons de résoudre les difficultés inhérentes à la variété des points de vues de la tâche de détection d'objets. Nous améliorons de manière significative les algorithmes existants de correspondance statistique de points d'intérêt, afin qu'ils puissent reconnaître rapidement et de manière robuste des morceaux d'image indépendamment de la pose de l'objet. Nos nouveaux développements sont testés sur plusieurs jeux de données en deux et trois dimensions.

Ces approches statistiques nécessitent de grandes quantités de données d'apprentissage, couvrant une large gamme de points de vue. Nous développons un algorithme faiblement supervisé afin de simplifier leur apprentissage pour des objets tridimensionnels. Nous intégrons également cet algorithme dans un système de suivi-par-détection en trois dimensions, qui rend possible la réalité augmentée en temps réel.

Finalement, nous étendons l'utilisation de grands ensembles d'apprentissage avec variation graduelle de point de vue à la détection de catégories d'objet. Nous introduisions un nouveau jeu de donnée avec annotation

continue de la pose, que nous utilisons pour entraîner des estimateurs de pose pour des objets d'une seule catégorie. En utilisant le résultat de ces estimateurs pour sélectionner un classifieur entraîné spécifiquement pour une pose, notre système peut simultanément localiser des objets dans une image et identifier leur pose. Le découplage de ces deux étapes d'estimation de pose et de classification génère de meilleurs taux de détection.

En résumé, nous nous basons sur des séquences d'images et vidéos pour entraîner des classifieurs qui peuvent soit opérer indépendamment de la pose des objets, soit retrouver les paramètres de pose indépendamment. Nous montrons que dans les deux cas, notre approche limite les effets dus aux changement de point de vue et améliore la qualité de la reconnaissance.

Mots-clés: Vision par ordinateur, reconnaissance de points d'intérêt, suivi-par-détection, détection d'objets, multi-vue.

Acknowledgements

In the years I have spent at CVLab, I have had the opportunity to meet many researchers in the field and discuss with them. This has been a great source of pleasure and motivation to me. I would like to start by thanking all of them for their advice and feedback. This engaging environment is a direct result of the efforts of Prof. Pascal Fua and I would like to thank him for giving me a chance to work at CVLab. I am also grateful for his constant feedback in writing and presenting my work over many years.

I think saying that I am indebted to Dr. Vincent Lepetit would be an under statement. His creativity and suggestions are the roots and the inspirations for my thesis work. Without his support and encouragement, I certainly would not have managed to learn and do half as much as I did. Many thanks Vincent, for being the person you are...

I would like to thank the thesis committee members Prof. Wulfram Gestner, Prof. Frederic Jurie, Prof. Jiri Matas, and the president of the thesis committee Prof. Pierre Dillenbourg for evaluating this thesis and providing valuable feedback. I would like to express my gratitude especially to Prof. Jiri Matas, with whom I had the chance to discuss many of the ideas in this thesis during the course of my Ph.D. studies. His inquisitiveness has always motivated me to lift the rug and look under it.

My thanks go to Dr. François Fleuret who has provided valuable feedback on and contributed to several parts of my thesis work. I greatly benefited from many discussions with him.

I would like to acknowledge the great support from our beloved secretary Josiane Gisclon. She has always been resourceful and genuinely concerned even in the face of the most unusual problems. She was the first person I

met at EPFL and I will always remember her words: “Mustafa, you should stop worrying!”

I would also like to acknowledge the diligent work and many CPU hours of *cvlabpc15* and *cvlabpc34* that went into producing the results and the text for this thesis.

There are too numerous people in the “lab” to list here who have enriched my life and work, made the difficulties easier to carry, and caused me to smile every now and then in the face of small jokes and gestures. I would like to sincerely thank everyone in CVLab, past and present, for their kindness and friendship. It has been great to be one of you and among you.

I would like to thank in particular to Engin Tola and Jérôme Berclaz for discussions on vision, cameras, N900, why we love Thinkpads, and life in general. I feel fortunate that their Ph.D. time mostly overlapped mine.

Julien Pilet has been my office mate, Linux Firewire driver support, crazy idea deflector, and the dancing Swiss guy in my wedding. Many thanks go to him for all these and many more. Without his help and support, some of the papers and consequently this thesis would not have been completed on time.

My trip to Lausanne started together with Zafer Arıcan and Ayfer Özgür, whose companionship helped me to settle, work, and finish this thesis much more easily and peacefully.

I would like to thank my parents for being always there for me, before and after my adventure in Switzerland.

Finally, without the support, love, encouragement, and understanding from my wife, Özden, it would not be possible to start, finish, or write this thesis.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES

CHAPTER

ONE

INTRODUCTION

Since the early days of computer vision, object tracking and detection has been a very active research topic [7, 46, 53, 54, 55, 78, 88, 121] with a wide range of potential applications in robotics, human computer interfaces, and augmented reality. This is a challenging task since the 2D appearance of image features such as edges and interest points depend to a great extent on the 3D configuration and pose of the objects in the scene. Tracking methods [20, 116] cope with this problem by using pose information from previous frames to constrain the pose space and reduce feature variation. However, they require initialization, are prone to drift and fail in the presence of fast motion or strong occlusions. Consequently, successful long-term tracking requires object detection, which usually involves matching image features across a wide range of viewpoints. This brings back the need to analyze the 2D image features independent of the 3D object pose.

There are two main approaches to mitigate the effects of changing viewpoint on the object appearance. The first one relies on finding representations that do not depend strongly on the viewpoint and combining them to analyze the image features independently of the object pose. The second one relies on the observation that if we can constrain at least a subset of the pose parameters, then the image features will vary much less. We can achieve this either by estimating the pose parameters directly from the image or by searching over a range of parameter values. In both cases the image features are extracted for fixed object pose.

In this thesis, we call the former approach *pose invariant* and the latter *pose covariant*. We argue that, in practice, usually a large training set of image sequences captured from smoothly varying viewpoints can be obtained and used to train the object detectors. This kind of training data greatly improves the performance of both pose invariant and covariant approaches. Such image sequences depict a continuous variation of the image features as the viewpoint changes. Hence, they provide essential information that is required to extract pose invariant representations and to train the pose estimators used by covariant classifiers. We demonstrate this for various computer vision tasks including wide baseline feature matching, real-time 3D tracking-by-detection of a single object, and localization of objects from a single category, which we discuss in more detail below.

Wide baseline feature matching. The dominant approach to matching image features is pose covariant and relies on estimating the orientation, scale, and even affine parameters from the texture around interest points [69, 72, 74, 108]. Although this is shown to perform very well in practice [75, 76], it usually requires an explicit search over pose parameters. For example, scale estimation involves finding the maxima of Laplacian in the scale space. Therefore it can not always cope with real-time requirements. We will argue that the pose invariant approach is faster and can be made just as reliable. Despite the fact that there are no general 3D geometric invariants [81] image features do not vary arbitrarily with viewpoint [12]. The stationary statistics of these variations can be captured using a probabilistic model and machine learning techniques [58] can be used to combine many such models to obtain a mostly pose invariant classifier. Indeed, [63] uses this approach and we build upon it.

Real-time 3D tracking-by-detection. Image acquisition, storage, and processing costs have declined tremendously in the last decade. As discussed above, when we have some prior information about the kind of object we want to detect, a large training set covering a wide range of viewpoints can be captured and used to train the object detector. However, in a fully supervised setting this still becomes burdensome due to financial and time costs of manual annotation. This problem can be solved by using a weakly supervised framework that requires annotation of only a small set of im-

ages. By using a classifier that can be trained incrementally [16], it is then possible to automatically annotate the remaining training data and update the classifier on-line.

A weakly supervised training framework requires perfect output by the classifier since any labeling errors will corrupt the training set used to update the classifier. In the presence of such corrupted labels the object appearance model encoded by the classifier will drift slowly and eventually the classifier will learn parts of the background as the object. Therefore, there is a need to filter the classifier output to keep only very high confidence samples to be integrated into the classifier updates.

We have developed a weakly supervised learning framework for matching interest points on a particular object to obtain a detector that can localize it in images captured from arbitrary viewpoints. This framework exploits 3D rigidity constraints to remove false matches between the interest points detected on the frames of a video sequence that depicts the object as it slowly moves against a relatively uncluttered background. This constrained training setup is used to perform tracking and detection reliably so that the object detection and pose recovery can be fully automated and the classifier is updated after processing each new frame. In this context, reliable tracking and detection means that the labels of the set of examples used for classifier updates are mostly correct. After all frames of the training video is processed, the trained classifier can be used to obtain feature matches to images of the object captured from any of the viewpoints included in the training set.

We also take advantage of the training video to decide which interest points on the object are repeatedly detected and reliably recognized by the classifier. We keep only high performing interest points according to these two criteria. For this reason, we refer to the developed weakly supervised framework as *Feature Harvesting*. It is similar in spirit to the way a human learns complex tasks gradually, starting from a toy example. It is especially effective considering the large volume of unsupervised visual data that is available for training.

Object Localization. Finally, we have designed an approach that can localize objects of a single category in images and also estimate their pose. Since an object category, such as cars or buildings, contains instances of widely varying geometry, we consider the specification of the location and size of the object bounding box in the image as sufficient for localization.

Similarly, we do not seek to recover every aspect of the object pose which could be as complex as recovering the six degrees of freedom for 3D rotation and translation plus tens of joint angles or deformation parameters respectively for articulated or deformable objects. Instead, we assume that certain aspects of the object pose dominate the way the appearance of the object changes and they can not be modeled in a trivial way. For example a rotation around the axis perpendicular to the ground plane will change image features significantly unless the object has a strong rotational symmetry. We aim to model such pose parameters since if the image features are mostly invariant to a pose parameter then estimation of its value will require the solution of a severely ill defined inverse problem. Chapter 5 contains a more in depth discussion of which aspects of object pose can be modeled by our approach and future directions on improving the proposed localization framework.

Note that a classifier that can recover object pose has to be pose covariant, since otherwise the pose information is lost during training. As a result, most of the state-of-the-art techniques [10, 19, 28, 29, 38, 82] are not directly suitable for this task.

Moreover, existing datasets either do not contain annotated pose information at all [26, 44] or contain just descriptive viewpoint labels which are heavily quantized such as *front view, top view* [25, 102]. Since training pose covariant classifiers often requires a more accurate and even continuous annotation of object pose, it is much easier to design and test a pose invariant approach using these datasets. Lack of pose information in image datasets contrasts with the human learning process, since ego-motion is an integral part of the human visual system. We believe the reasons for this omission and tendency towards pose invariant object detection is of a practical nature: It is hard to obtain accurate pose information. To overcome this, at least in a specialized context, we have collected a dataset that contains images of cars with accurate and continuous pose information. This dataset allows us to train an object detection pipeline that simultaneously localizes the objects of a single category and estimates their pose.

Overall, the approaches that we advocate in this thesis call for increased use of video sequences that include slow variation of the viewpoint and image features for training purposes. This allows a gradual shift to weaker supervision during training. Indeed, recent tracking-by-detection approaches [4, 43, 57] already follow this trend.

(a) (b) (c)

Figure 1.1: Vehicles capable of Autonomous Navigation. **(a)** A hands-free capable car developed at Stanford University. In the future, robust driver-less navigation can significantly reduce accidents due to human factors. Photo by Steve Jurvetson (CC BY). **(b)** Autonomous flight allows data collection over a large area, in this case to aid fisherman to locate tuna fish. **(c)** NASA *Sojourner* on Mars. Autonomous rovers are an essential part of space exploration since remote control is not feasible due to latency in communication.

We also hope that the results of this thesis will convince its readers that the choice between pose invariant and covariant approaches involves fundamental trade-offs on training and run-time constraints, speed and reliability. Therefore, it should be kept in mind when designing a new method and it can be exploited to adapt object detection to different application requirements.

In the rest of this chapter, we first present several applications that can directly benefit from the results obtained in this thesis. In Section 1.2, we briefly review the relevant literature. We list the contributions of this thesis in Section 1.3. Section 1.4 outlines the contents of the thesis chapters and also summarizes the techniques we have developed for feature matching and object detection.

1.1 Applications

The main target applications of this thesis are Autonomous Navigation and Augmented Reality (AR). Both of these application domains demand robust localization of the camera and the objects in the scene.

Autonomous navigation refers to determining the course that should be taken by a vehicle without human control or supervision. Figure 1.1 shows several vehicles that can benefit from such automated guidance. The vehicle's current position and sometimes velocity is calculated constantly to correct for errors in its current path.

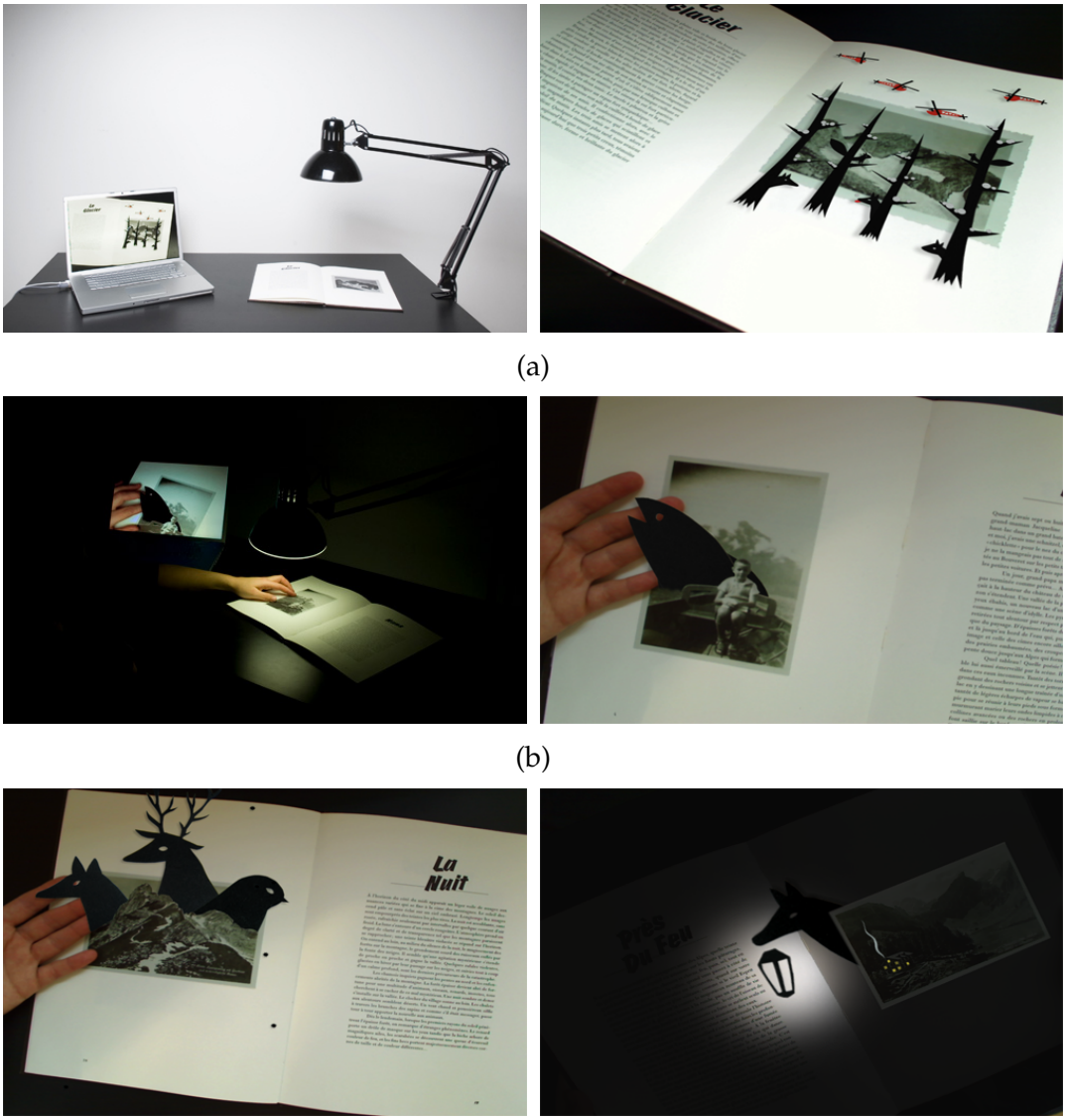


Figure 1.2: *Le Monde des Montagnes* is an animated book developed by EPFL+ECAL Lab. It uses Ferns to detect images on the book pages and to register the animations on top of the video frames. Courtesy of Camille Scherrer of EPFL+ECAL Lab [92]. **(a)** The setup to read the animated book and a sample frame depicting the animation. A camera is hidden in the lamp. **(b)** Different pages of the book are illustrated with different animations according to the story line. **(c)** Two more animation samples.

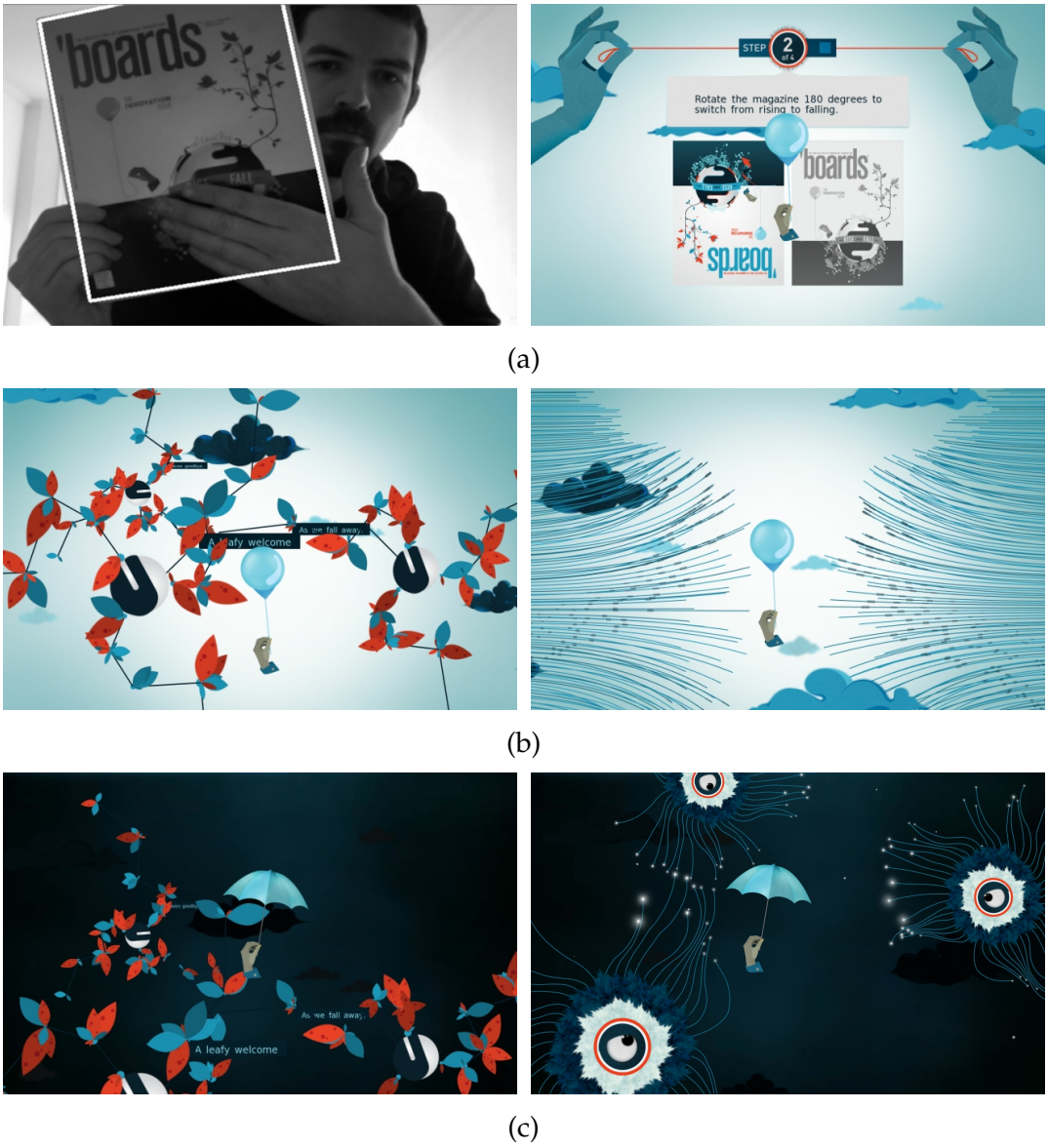


Figure 1.3: *Rise and Fall* application from Nexus Productions using Ferns for tracking. Courtesy of Emily Gobeille and Theo Watson © 2010. **(a)** The interactive application tracks the magazine cover and the scenario changes according to whether it is facing up or down. The slant of the plane determines the user's viewpoint. **(b)** Two images from the rise modality that depicts a rising balloon and its interaction with several objects. **(c)** The fall modality depicts a falling umbrella and it has a dark theme with the interactions usually resulting in negative events.

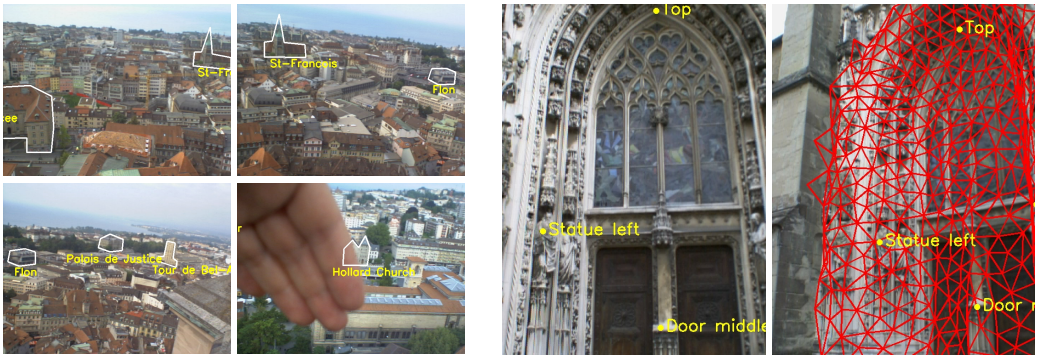
Detection of landmarks and obstacles are also necessary to guide it towards the final destination on a collision free path. In most applications, collisions are considered as fatal and have to be prevented to facilitate acceptable operation. Consequently, real-time detection and localization is an operational requirement. Moreover, it is often necessary to detect objects from far away so that the vehicle can be steered on time to avoid collision. Our keypoint matching and object detection framework has been successfully tested in the scope of the PEGASE European project [21] to detect runways for automated guidance of airplanes.

Augmented Reality requires registering virtual objects and annotations onto frames acquired by a video camera. Successful AR depends on the quality of this registration as much as on the design of virtual elements and their interaction with the user. These elements may or may not be rendered at photo realistic quality. The users will accept them as part of the real-world as long as their behavior is consistent and relatively similar to that of real objects. As a result, inaccurate position and trajectory recovery results in poor user experience.

Another factor that affects user experience is application responsiveness. Users expect their input to result in action-reaction events just as in the real world. Any delay in this process increases the user's awareness that a computer algorithm is at work. Similarly, artificial landmarks, such as fiducial markers used in older AR systems [115], help the computer algorithm to locate the camera position more reliably, but are detrimental to the atmosphere necessary to maintain an illusion of reality. They also make the application more cumbersome to deploy.

The object detection approaches we have developed only need natural image features and run at real-time speed on a variety of platforms. They can be used together with a robust tracking algorithm [109] to satisfy all of the above constraints. As shown in Figures 1.2 and 1.3, they have already been successfully integrated in several AR systems [92, 111].

In recent years, AR has generated increased interest in artistic applications. This is partly due to increased demand from artists trying to innovate new modes of interaction. However, this is not the only reason. This trend is also a sign of increased maturity of the AR algorithms that have become robust, fast, and unconstrained enough to be practical.



(a) (b)

Figure 1.4: Augmented reality for tourism. **(a)** We match an input image against a panorama and augment the scene with annotations on places of interest. **(b)** We match an image against a 3D model mode of the door, overlaid on the rightmost image. This lets us display annotations at the right place.

Detection and tracking objects in 3D can also be used to invent new modes to control artistic animations. Figure 1.3 illustrates images from an interactive storyboard application. Using our approach, pose of a magazine cover is determined at each frame to guide the story-line and also to change viewpoint of the user.

Emerging smaller hand-held devices with higher processing power and longer battery life broke the computational barriers required for AR. Mobile devices can be used to apply AR to tourism, by annotating maps, landmarks and scenery. Tourist maps contain static information about the road and places of interest. Such maps can be viewed electronically on a mobile device and various information overlays can be used to dynamically change the content. But portability requirements limit the size of these devices and their screens are usually too small to facilitate comfortable viewing. A combination of these two media can provide the best of both worlds. Figure 1.5 shows that our approach extends naturally to this domain [111]. We demonstrate similar usage by annotating images of city scenery and a landmark building in Chapter 2. Figure 1.4 shows a sample from the resulting images.

AR also finds applications in sports and advertisement. Currently, this involves more manual labor and mechanical sensors, but automated computer vision algorithms are employed increasingly more often. Visual sports commentary is usually very static and is attached to screen borders. Even rough tracking and identification

of players and racers can be used to personalize and attach such commentary to each one. A similar situation is present for live advertisements, which are usually displayed at the bottom of the frame as an overlay to the video stream. AR allows the registration of such adverts onto objects in the scene of a music clip or movie, which is less distracting for viewers. There is also the possibility to change advertisements for each channel or different airings on the same channel.

Finally, 3D detection and tracking of devices used in medical operations can help doctors to train. This usually requires more precision than we attempt to achieve in this thesis, but the same principles can be applied with tighter constraints. For example sub-pixel edge information and several cameras can be used to obtain more accurate 3D pose. Augmenting the operation sites on the patient's body is another possibility, but this requires detection of deformable surfaces, which we do not attempt in this thesis. [86] gives an excellent account of real-time methods for deformable detection and tracking.

1.2 Related Work

In this section, we briefly review the literature on feature matching and object detection. The discussion of methods that are directly related to the developments in this thesis are deferred until the corresponding chapter, where we also provide a comparison.

1.2.1 Keypoint Matching

Due to its robustness to partial occlusions and computational efficiency, recognition of image patches extracted around detected keypoints is crucial for many vision problems. As a result, two main classes of approaches to keypoint matching have been developed to achieve robustness to perspective and lighting changes.

The first family relies on covariant interest point detectors [69, 72, 74, 108] and on local descriptors designed to be robust to specific classes of deformations [69, 93]. The keypoint detector provides estimates of the scale and rotation at which the descriptor should be computed. Among such descriptors, the SIFT vector [69], computed from local histograms of gradients, has been shown to work remarkably well.

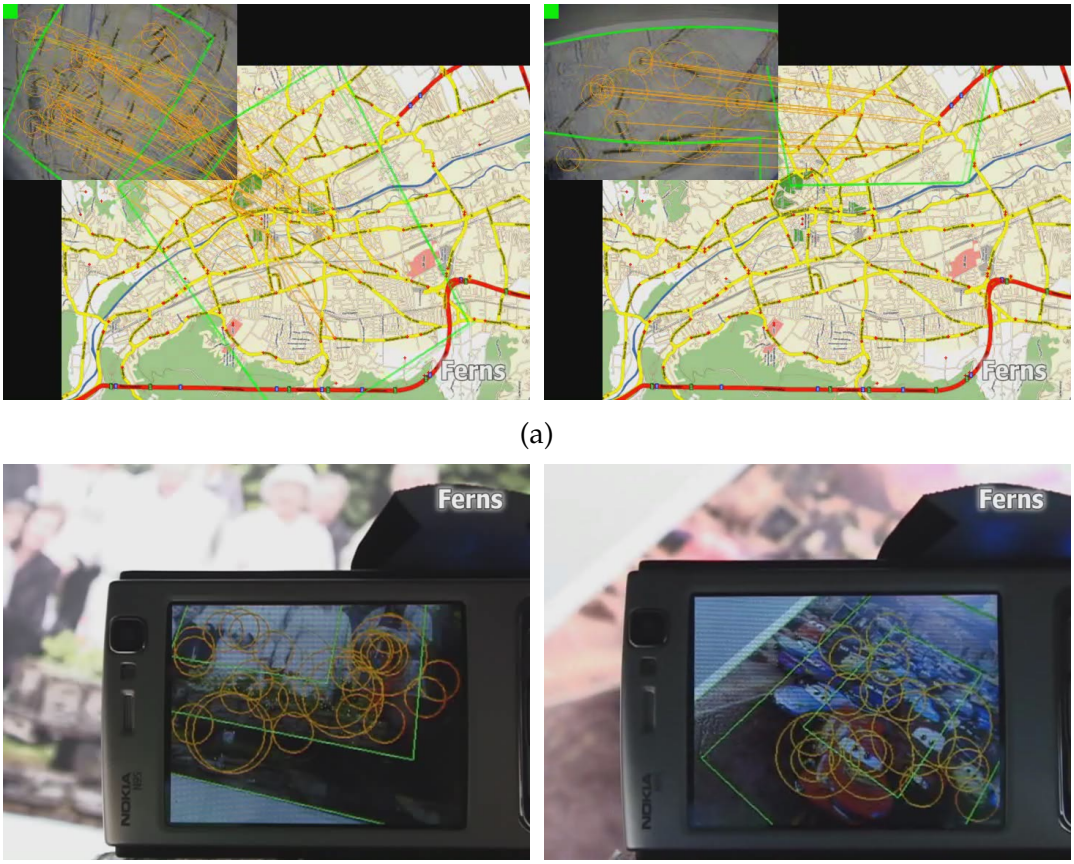


Figure 1.5: Augmented Reality on a mobile device. The lower computational power and limited memory of the mobile platform requires clever adaptations of existing techniques to meet real-time constraints. Courtesy of Gerhard Reitmayr and University of Cambridge [111]. **(a)** Augmented Reality can be used in navigation applications to provide complex real-time annotations on tourist maps. **(b)** Detection of two image models on a mobile phone.

The first step of keypoint matching using descriptors is to collect the descriptors on the images to be matched in a database. An interest point is matched by finding the nearest neighbors of its descriptor in the database. Due to efficiency concerns, usually only an approximate nearest neighbor search is performed using an efficient data structure, such as KD-Trees [8].

Since these descriptor based approaches require a complex interest point detector and the fast matching uses a complex data structure they are usually not preferred for real-time applications. A simplified version of SIFT and a Spill-Tree [67] based nearest neighbor matching algorithm has been recently used for AR on mobile devices [111]. The approach we describe in Chapter 2 is much simpler and performs equally well.

It has also been shown that keypoints can be matched by vector quantizing [98] each one by using either hierarchical K-Means clustering [83], a Randomized KD-Forest [85], or an Extremely Randomized Clustering Forest [77]. This allows fast approximate nearest neighbor computation and is used for image retrieval and classification in very large image databases. However, performance is measured in terms of the number of correctly retrieved or classified documents rather than the number of correctly classified keypoints, which is the important criterion for applications such as pose estimation or Simultaneous Localization and Mapping.

Recently, techniques have been developed to transform descriptors into short binary representations. This significantly speeds up matching since the nearest neighbor search is performed in the Hamming space and the distance computation reduces to bit counting. [96] converts SIFT vectors by learning a binary embedding that maximizes patch similarity. [106] uses Neighborhood Component Analysis [49] to find a binary representation that preserves the nearest neighbors in the descriptor space. Although it does not result in a binary vector, [113] learns a descriptor embedding through Local Discriminant Embedding that yields a high performance descriptor of a small number of dimensions. These techniques greatly improve matching speed and quality. But they still require extraction of the original descriptor and as a result they are currently not suited for real-time object detection.

A second class of feature matching algorithms relies on statistical learning techniques to compute a probabilistic model of the patch. Among these, the most relevant one for this thesis is that of [63] and we review it in detail below. Section 2.1 gives a

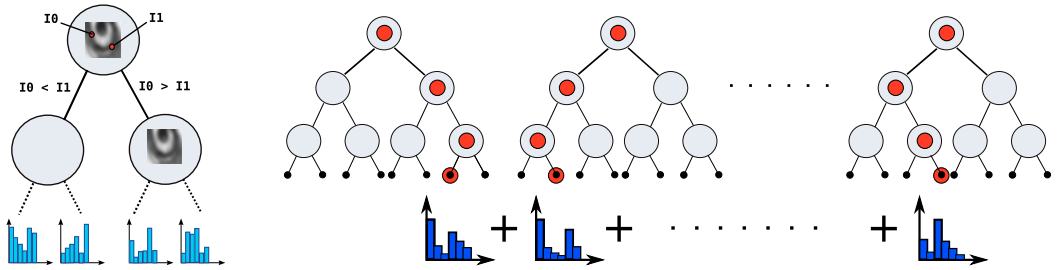


Figure 1.6: Keypoint matching using Randomized Trees. **(a)** Each image patch follows a path down the tree according to the result of a binary comparison between two pixel values at each node. The leaves contain the posterior over class labels and every keypoint has a unique class. **(b)** A new image patch is classified by dropping it down a set of Randomized trees, summing up the distributions at each leaf node reached in this way, and picking the class with the highest probability.

more detailed account of the remaining methods in this category and compares them to our feature matching approach.

1.2.2 Randomized Trees

Since the set of possible patches around an image feature under changing perspective and lightning conditions can be seen as a class, [63] showed that the feature matching problem can be solved by training a set of Randomized Trees [1] to recognize feature points independently of pose. This is done using a database of patches obtained by warping those found in a training image by randomly chosen homographies. Figure 1.6 gives an overview of keypoint matching using Randomized Trees.

Note that unlike in traditional classification problems, a close-to-perfect method is not required, because the output of the classifier can be filtered by a robust estimator. However the classifier should be able to handle many classes—typically in the order of hundreds— simultaneously without compromising performance or speed. This is required for successful object detection since the repeatability of the interest points will drop significantly under perspective distortion and modeling a high number of them ensures presence of enough features for a wide range of viewpoints. A classification tree can naturally handle such multi-class problems.

Training methods for such tree structured classifiers usually exploit the data distribution in each node to select optimal tests that reduce the expected classification error in the child nodes. Since using a single tree with many levels overfits the data, an ensemble of shorter trees is preferred by combining the output from each. When an ensemble of trees is used, randomizing the construction of each tree has been found to increase the classification performance [1, 41] for various data and problem sets.

For keypoint matching, [63] observed that picking the node tests by an optimization scheme does not reduce classification errors compared to completely random selection. This is expected since keypoint matching involves classification into hundred of classes and a single tree have very low accuracy even with the optimization scheme. Therefore, optimizing the trees individually does not significantly reduce the classification error of the ensemble. In Section 2.2.3, we show that when used as in [63], the trees function as random hash functions and total randomization is an essential component of the approach.

1.2.3 Object Detection

Given an image, object detection methods aim to check if the sought after object or objects are present in it and if they are, to localize them in the image by specifying a region of interest that contains each object. In the case we are interested in localizing only a particular object we refer to the localization problem as *instance-level* object detection. It is also possible to search for any object from a single category such as cars or pedestrians. In this case, we refer to the localization problem as *category-level* object detection. Although these two problems look somewhat similar, they involve different constraints and a method that works well for one does not necessarily perform well for the other. Therefore, different approaches have been developed for each of these problems.

As discussed in the beginning, instance-level object detection is almost a prerequisite to successful long-term object tracking. It entails locating a previously known object by using prior information on its appearance. This information can be in the form of a single image [68] or a sequence of images [89]. It is also possible to simultaneously detect multiple objects using an efficient data structure, such as a binary decision tree, to access the prior on the objects [84]. In this thesis, we use instance-level object detection within a tracking-by-detection framework to localize a target object

in the frames of an input video frame. Chapter 3 includes a more detailed review of relevant literature after introducing our approach.

Category-level object detection is one of the oldest and most difficult problems of Computer Vision. One could argue that different attempts at its solution define particular eras in the history of the field [80]. There are two issues that greatly contribute to its difficulty.

First, the variation of image features between instances of the same category can be almost as severe as variations due to viewpoint change. Although local descriptors are robust to such changes in viewpoint, they also strongly discriminate different textures. This makes them highly suitable for instance-level object detection, but significantly reduces their effectiveness for matching two instances of the same category. [31, 82, 95] try to overcome this by building a feature hierarchy that is partly or fully learned from training images. The learned features are robust to changes within each category and they can be shared between categories. Similarly, [107] uses boosting to learn the features in a discriminative way. [97] builds a descriptor using self-similarity of image patches independent of the texture element itself. [30, 56] relies on shape information extracted from image edges, which is more robust to intra-category variations than image texture.

The second issue, which we target in this thesis, is lack of a strong geometric model for an object category that works for arbitrary viewpoints. This reduces the overall robustness of the approaches since appearance alone is usually not enough to filter out all background clutter and interference between similar categories. [61] builds a geometric prior by using an *Implicit Shape Model* in Hough transform space. Image features are quantized into a codebook using K-Means and at detection time they vote for the object center. This works very well for a limited viewpoint range, but the Hough transform limits the range of views that can be included. [19] extracts an *Exemplar Model* from training data in a weakly supervised manner. This model encodes the spatial distribution of image features of class members that are similar geometric layout. [35] introduces the notion of *Pose Indexed* features by defining local coordinate frames similar to covariant interest point detectors. This again improves geometric invariance to limited changes in viewpoint an in-plane rotations, but defining such coordinate frames for 3D perspective changes is difficult. [29] recently extended the pictorial structures framework [27] to include multiple deformable part models. Each

model represents the geometric layout of a set of parts and the part locations interact with each other from a different viewpoint. A latent variable controls which model best explains the image features. A latent SVM classifier is used to train the model.

The methods above are effective to detect object categories from a limited range of viewpoints and they are also locally invariant to small changes in viewpoint. In Chapter 4, we propose a pose covariant framework that can be used in conjunction with any of these to extend the operation range of the classifier. In Section 4.1, we also compare our approach to recent 3D pose-aware literature.

1.3 Thesis Goals and Contributions

This thesis aims to develop novel approaches to learn features from image and video sequences for fast and reliable object tracking-by-detection, camera ego-motion estimation and augmented reality. We require that these sequences depict a slow variation of image features as the viewpoint is varied. This requirement allows us to extract salient feature statistics in the training phase and to increase the robustness of object detection to viewpoint changes at run-time. We specifically aimed to

- improve the robustness and reliability of fast wide-baseline feature matching approaches,
- design a fast and generic instance-level object detection approach that can be easily integrated into object tracking frameworks,
- extend these methods to 3D object tracking-by-detection,
- simplify the training phase to reduce the required amount of prior knowledge on object appearance and geometry,
- apply similar ideas to category-level object detection to improve robustness to viewpoint changes,
- and to link the developed approaches with a mathematical framework to provide a better understanding of their advantages and limitations.

The common theme linking all these goals is the special emphasis placed on learning to be robust to viewpoint changes. As stated in the beginning of the thesis, there are multiple ways to achieve this and we have exploited both pose invariant and covariant alternatives to reach the set of goals stated above. In this way, we gained the flexibility necessary to develop novel frameworks that compete with the current state-of-the-art approaches.

Among our contributions, the *Ferns* algorithm to match keypoints and its software implementation found the most wide spread distribution. It has been downloaded more than five thousand times in the first four months of the year 2010. Our *Feature Harvesting* approach to training 3D object detectors can easily be integrated into existing tracking-by-detection applications and it achieves real-time and reliable detection. To the best of our knowledge, our work on object category pose estimation is the first one to report a detailed analysis of the continuous pose estimation quality. Our image database is currently used by researchers working on manifold learning and its application to pose regression.

In Section 2.2.3, we link keypoint matching using Ferns and Randomized Trees to Locality Sensitive Hashing [3], which provides a well-known mathematical framework. This greatly enhanced our understanding of the benefits and limitations of both approaches. However, the analysis we present assumes that the image patches are already transformed from the Euclidean space of intensities to the Hamming space of features, which are taken to be intensity comparisons. More work is required to highlight the effects of the feature extraction step and possible alternatives to it. We discuss future directions in Chapter 5.

1.4 Organization of the Thesis

In this section, we outline the thesis contents and the frameworks we have developed. We omit the technical details and just present the general ideas together with the associated assumptions and requirements. We give pointers to thesis chapters where appropriate. We first describe a 3D wide-baseline feature matching framework that requires only weak supervision. We then summarize our pose covariant object detection pipeline.

1.4.1 Framework for Keypoint Matching

Our keypoint matching framework is motivated by the approach of [63]. To achieve wide-baseline viewpoint invariance, it introduces the concept of a *view-set*, which is the set of all possible images of an image patch that is centered around a keypoint. It usually covers a wide range of views but can easily be constrained to exclude a selected viewpoint range depending on the application. A multi-class classifier can then be trained with this view-set to recognize image patches of several hundred keypoint classes independent of the viewpoint. Chapter 2 presents a naive Bayesian classifier that can perform this task efficiently and much better than the Randomized Trees used in [63].

As discussed in Section 1.2.2, it is easy to obtain the view-set of keypoints detected on the images of planar surfaces by sampling the homography space and applying these to a single known image of the surface. However in case of a 3D object, self-occlusions and depth variations cause more complex visual changes that are very hard to model. It is even more tedious to simulate complex lighting effects such as specular reflections and shadows even for a planar object. Figure 1.7 illustrates our framework for keypoint matching and 3D tracking-by-detection.

We overcome these difficulties by using real-world image sequences during training. The viewpoints in the image sequence have to cover the view-set for all keypoints to be learned. To simplify the training, we require the background to be relatively uncluttered and the object has to move slowly with respect to the camera. Thanks to these simplifications, unlike [63], we do not require the knowledge of the camera position for all frames in the view-set. Our approach only needs a rough geometric model of the object, which can be as simple as an ellipsoid, and its position in the first frame. We then train an initial classifier and use it to automatically track the keypoints and to estimate the remaining viewpoints. During this process, the classifier is updated on-line with the image patches extracted from only the most recent frame. Section 3.2 describes how to perform these updates and Section 3.3 gives the details of the training steps.

It is usually better to select a distinctive subset of the keypoints on the object rather than trying to learn the appearance of all. This both speeds up detection and also improves robustness by reducing the number of wrong matches. Since we use a long

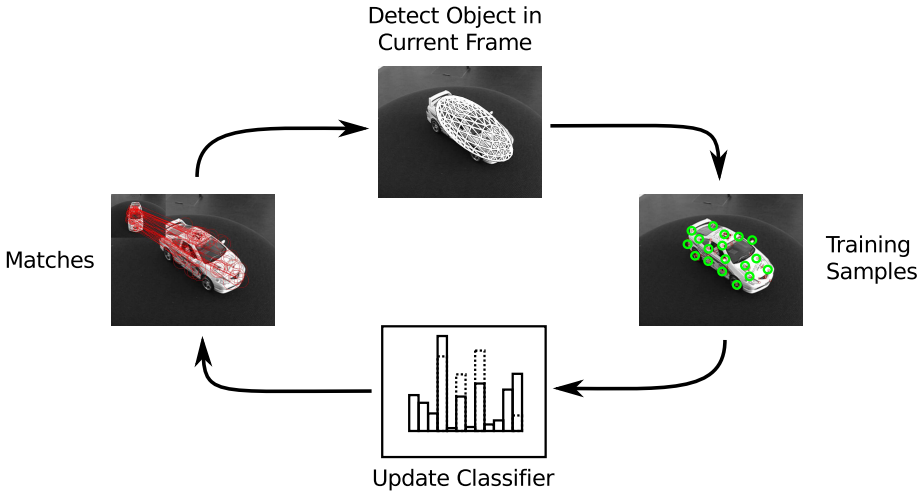


Figure 1.7: Our keypoint matching framework relies on a statistical description of keypoint appearance under perspective distortion. The training phase starts from a known object pose and image patches around keypoints are used to train an initial classifier. This classifier can match the learned keypoints to the ones detected on similar images and recover the object pose. These newly registered frames generate more training patches. 3D rigid geometric constraints are used to filter these to ensure that the set of training patches do not contain erroneous labels. The classifier is updated with these without the need to store previously observed image patches. We stop the updates at the end of training and the classifier is ready to detect the object at run-time from the viewpoints present in the training set.

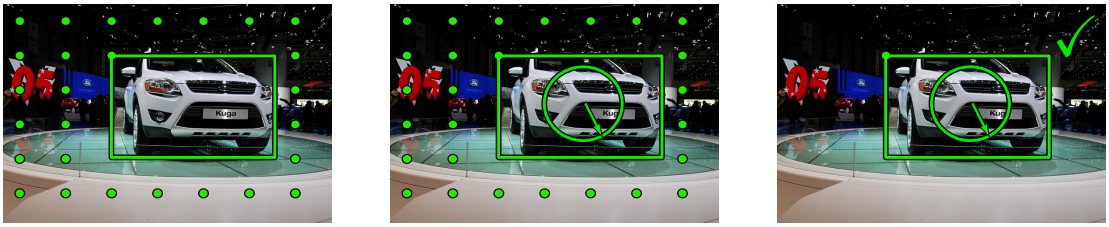
image sequence for training, we can gather performance statistics for each keypoint. We keep track of the number of times each keypoint has been detected and also the percentage of frames it is correctly recognized. We filter out keypoints with lowest scores. The experimental results in Section 3.4 show that this eliminates keypoints arising from reflections and occlusion boundaries, which are unstable, and also keypoints detected on ambiguous texture, which are hard to recognize.

1.4.2 Framework for Object Detection

Our object detection framework replicates the way covariant interest point detectors work. These detectors estimate the orientation and scale at which the image patch around interest points should be analyzed. Their estimates guide the extraction of local descriptors and as a result which points are matched to each other. This assumes that given an image patch, irrespective of its texture, we can define a canonical pose and recover orientation and scale with respect to it.

In Section 4.2, we show that it is possible to train similar estimators for bounding box dimensions and object pose that work on images of objects from a specific category. Given any 2D point in the image, the first of these estimators predicts the size of the area of interest. The second one returns an estimate of the object pose. Both estimates are given irrespective of whether there is actually an object or not. As a result on a background image patch these estimators produce mostly random results. Indeed, we do not use any background patches to train them. The estimators assume the existence of an object. This greatly simplifies their design since they do not need to discriminate between objects and background which usually contains widely varying feature statistics. Figure 1.8 shows a sample run of this multi-step estimation and classification approach.

To detect objects in a given image, we simply generate object hypotheses on a moderately dense grid. We run the estimators trained as above to obtain the area of interest and object pose at each location. We then extract image features using the estimated bounding box and compute a score for object presence using a pose specific classifier. This classifier is trained using positive examples from only a limited pose range that is as close as possible to the estimated object pose and all available negative examples. We have one such classifier for every possible output of the pose estimator and its output determines which one will be selected. Note that if there is an object at



(a) (b) (c)

Figure 1.8: Framework for object detection. **(a)** We scan the image at the corners of a relatively dense grid but at every test location the area of interest is estimated directly from image content instead of searching thru all possible sizes. **(b)** For each location, we estimate the object pose from image features extracted from the area of interest. **(c)** A classifier tuned to the estimated orientation checks for the existence of an object.

any tested location, estimated pose should be close to the real object pose. As a result correct classifier will be used to assign a high score to this location. In case there is no object, pose estimator output will select a classifier almost at random. Since all pose specific classifiers should assign a low score to background patches this randomness should not cause any misclassification. Finally, a non-maxima suppression algorithm is run on the computed scores and they are thresholded to obtain regions containing an object. The details of each step of the object detection pipeline are presented in Chapter 4.

This detection pipeline decouples object/background decision from feature variations due to viewpoint changes. The positive examples that are presented to each pose specific classifier are relatively similar in appearance compared to objects of different poses. Therefore, the training for these classifiers can easily focus on image features that are not likely to appear on the background. Moreover, the pose estimators are not confused by the complex background statistics. They just need to select features that correspond to a change in viewpoint. This decoupling is fundamental to higher detection and localization performance and requires just the existence of reliable and stable pose estimators. If for an object category the image features depend on viewpoint a lot then it is easier to satisfy this requirement. In Section 4.4, we discuss these issues in more detail.

CHAPTER

TWO

RANDOM FERNS

In this chapter, we argue that keypoint recognition using Randomized Trees [63], as described in Section 1.2.1, is effective and fast but can be much improved in two respects. First, the total randomization of the node tests renders the tree structure unnecessary. Second, the combination of keypoint class probabilities by averaging is overly conservative and can be profitably replaced by a naive Bayesian scheme. We first discuss the rationale behind our modifications, formalize our approach in Section 2.2, and finally show that it yields increased performance and scalability compared to Randomized Trees in Section 2.3.

In Section 1.2.2, we have reviewed keypoint matching using Randomized Trees, and noted that picking the node tests at random yields similar classification performance as optimizing the tests by entropy minimization. We argue that the discriminative power is the result of the data distributions at each leaf node and not the tree structure itself. Therefore, replacing the tree structure with a non-hierarchical set of tests, that we call as a *Fern*, will be equally effective for keypoint recognition. Indeed, as shown in Figures 2.1 and 2.2, a random fern can be considered as a simplified random tree. This simplification yields a faster and more compact implementation without performance loss. In Section 2.2.3, we use this simplified structure to link keypoint matching using Randomized Trees and Ferns to a probabilistic version of Locality-Sensitive Hashing [42] and provide more insight into both methods.

[63] computes the distribution over keypoint classes by averaging the response of each Randomized Tree. This can be considered optimal in case each Tree computes a

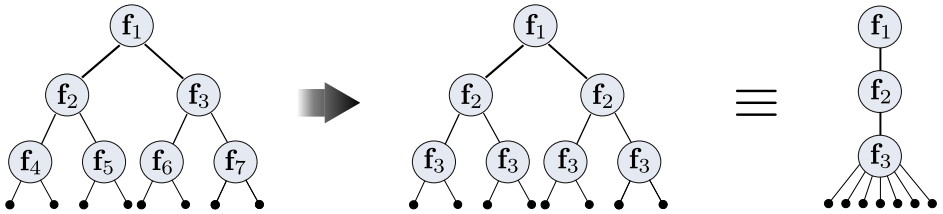


Figure 2.1: Ferns versus Trees. A tree can be transformed into a Fern by performing the following steps. First, we constrain the tree to systematically perform the same test across any given hierarchy level, which results in the same feature being evaluated independently of the path taken to get to a particular node. Second, we do away with the hierarchical structure and simply store the feature values at each level. This means applying a set of tests to the patch, which is what Ferns do.

posterior class distribution corrupted by additive Gaussian noise. However, when an additive mixture is used to combine estimates for the keypoint class from each tree, the resulting distribution has higher variance than the individual mixture components. Consequently it will be relatively flat and have low discriminative power. We use an alternative approach and combine the response from each Fern multiplicatively. Unlike averaging, the product models can represent much sharper distributions [50]. In other words, if a single Fern strongly rejects a keypoint class, it can counter the combined effect of all the other Ferns that gives a weak positive response. This increases the discriminative power but requires larger amounts of training data and the help of a prior regularization term that we will introduce in Section 2.2.

Such multiplicative combination implies a naive Bayesian formulation that assumes class conditional independence between the distributions computed for each Fern. Although this assumption will not hold in general, the classification task, which just picks a single class, will not be adversely affected by the approximation errors in the joint distribution as long as the maximum probability is assigned to the correct class [24, 37]. To the best of our knowledge, a clear theoretical argument motivating the superiority of naive Bayesian techniques in certain application domains does not exist. There is however strong empirical evidence that they are very effective in several problem domains such as spam filtering [90]. We will show that such a naive combination is a powerful alternative to additive mixtures for keypoint recognition.

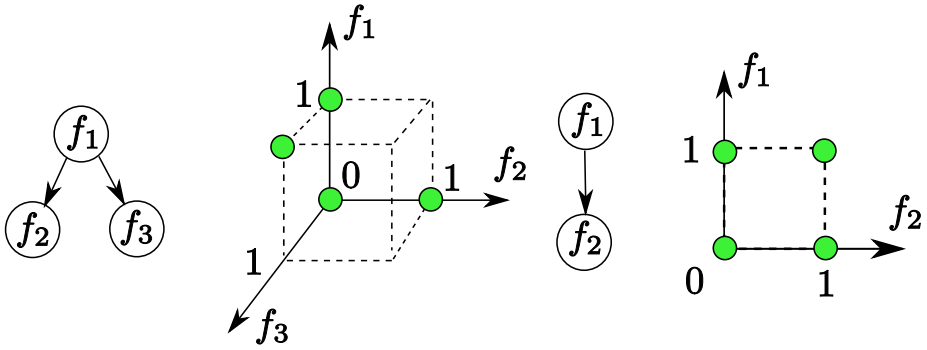


Figure 2.2: The feature spaces of Trees and Ferns. Although the space of tree features seems much higher dimensional, it is not because only a subset of features can be evaluated. **(a)** For the simple tree on the left only the four combinations of feature values denoted by green circles are possible. **(b)** The even simpler Fern on the left also yields four possible combinations of feature values but a much simpler structure. As we will show in Section 2.3, this simplicity does not entail any performance loss in the case of randomly selected features.

2.1 Related Work

We have reviewed the related work on keypoint recognition using descriptors and also Randomized Trees in Section 1.2.1. Here we present a more detailed outlook on methods that use statistical learning techniques that compute a probabilistic model of the patch.

The one-shot approach of [26] uses PCA and Gaussian Mixture Models but does not account for perspective distortion.

Local Binary Patterns [103] rely on binary feature statistics by describing the underlying texture in terms of histograms of binary features over all pixels of a target region. While such a description is appropriate for texture classification, it is not directly suitable for keypoint characterization since histograms are built in this way, it will lose the spatial information between the features. By contrast, we compute the statistics of the binary features over example patches seen from different viewpoints and use independence assumptions between groups of features, hence using many more features centered on the keypoint location, to improve the recognition rate.

The Real-Time SLAM method of [112] also extends the Randomized Trees into lists of features (similar to our Ferns), but the full posterior distribution over binary

features are replaced by a single bit. This design choice is aimed at significantly reducing the memory requirements while correctly matching the sparse set of visible landmarks at a time. For maximum performance, we model the full joint probability. Memory requirements can be tackled by using fixed point representations that require fewer bits than the standard floating point representation.

[13] trains a set of Randomized Trees with a couple hundred keypoint classes. This *base set* of keypoints is sampled at random from detections on the images of natural scenery. New keypoints are represented by the distributions output by the trained Randomized Trees. For each keypoint, the identity of the base set keypoints corresponding to the maxima of these distributions is stable under viewpoint variations. Therefore only a few elements are kept and the rest is set to zero. This results in a sparse descriptor for keypoints that is called a *keypoint signature*. [15] employs Compressive Sensing techniques [5] to compact these signatures into short vectors of 176 dimensions. [14] optimizes the selection of the base set using Genetic Algorithms [122] to yield even shorter descriptors. While the signatures are not entirely pose invariant, this method combines the statistical representations with a descriptor based approach to provide speed and compactness at the same time.

Trees and Ferns have also been used for image classification, as a replacement for a multi-way Support Vector Machine [11]. The binary features are computed on shape and texture descriptors, hence gaining invariance to local deformations. The distributions are computed over different instances of the same class, but unlike our approach the posteriors from different Trees and Ferns are combined by averaging. The results match our own observations that using either Fern or Tree structures leads to similar classification performance.

A recent evaluation on template based tracking techniques [66] also reports that Ferns have improved performance compared to approaches with similar run-time speed such as SURF [6].

2.2 A Semi-Naive Bayesian Approach to Patch Recognition

In this section we argue that, when the tests are chosen randomly, replacing the Randomized Trees of [63] by our non-hierarchical ferns and pooling their answers in a Naive Bayesian manner yields better results and scalability in terms of number of

classes. As a result, we can combine many more features, which is key to improved recognition rates.

We first show that our non-hierarchical Ferns fit nicely into a Naive Bayesian framework and then explain the training protocol which is similar to the one used for RTs.

2.2.1 Formulation of Feature Combination

As discussed in Section 1.4.1 we treat the set of all possible appearances of the image patch surrounding a keypoint as a class. Therefore, given the patch surrounding a keypoint detected in an image, our task is to assign it to the most likely class. Let $c_i, i = 1, \ldots, H$ be the set of classes and let $f_j, j = 1, \ldots, N$ be the set of binary features that will be calculated over the patch we are trying to classify. Formally, we are looking for

$$\hat{c_i} = \underset{c_i}{\text{argmax}}\, P(C = c_i \,|\, f_1, f_2, \ldots, f_N)\,, \tag{2.1}$$

where C is a random variable that represents the keypoint class. Bayes' Formula yields

$$P(C = c_i \,|\, f_1, f_2, \ldots, f_N) = \frac{P(f_1, f_2, \ldots, f_N \,|\, C = c_i) P(C = c_i)}{P(f_1, f_2, \ldots, f_N)}\,. \tag{2.2}$$

Assuming a uniform prior $P(C)$, since the denominator is simply a scaling factor that is independent from the class, our problem reduces to finding

$$\hat{c_i} = \underset{c_i}{\text{argmax}}\, P(f_1, f_2, \ldots, f_N \,|\, C = c_i)\,. \tag{2.3}$$

In our implementation, the value of each binary feature f_j only depends on the intensities of two pixel locations $\mathbf{d}_{j,1}$ and $\mathbf{d}_{j,2}$ of the image patch. We therefore write

$$f_j = \begin{cases} 1 \text{ if } I(\mathbf{d}_{j,1}) < I(\mathbf{d}_{j,2}) \\ 0 \text{ otherwise} \end{cases}\,, \tag{2.4}$$

where I represents the image patch. Since these features are very simple, we require many ($N \approx 300$) for accurate classification. Therefore a complete representation of the joint probability in Equation (2.3) is not feasible since it would require estimating and storing 2^N entries for each class. One way to overcome this difficulty is to

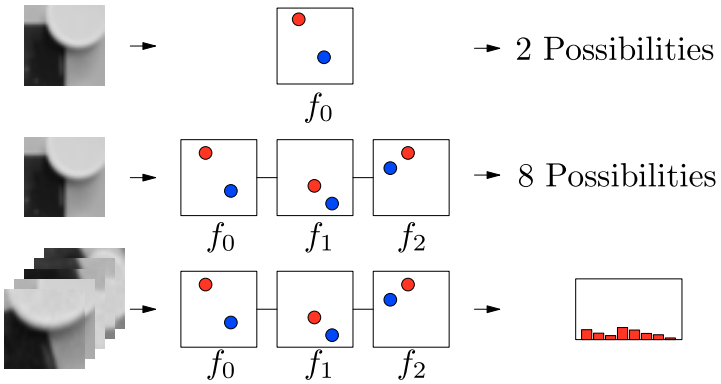


Figure 2.3: Image features and the conditional probability distribution for a single Fern. A single node in a Fern outputs a binary number depending on the result of an intensity comparison. A set of S nodes forms a Fern and for each image patch it generates a binary number with S bits. A set of patches of the same keypoint class generates a distribution over S-bit Fern outputs.

assume independence between features. An extreme version is to assume complete independence, that is,

$$P(f_1, f_2, \ldots, f_N \,|\, C = c_i) = \prod_{j=1}^{N} P(f_j \,|\, C = c_i)\,. \tag{2.5}$$

However, this completely ignores the correlation between features. To make the problem tractable while accounting for these dependencies, a good compromise is to partition our features into M groups of size $S = \frac{N}{M}$. These groups are what we define as *Ferns* and we compute the joint probability for features in each Fern. The conditional probability becomes

$$P(f_1, f_2, \ldots, f_N \,|\, C = c_i) = \prod_{k=1}^{M} P(F_k \,|\, C = c_i)\,, \tag{2.6}$$

where $F_k = \{f_{\sigma(k,1)}, f_{\sigma(k,2)}, \ldots, f_{\sigma(k,S)}\}, k = 1, \ldots, M$ represents the k^{th} fern and $\sigma(k, j)$ is a random permutation function with range $1, \ldots, N$. Hence, we follow a Semi-Naive Bayesian [120] approach by modelling only some of the dependencies between features. The viability of such an approach has been shown by [51] in the context of image retrieval applications. Figure 2.3 illustrates both the image features, their combination as a Fern and computation of the conditional probability distributions.

Algorithm 1: The pseudo-code of the run-time algorithm that computes $P(f_1, f_2, \ldots, f_N \,|\, C = c_i)$ as given by Equation (2.6) to classify the image patch I, where *index* is an integer index computed from the binary features. No image rectification, illumination normalization, or parameter tuning are required.

```
for i = 1 to H do
    log P_{I|C}[i] ⟵ 0;
end
forall Fern F_k do
    index ⟵ 0;
    for j = 1 to S do
        index ⟵ 2 × index;
        if I(d_{σ(k,j,1)}) < I(d_{σ(k,j,2)}) then
            index ⟵ index + 1;
        end
    end
    for i = 1 to H do
        log P_{I|C}[i] ⟵ log P_{I|C}[i] + log P_{F_k}[index, i];
    end
end
```

This formulation yields a tractable problem that involves $M \times 2^S$ parameters, with M between 30-50. In practice, as will be shown in Section 2.3, $S = 11$ yields good results. $M \times 2^S$ is therefore in the order of 80000, which is much smaller than 2^N with $N \approx 450$ that the full joint probability representation would require. Our formulation is also flexible since performance/memory trade-offs can be made by changing the number of Ferns and their sizes. The pseudo-code, given in Algorithm 1, shows the simplicity of the implementation.

Note that we use randomization in feature selection but also in grouping. An alternative approach would involve selecting feature groups to be as independent of each other as possible. This is routinely done by Semi-Naive Bayesian classifiers based on a criteria such as the mutual information between features [17]. However, in practice, we have not found this to be necessary to achieve good performance. We have therefore chosen not to use such a strategy to preserve the simplicity and efficiency of our training scheme and to allow for incremental training.

2.2.2 Training

We assume that at least one image of the object to be detected is available for training. We call any such image as a *model* image. Training starts by selecting a subset of the keypoints detected on these model images. This is done by deforming the images many times, applying the keypoint detector, and keeping track of the number of times the same keypoint is detected. The keypoints that are found most often are assumed to be the most stable and retained. These stable keypoints are assigned a unique class number.

To generate the training set, we represent affine image deformations as 2×2 matrices of the form $R_\theta R_{-\phi} \mathrm{diag}(\lambda_1, \lambda_2) R_\phi$, where $\mathrm{diag}(\lambda_1, \lambda_2)$ is a diagonal 2×2 matrix and R_γ represents a rotation of angle γ. The training set for each class is formed by generating 10000 sample images with randomly picked affine deformations by sampling the deformation parameters $\{\theta, \phi, \lambda_1, \lambda_2\}$ from a uniform distribution. We add Gaussian noise to each sample image and then smooth it with a Gaussian filter of size 7×7. This increases the robustness of the resulting classifier to run-time noise, especially when there are features that compare two pixels on a uniform area.

The training phase estimates the class conditional probabilities $P(F_m \,|\, C = c_i)$ for each Fern F_m and class c_i, as described in Equation (2.6). For each Fern F_m we write these terms as:

$$p_{k,c_i} = P(F_m = k \,|\, C = c_i)\,, \tag{2.7}$$

where we simplify our notations by considering F_m to be equal to k if the base 2 number formed by the binary features of F_m taken in sequence is equal to k. With this convention, Ferns can take $K = 2^S$ values and, for each one, we need to estimate the $p_{k,c_i}, k = 1, 2, \ldots, K$ under the constraint

$$\sum_{k=1}^{K} p_{k,c_i} = 1. \tag{2.8}$$

The simplest approach would be to assign the maximum likelihood estimate to these parameters from the training samples. For parameter p_{k,c_i} it is

$$p_{k,c_i} = \frac{N_{k,c_i}}{N_{c_i}}, \tag{2.9}$$

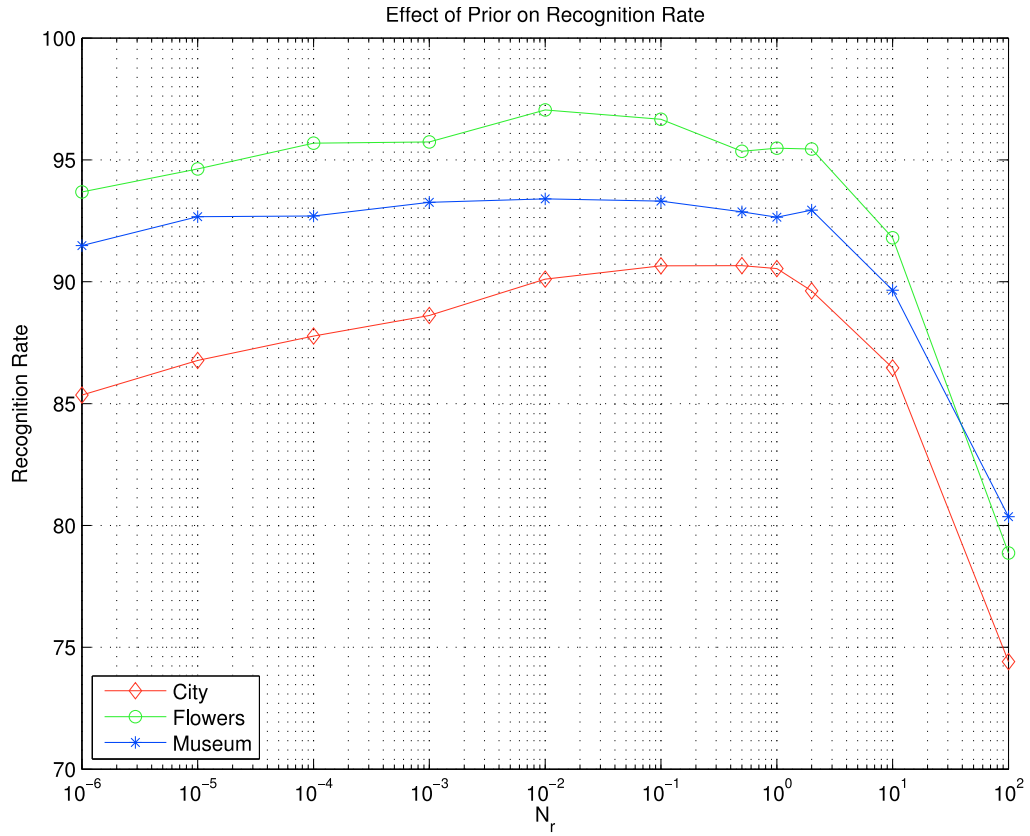


Figure 2.4: Recognition rate as a function of $\log(N_r)$ using the three test images of Section 2.3. The recognition rate remains relatively constant for $0.001 < N_r < 2$. For $N_r < 0.001$ it begins a slow decline, which ends in a sudden drop to about 50% when $N_r = 0$. The rate also drops when N_r is too large because too strong a prior decreases the effect of the actual training data, which is around 10000 samples for this experiment.

where N_{k,c_i} is the number of training samples of class c_i that evaluate to Fern value k and N_{c_i} is the total number of samples for class c_i. These parameters can therefore be estimated for each Fern independently.

In practice however, this simple scheme yields poor results because if no training sample for class c_i evaluates to k, which can easily happen when the number of samples is not infinitely large, both N_{k,c_i} and p_{k,c_i} will be zero. Since we multiply the p_{k,c_j} for all Ferns, it implies that, if the Fern evaluates to k, the corresponding patch can *never* be associated to class c_i, no matter the response of the other Ferns. This makes the Ferns far too selective because the fact that $p_{k,c_i} = 0$ may simply be an artifact of the necessarily limited size of the training set. To overcome this problem we take p_{k,c_i}

to be

$$p_{k,c_i} = \frac{N_{k,c_i} + N_r}{N_{c_i} + K \times N_r}\,, \tag{2.10}$$

where N_r represents a regularization term, which behaves as a uniform Dirichlet prior [9] over feature values. If a sample with a specific Fern value is not encountered during training, this scheme will still assign a non-zero value to the corresponding probability. As illustrated by Figure 2.4, we have found our estimator to be insensitive to the exact value of N_r and we use $N_r = 1$ in all our experiments. However, having N_r be strictly greater than zero is essential. This tallies with the observation that combining classifiers in a naive Bayesian fashion can be unreliable if improperly done [58].

In effect, our training scheme marginalizes over the pose space since the class conditional probabilities $P(F_m \,|\, C = c_i)$ depend on the camera poses relative to the object. By densely sampling the pose space and summing over all samples, we marginalize over these pose parameters. Hence at run-time, the statistics can be used in a pose independent manner, which is key to real-time performance. Furthermore, the training algorithm itself is very efficient since it only requires storing the N_{k,c_i} counts for each fern while discarding the training samples immediately after use, which means that we can use arbitrarily many if need be.

2.2.3 Ferns and Locality-Sensitive Hashing

Locality-Sensitive Hashing [42] (LSH) is an algorithm to perform approximate nearest neighbor (ANN) search in high dimensional spaces. The original LSH version uses an ensemble of binary random hashes to constrain this search to only a subset of the data points. Since a Fern can also be considered as a random hash function, we can link the two approaches and show that the Ferns perform a probabilistic version of LSH to achieve ANN classification at constant time, i.e. independent of the number of training samples. We first give a brief overview of LSH. We will then show that given a binary feature representation of a query patch, Ferns compute an estimate for the keypoint class of the nearest neighbor (NN) in the training set.

Locality-Sensitive Hashing. Locating the exact NN of a point in high dimensional spaces is a difficult problem and the run-time of a brute-force implementation scales

linearly with the number of points in the dataset. Even complex data structures such as Kd-Trees can not perform much better [3]. However, often an approximate solution is enough and it can be found efficiently. In particular, given a query point $q \in \mathbb{R}^d$, LSH returns with high probability a point p whose distance to q is at most within a constant factor c of the distance of the NN point $R \in \mathbb{R}$. p is called a c-approximate R-near neighbor and satisfies

$$\| p - q \| \leq cR \text{ with probability } (1 - \delta), \tag{2.11}$$

if $\exists\, r \in \mathbb{R}^d$, $\| p - r \| = R$, $R > 0$, $\delta > 0$. c and δ are design parameters representing the quality of approximation compared to the exact NN search in which case $c = 1$ and $\delta = 0$. To find the point p, LSH first transforms the query point q using a set of hash functions $\mathcal{H} = \{h_1, h_2, \ldots, h_M\}$ and then searches within dataset points that have the same hash value for at least one element of $\mathcal{H}$. This significantly speeds up the search for the ANN and we can increase the number of hash functions M to increase the number of successful retrievals by decreasing δ.

However for p to satisfy Equation (2.11), the hash functions h_k should be selected such that they are more likely to transform close points to close hash values than points that are far away. Such a hash function satisfies $\forall\, p, q \in \mathbb{R}^d$

$$\text{if} \quad \| p - q \| \leq R \quad \text{then} \quad P\left(h_k(p) = h_k(q)\right) \geq \alpha, \tag{2.12}$$

$$\text{if} \quad \| p - q \| > cR \quad \text{then} \quad P\left(h_k(p) = h_k(q)\right) < \beta, \tag{2.13}$$

where $\alpha > \beta$ and is called *locality-sensitive*.

It is then straightforward to show that projections onto a random subset of dimensions in the Hamming space are locality-sensitive. Considering two points $p, q \in \{0, 1\}^d$, if the Hamming distance between them is $\| p - q \|_H = R$ then a random projection onto a subset of dimensions will result in the same hash code with probability $(d - R)/d$, and we can write

$$P\left(h_k(p) = h_k(q)\right) = 1 - \frac{R}{d}. \tag{2.14}$$

Plugging this into Equations (2.12) and (2.13) yields $\alpha = 1 - R/d$, $\beta = 1 - cR/d$. We can also reasonably require $c > 1$, since $c = 1$ means exact nearest neighbor retrieval and we seek an approximate solution. This leads to $\alpha > \beta$ and the random projections will be locality-sensitive [3].

As explained in Section 2.2.2, at the training step we perform the pixel comparisons of Equation (2.4) to every training sample. This transforms the image patches into a high dimensional Hamming space with $d \approx 400$. Since each Fern uses only a subset of these d dimensions, typically $S = 11$, each Fern is a locality-sensitive random hash function in this feature space.

ANN for keypoint matching. Once we obtain the binary hash codes for each training sample, we can recognize new keypoint patches by finding their nearest neighbors in the Hamming space and classify each one as the same class as its NN. However, even an ANN algorithm can not run at high frame rates since we require thousands of training samples for each class due to the simplicity of the features.

The semi-naive Bayesian approach we have formalized in Section 2.2.1 solves this problem by estimating the keypoint class of the NN in the Hamming space instead of actually locating it. More specifically, the training step computes the joint probability distribution of hash codes and keypoint classes. This allows us to use a large number of training samples without adversely affecting the run-time speed, to update these distributions as more training data becomes available as in Chapter 3, and a very simple and fast classification step as shown in Algorithm 1. What we lose is the location of the actual NN and its Hamming distance to the binary representation of the query image patch. We will show that at least for a wide range of problems involving several hundreds of keypoint classes benefits far outweigh this loss.

2.3 Comparison with Randomized Trees

In this Section we compare RTs and Ferns. We separately quantify the effects of the structural difference and the change in the probabilistic model.

For this comparison, we experimented with the three images of Figure 2.5. We warp each image by repeatedly applying random affine deformations and detect Harris corners in the deformed images. We then select the most stable 250 keypoints per image based on how many times they are detected in the deformed versions to use in the following experiments and assign a unique class id to each of them. The classification is done using patches that are 32×32 pixels in size.

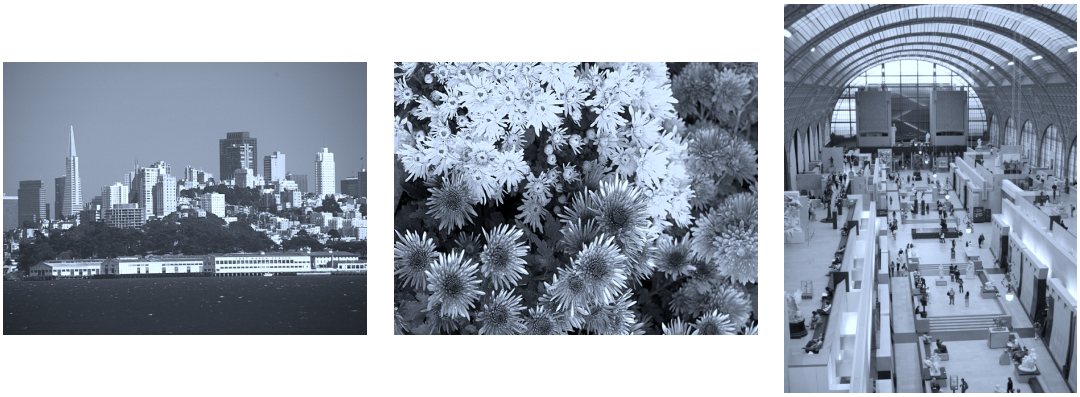


Figure 2.5: The recognition rate experiments are performed on three images that show different texture and structures. Image size is 640×480 pixels.

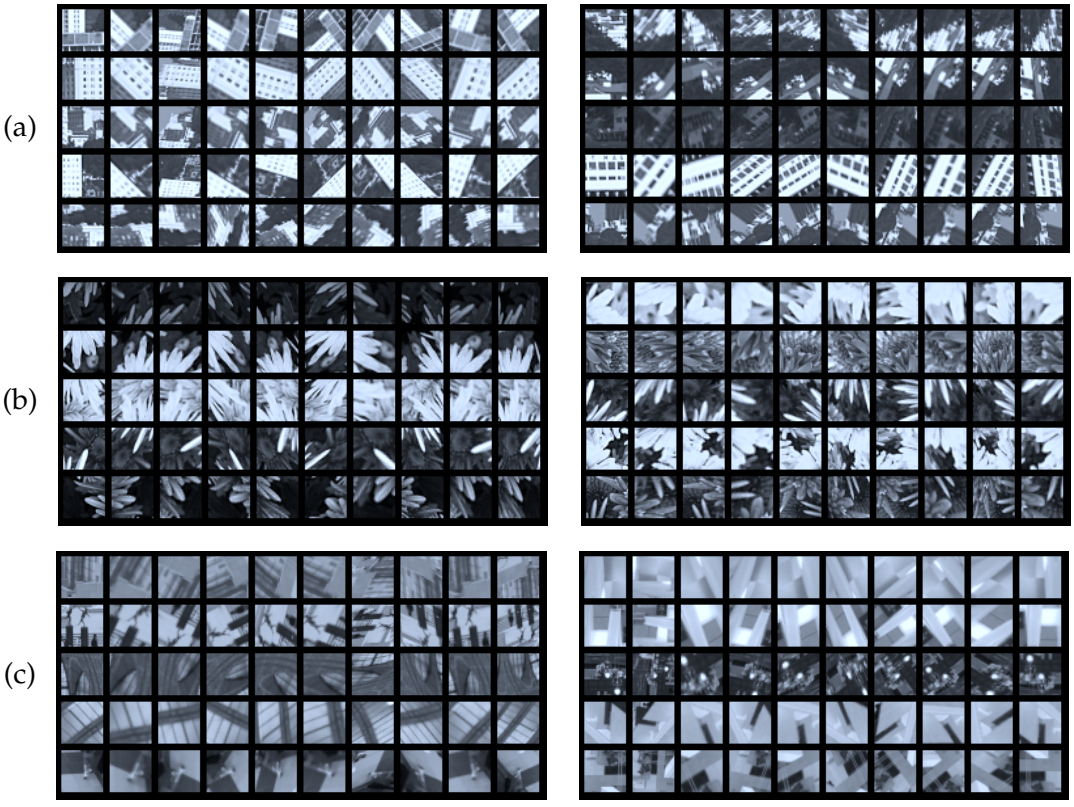


Figure 2.6: Warped patches from the images of Figure 2.5 show the range of affine deformations we considered. In each line, the left most patch is the original one and the others are deformed versions of it. **(a)** Sample patches from the *City* image. **(b)** Sample patches from the *Flowers* image. **(c)** Sample patches from the *Museum* image.

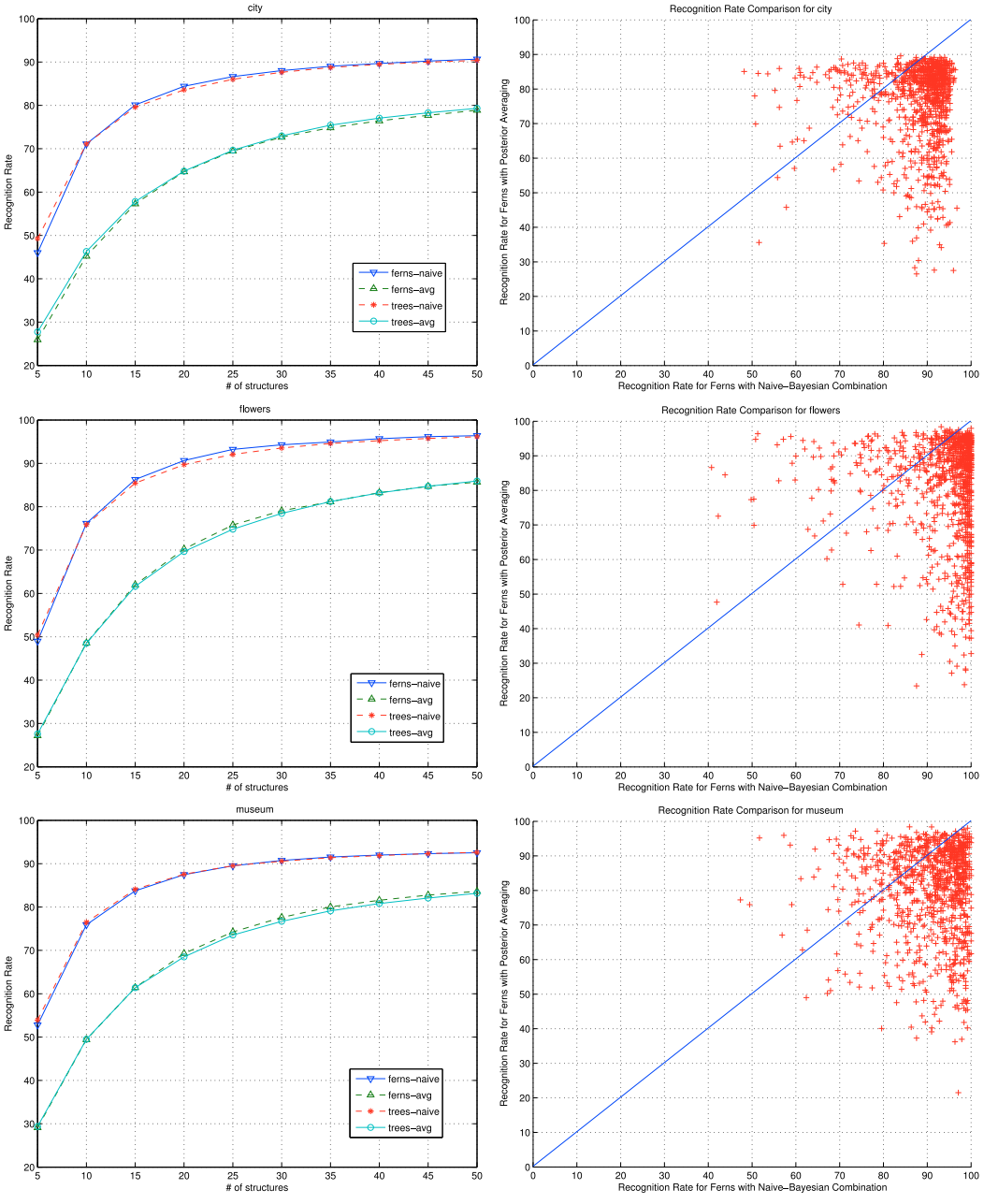
Ferns differ from trees in two important respects: The probabilities are multiplied in a Naive-Bayesian way instead of being averaged and the hierarchical structure is replaced by a flat one. To disentangle the influence of these changes, we consider four different scenarios:

- Using Randomized Trees and averaging of class posterior distributions, as in [63],
- Using Randomized Trees and combining class conditional distributions in a Naive-Bayesian way,
- Using Ferns and averaging of class posteriors,
- Using Ferns and combining class conditional distributions in a Naive-Bayesian way, as we advocate in this paper.

The trees are of depth 11 and each Fern has 11 features, yielding the same number of parameters for the estimated distributions. Also the number of features evaluated per patch is equal in all cases.

The training set is obtained by randomly deforming images of Figure 2.5. To perform these experiments, we represent affine image deformations as 2×2 matrices of the form $R_\theta R_{-\phi} \mathrm{diag}(\lambda_1, \lambda_2) R_\phi$, where $\mathrm{diag}(\lambda_1, \lambda_2)$ is a diagonal 2×2 matrix and R_γ represents a rotation of angle γ. Both to train and to test our ferns, we warped the original images using such deformations computed by randomly choosing θ and ϕ in the $[0 : 2\pi]$ range and λ_1 and λ_2 in the $[0.6 : 1.5]$ range. Figure 2.6 depicts patches surrounding individual interest points first in the original images and then in the warped ones. We used 30 random affine deformations per degree of rotation to produce 10800 images. As explained in Section 2.2.2, we then added Gaussian noise with zero mean and a large variance—25 for gray levels ranging from 0 to 255—to these warped images to increase the robustness of the resulting ferns. Gaussian smoothing with a mask of 7×7 is applied to both training and test images.

The test set is obtained by generating a separate set of 1000 images in the same affine deformation range and adding noise. Note that we simply transform the original keypoint locations, therefore we ignore the keypoint detector's repeatability in the tests and measure only the recognition performance. In Figure 2.7, we plot the results as a function of the number of trees or Ferns being used.



(a) (b)

Figure 2.7: (a) The average percentage of correctly classified image patches over many trials is shown as the number of Trees or Ferns is changed. Using the Naive Bayesian assumption gives much better rates at reduced number of structures, while the Fern and tree structures are interchangeable. **(b)** The scatter plots show the recognition rate over individual trials with 50 Ferns, given in the **x** and **y** axes, respectively for the Naive-Bayesian combination and posterior averaging. The Naive-Bayesian combination of features usually performs better, as evidenced by the fact that most points are below the diagonal, and only very rarely produce recognition rates below 80%. By contrast the averaging produces rates below 60%.

Table 2.1: Variance of the recognition rate.

Number of Structures	10	15	20	25	30	35	40	45	50
Fern-Naive	0.59	0.24	0.11	0.06	0.05	0.07	0.06	0.08	0.05
Fern-Average	0.33	0.21	0.29	0.24	0.35	0.39	0.35	0.41	0.30
Tree-Naive	0.37	0.19	0.13	0.07	0.06	0.04	0.04	0.02	0.01
Tree-Average	0.30	0.22	0.13	0.17	0.16	0.17	0.15	0.17	0.20

We first note that using either flat Fern or hierarchical tree structures does not affect the recognition rate, which was to be expected as the features are taken completely at random. By contrast the Naive-Bayesian combination strategy outperforms the averaging of posteriors and achieves a higher recognition rate even when using relatively few structures. Furthermore as the scatter plots of Figure 2.7 show, for the Naive-Bayesian combination the recognition rate on individual deformed images never falls below an acceptable rate. Since the features are taken randomly, the recognition rate changes and the variance of the recognition rate is given as Table 2.1. As more Ferns or Trees are used the variance decreases and more rapidly for the naive combination. If the number of Ferns or Trees are below 10, the recognition rate starts to change more erratically and entropy based optimization of feature selection becomes a necessity.

To test the behavior of the methods as the number of classes is increased, we have trained classifiers for matching up to 1500 classes. Figure 2.8 shows that the performance of the Naive-Bayesian combination does not degrade rapidly and scales much better than averaging posteriors. For both methods, the required amounts of memory and computation times increase linearly with the number of classes, since we assign a separate class for each keypoint.

So far, we have used 11 features for each Fern, and we now discuss the influence of this number on recognition performance, and memory requirements.

Increasing the Fern size by one doubles the number of parameters hence the memory required to store the distributions. It also implies that more training samples should be used to estimate the increased number of parameters. It has however negli-

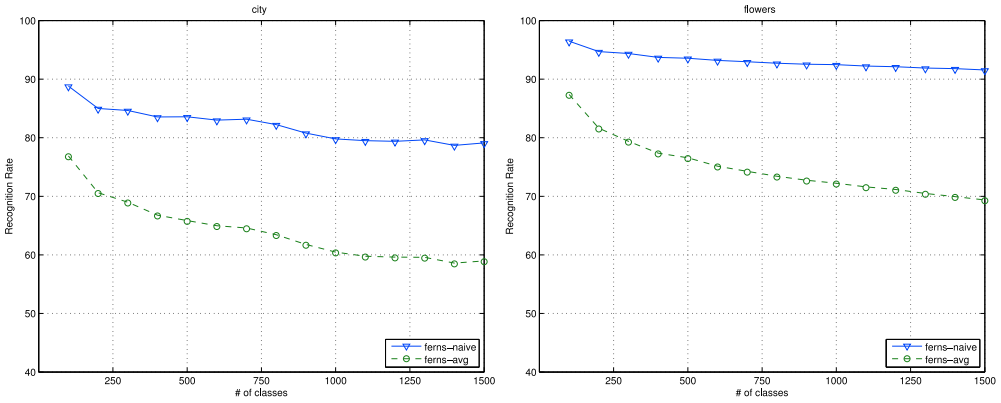
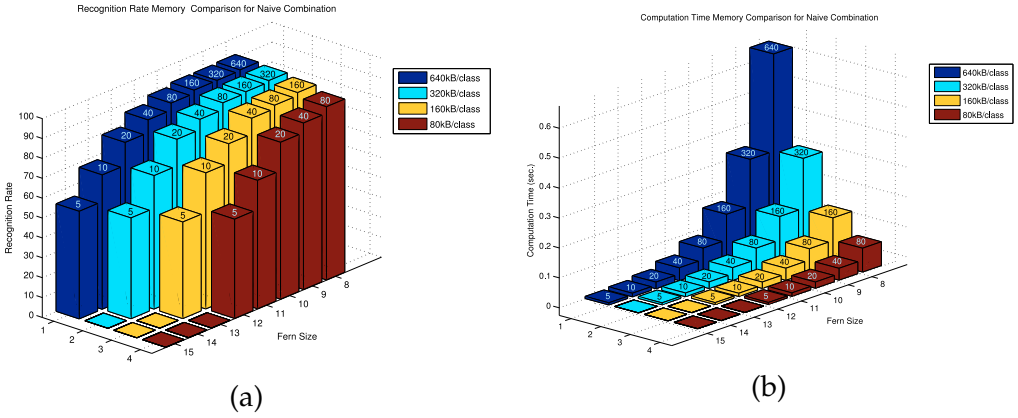


Figure 2.8: Recognition rate as a function of the number of classes. While the naive combination produces a very slow decrease in performance, posterior averaging exhibits a much sharper drop. The tests are performed on the high resolution versions of the *City* and *Flowers* data, respectively.



(a) (b)

Figure 2.9: Recognition rate **(a)** and computation time **(b)** as a function of the amount of memory available and the size of the Ferns being used. The number of ferns used is indicated on the top of each bar and the y-axis shows the Fern size. The color of the bar represents the required memory amount, which is computed for distributions stored with single precision floating numbers. Note that while using many small ferns achieves higher recognition rates, it also entails a higher computational cost.

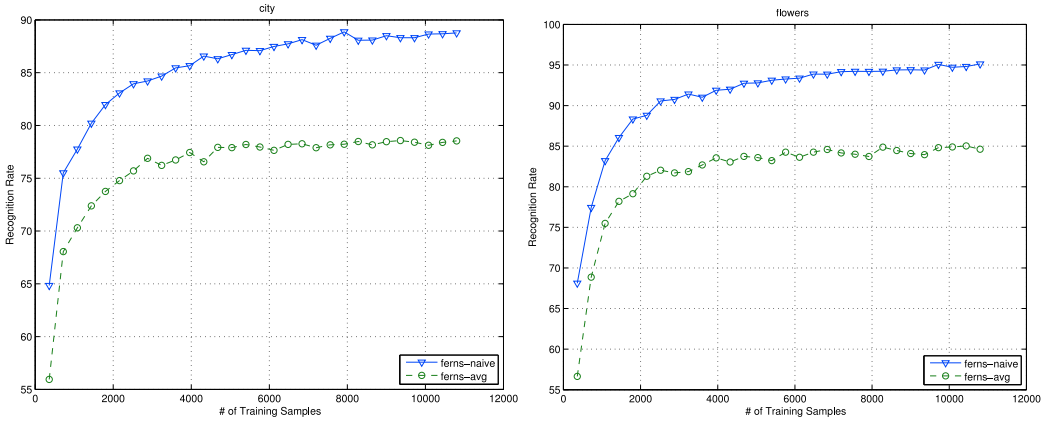


Figure 2.10: Recognition rate as a function of the number of training samples for each one of the three images of Figure 2.5. As more training samples are used the recognition rate increases and it does so faster for the Naive-Bayesian combination of Ferns.

gible effect on the run-time speed and larger Ferns can therefore handle more variation at the cost of training time and memory but without much of a slow-down.

By contrast adding more Ferns to the classifier requires only a linear increase in memory but also in computation time. Since the training samples for other Ferns can be reused it only has a negligible effect on training time. As shown in Figure 2.9, for a given amount of memory the best recognition rate is obtained by using many relatively small Ferns. However this comes at the expense of run-time speed and when sufficient memory is available, a Fern size of 11 represents a good compromise, which is why we have used this value in the experiments.

Finally we evaluate the behavior of the classifier as a function of the number of training samples. Initially, we use 180 training images that we generate by warping the images of Figure 2.5 by random scaling parameters and deformation angle ϕ, while the rotation angle θ is uniformly sampled at every two degrees. We then increase the number of training samples by 180 at each step. The graphs depicted by Figure 2.10 show that the Naive-Bayesian combination performs consistently better than the averaging, even when only a small number of training samples are used.

2.4 Results

We evaluate the performance of Fern based classification for both planar and fully three dimensional object detection. We also compare our approach against SIFT [69], which is among the most reliable descriptors for patch matching.

2.4.1 Ferns vs SIFT to detect planar objects

We used the 1074-frame video depicted by Figure 2.11 to compare Ferns against SIFT for planar object detection. It shows a mouse pad undergoing motions involving a large range of rotations, scalings, and perspective deformations against a cluttered background. We used as a reference an image in which the mouse pad is seen frontally. We match the keypoints extracted from each input image against those found in the reference image using either Ferns or SIFT and then eliminate outliers by computing a homography using RANSAC.

The SIFT keypoints and the corresponding descriptors are computed using the publicly available code kindly provided by David Lowe [70]. The keypoint detection is based on the Difference of Gaussians over several scales and for each keypoint dominant orientations are precomputed. By contrast the Ferns rely on a simpler keypoint detector that computes the maxima of Laplacian on three scales, which provides neither dominant orientation information nor a finely estimated scale. We retain the 400 strongest keypoints in the reference, and 1000 keypoints in the input images for the two methods.

We train 20 Ferns of size 14 to establish matches with the keypoints on the video frame by selecting the most probable class. In parallel, we match the SIFT descriptors for the keypoints on the reference image against the keypoints on the input image by selecting the one which has the nearest SIFT descriptor. Given the matches between the reference and input image, we use a robust estimation followed by non-linear refinement to estimate a homography. We then take all matches with re-projection error less than 10 pixels to be inliers. Figure 2.11 shows that the Ferns can match as many points as SIFT and sometimes even more.

It is difficult to perform a completely fair speed comparison between our Ferns and SIFT for several reasons. SIFT reuses intermediate data from the keypoint extraction to compute canonic scale and orientations and the descriptors, while ferns can rely on

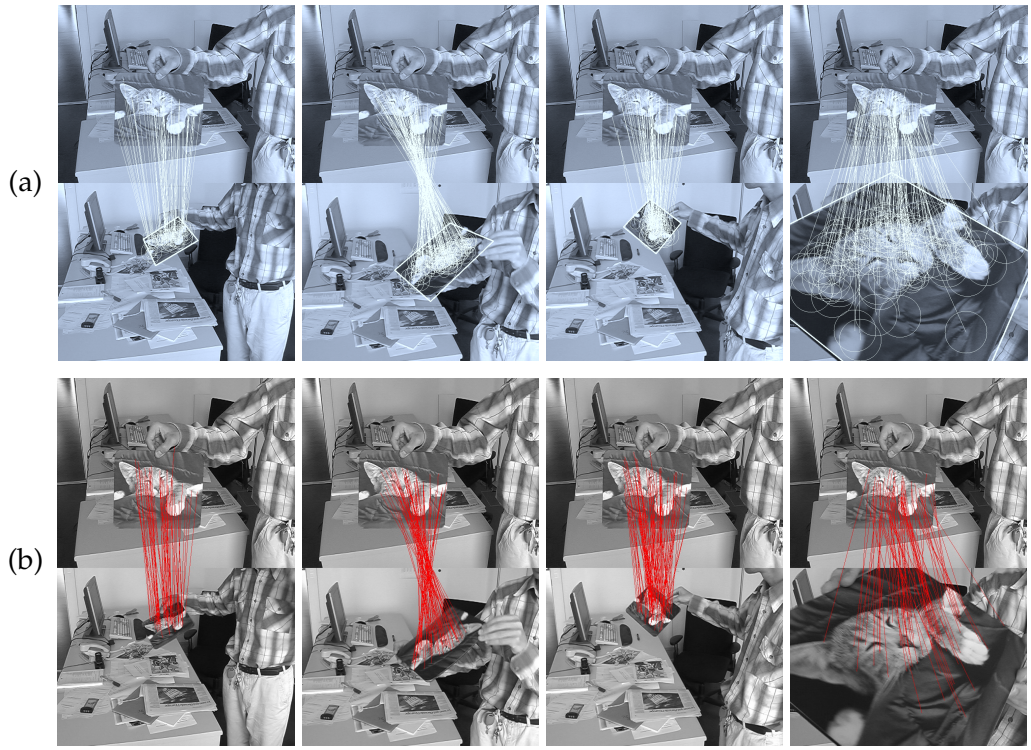


Figure 2.11: Matching a mouse pad in a 1074-frame sequence against a reference image. The reference image appears at the top and the input image from the video sequence at the bottom. **(a)** Matches obtained using ferns in a few frames. **(b)** Matches obtained using SIFT in the same frames.

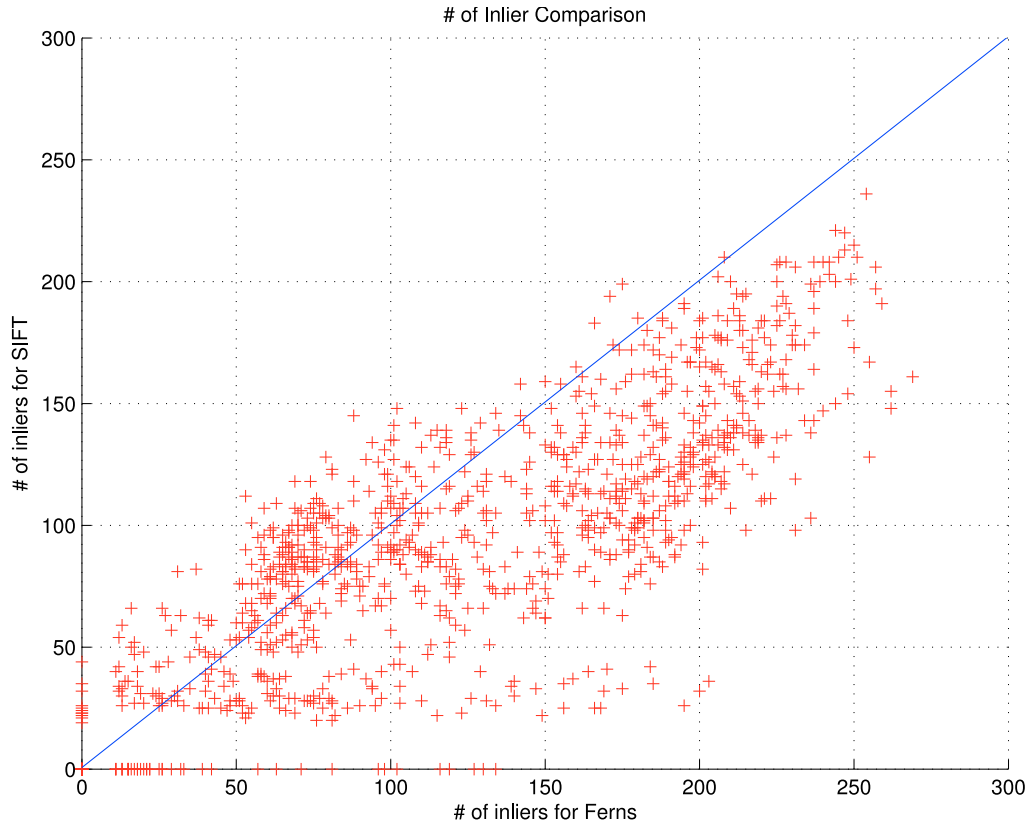


Figure 2.12: Scatter plot showing the number of inliers for each frame for the experiment in Figure 2.11. The values on the axes give the number of inliers for Ferns and SIFT. Most of the time, the Ferns match at least as many points as SIFT and often even more, as can be seen from the fact that most of the points lay below the diagonal.

a low-cost keypoint extraction. On the other hand, the distributed SIFT C code is not optimized, and the Best-Bin-First KD-tree of [8] is not used to speed up the nearest-neighbor search.

However, it is relatively easy to see that our approach requires much less computation. Performing the individual tests of Section 2.2 requires very little time and most of the time is spent computing the sums of the posterior probabilities. The classification of a keypoint requires $H \times M$ additions, with H the number of classes, and M the number of Ferns. By contrast, SIFT uses $128H$ additions and as many multiplications when the Best-Bin-First KD-tree is not used. This represents an obvious advantage of our method at run-time since M can be much less than 128 and is taken to be 20 in practice, while selecting a large number of features for each fern and using tens of thousands of training samples.

The major gain actually comes from the fact that Ferns do not require descriptors. This is significant because computing the SIFT descriptors, which is the most difficult part to optimize, takes about 1ms on a MacBook Pro laptop without including the time required to convolve the image. By contrast, Ferns take $13.5\ 10^{-3}$ milliseconds to classify one keypoint into 200 classes on the same machine. Moreover, ferns still run nicely with a primitive keypoint extractor, such as the one we used in our experiments. When 300 keypoints are extracted and matched against 200 classes, our implementation on the MacBook Pro laptop requires 20ms per frame for both keypoint extraction and recognition in 640×480 images, and four fifths of this time are devoted to keypoint extraction. This corresponds to a theoretical 50Hz frame rate if one does ignore the time required for frame acquisition. Training takes less than five minutes.

Of course, the ability to classify keypoints fast and work with a simple keypoint detector comes at the cost of requiring a training stage, which is usually off-line. By contrast, SIFT does not require training and for some applications, this is clearly an advantage. However for other applications Ferns offer greater flexibility by allowing us to precisely state the kind of invariance we require through the choice of the training samples. Ferns also let us incrementally update the classifiers as more training samples become available as we will demonstrate in Chapter 3 using the *Feature Harvesting* framework. This flexibility is key to the ability to carry out off-line computations and significantly simplify and speed-up the run-time operation.

Figure 2.13: When detecting a 3D object viewpoint change is more challenging due to self-occlusions and non-trivial lighting effects. The images are taken from a database presented in [79] and cover a total range of 70° of camera rotation. They are cropped around the object, while we used the original images in the experiments. **(a)** *Horse* dataset. **(b)** *Vase* dataset. **(c)** *Desk* dataset. **(d)** *Dog* dataset.

2.4.2 Ferns vs SIFT to detect 3D objects

So far we have considered that the keypoints lie on a planar object and evaluated the robustness of Ferns with respect to perspective effects. This simplifies training as a single view is sufficient and the known 2D geometry can be used to compute ground truth correspondences. However most objects have truly three dimensional appearance, which implies that self occlusions and complex illuminations effects have to be taken into account to correctly evaluate the performance of any keypoint matching algorithm.

Recently, an extensive comparison of different keypoint detection and matching algorithms on a large database of 3D objects has been published [79]. It was performed on images taken by a stereo camera pair of objects rotating on a turntable. Figure 2.13 shows such images spanning a 70° camera rotation range. We used this image database to evaluate the performance of Ferns for a variety of 3D objects. We compare our results against the SIFT detector/descriptor pair which has been found to perform very well on this database. The keypoints and the descriptors are computed using the same software as before [70].

As in [79], we obtained the ground truth by using purely geometric methods, which is possible because the cameras and the turn table are calibrated. The initial correspondences are obtained by using the trifocal geometry between the top/bottom cameras in the center view and every other camera as illustrated by Figure 2.14. We then reconstruct the 3D points for each such correspondence in the bottom/center camera coordinate frame and use these to form the initial tracks that span the $-35^\circ/+35^\circ$ rotation range around a central view. Since the database images are separated by 5°, the tracks span 15 images each for the top and bottom cameras. We eliminate very short tracks and remaining tracks are extended by projecting the 3D point to each image and searching for a keypoint in the vicinity of the projection. Finally to increase robustness against spurious tracks formed by outliers, we eliminate tracks covering less then 30% of the views and the remaining tracks form the ground truth for the evaluation, which is almost free of outliers. Sample ground truth data is depicted by Figure 2.15, which shows the complex variations in patch appearance induced by the 3D structure of the objects.

The training is done using views separated by 10°, skipping every other frame in the ground truth data. We then use the views we skipped for testing purposes.. This geometry based sampling is shown in Figure 2.15. Sampling based on geometry creates uneven number of training and test samples for different keypoints as there are gaps in the tracks. The sampling could have been done differently to balance the number of test and training samples for each keypoint. However our approach to sampling closely mimics what happens in practice when training data comes from sparse views and the classifier must account for unequal numbers of training samples.

We train the Ferns in virtually the same way as we do in the planar case. Each track is assigned a class number and the training images are deformed by applying random affine deformations. We then use all of them to estimate the probability distributions, as discussed in Section 2.2. 1000 random affine deformations per training image are used to train 50 Ferns of size 11. Ferns classify the test patches by selecting the track with the maximum probability. For SIFT, each test example is classified by selecting the track number of the keypoint in the training set with the nearest SIFT descriptor.

In our tests, we learn the appearance and the geometry of a 3D object from several views and then detect it in new ones by matching keypoints. Hence the learned geometry can be used to eliminate outliers while estimating the camera pose using the

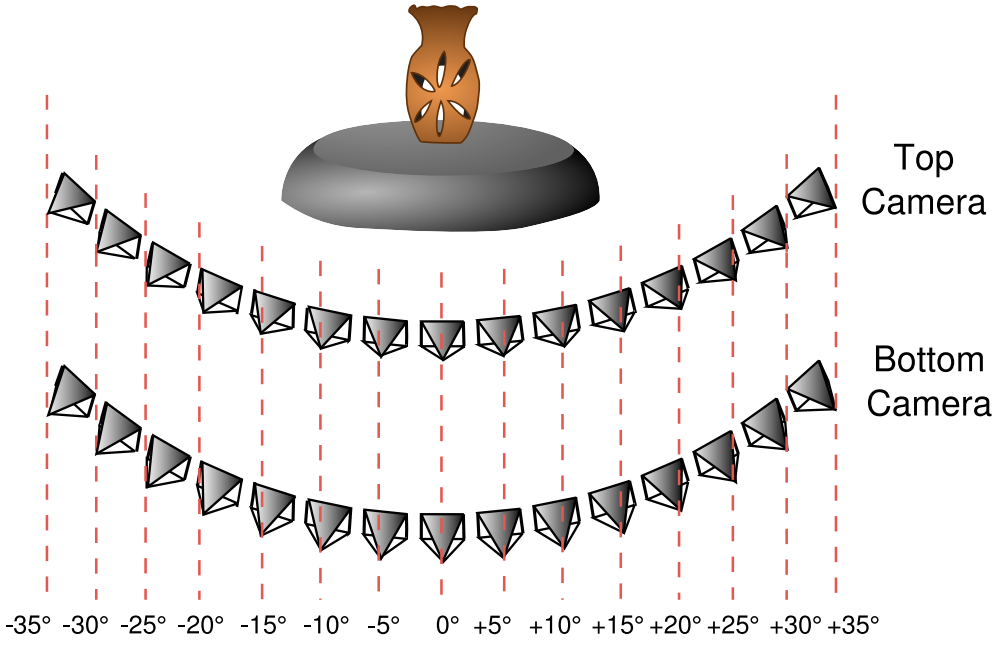


Figure 2.14: Generating ground truth data for 3D object detection. Each test object contains two sequences of images taken by the *Top* and *Bottom* cameras while the object rotates on the turntable. The camera geometry has been calibrated using a checkerboard calibration pattern. We use 15 consecutive camera views for evaluation purposes, because it is easy to obtain high quality calibration for this range of rotation.

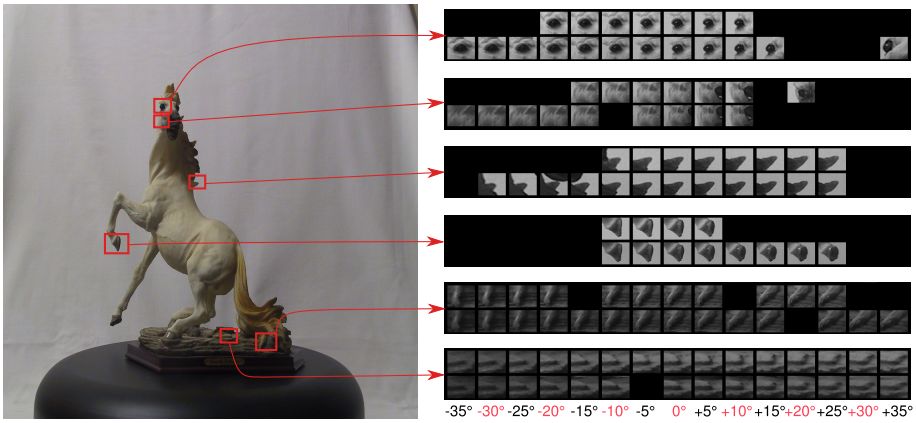


Figure 2.15: Samples from the ground truth for the *Horse* dataset. The image on the left shows the center view. The six keypoint tracks on the right show the content variation for each patch as the camera center rotates around the turntable center. Each track contains two lines that correspond to the *Top* and *Bottom* cameras, respectively. Black areas denote frames for which the keypoint detector did not respond. The views produced by a rotation that is a multiple of 10° are used for training and are denoted by red labels. The others are used for testing.

P3P algorithm [40] together with a robust matching strategy such as RANSAC [32]. Unlike [79], we therefore do not use the ratio test on descriptor distances or a similar heuristics to reject matches, as this might reject correct correspondences. The additional computational burden can easily be overcome by using the classification score for RANSAC sampling as presented by [18].

We compare the recognition rates of both methods on objects with different kinds of texture. Figure 2.16 shows the recognition rate on each test image together with the average over all frames. The Ferns perform as well as nearest neighbor matching with SIFT for a whole range of objects with very different textures.

Note that, when using Ferns, there is almost no run-time speed penalty for using multiple frames, since as more training frames are added we can increase the size of our Ferns. As discussed in Section 2.3 this requires more memory but does not slow down the algorithm in any meaningful way. By contrast, using more frames for nearest neighbor SIFT matching linearly slows down the matching, although a clever and approximate implementation might mitigate the problem.

In theory it should be possible to improve the performance of SIFT-based approach

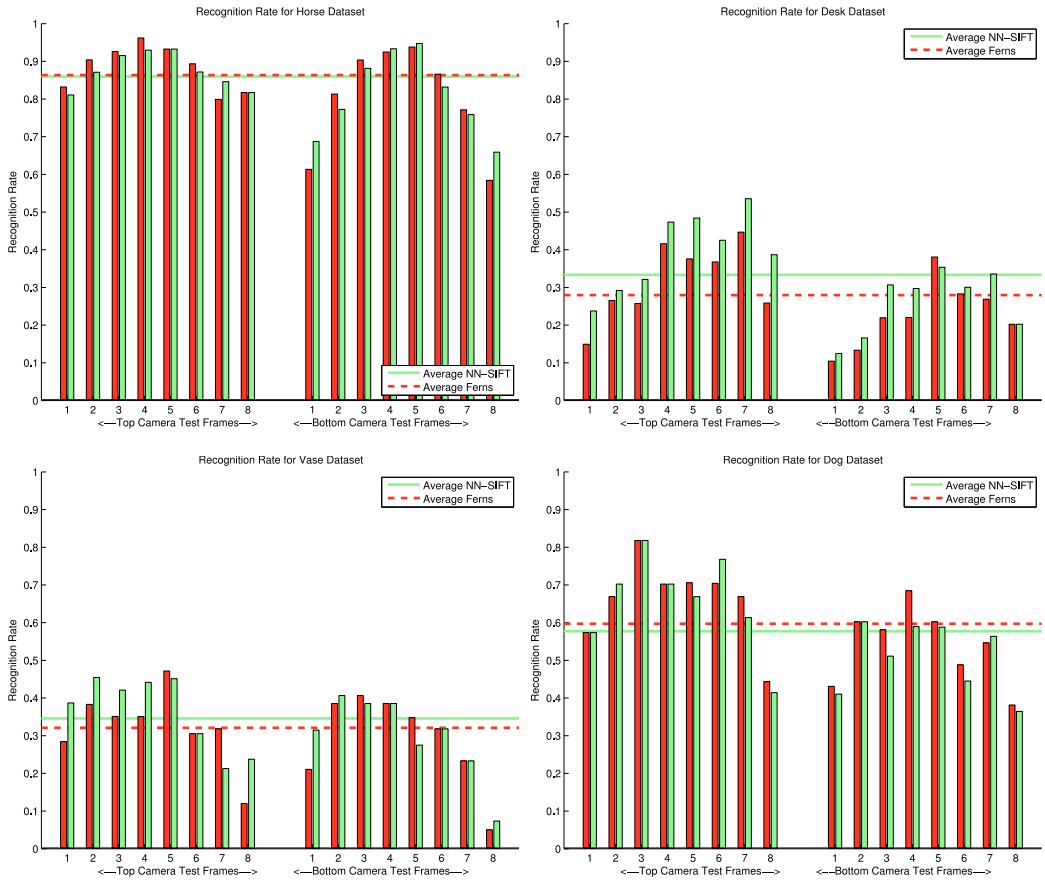


Figure 2.16: Recognition rates for 3D objects. Each pair of bars correspond to a test frame. The red bar on the left represents the rate for Ferns and the light green bar on the right the rate for Nearest Neighbor SIFT matching. The weighted averages over all frames also appear as dashed line for Ferns and solid line for NN-SIFT. The weights we use are the number of keypoints per frame.

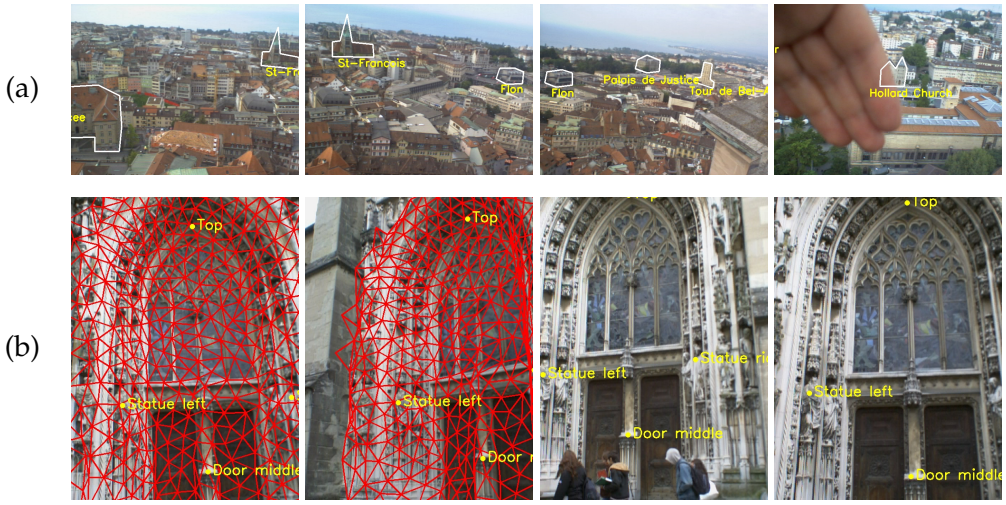


Figure 2.17: Automated image annotation. **(a)** We match an input image against a panorama. Despite occlusions, changes in weather conditions and lighting, the Ferns return enough matches for reliable annotation of major landmarks in the city. **(b)** We match an image to an annotated 3D model, overlaid on the two leftmost images. This lets us display annotations at the right place.

by replacing nearest neighbor matching with a more sophisticated technique such as K-Nearest Neighbors with voting. However, this would further slow down the algorithm. Our purpose is to show that the Fern based classifier can naturally integrate data from multiple images without the need for a more complex training phase or any handicap in the run-time performance, reaching the performance of standard SIFT matching.

2.4.3 Panorama and 3D Scene Annotation

With the recent proliferation of mobile devices with significant processing power, there has been a surge of interest in building real-world applications that can automatically annotate the photos and provide useful information about places of interest. These applications test keypoint matching algorithms to their limits because they must operate under constantly changing lighting conditions and potentially changing scene texture, both of which reduce the number of reliable keypoints. We have tested Ferns on two such applications, annotation of panorama scenes and parts of a historical building with 3D structure. Both applications run smoothly at frame rate using a standard laptop and an of the shelf web camera. By applying standard optimizations

for embedded hardware, we have ported this implementation onto a mobile device that runs at a few frames per second. More recently, Ferns have been successfully integrated into commercially available mobile phones to run at frame rates by taking into account specific limitations of the hardware and integrating detection with frame-to-frame tracking [111].

For the panorama application, we trained Ferns using an annotated panorama image stitched from multiple images. At run-time given an input image and after having established correspondences between the panorama and the test image, we compute a 2D homography and use it to eliminate outliers and to transfer the annotation from the training image to the input image as shown in Figure 2.17. We successfully run a number of tests under different weather conditions and different times of day.

Annotating a 3D object requires training using multiple images from different viewpoints, which is easy to do in our framework as discussed in the previous subsection. We also built a 3D model for the object using standard structure from motion algorithms to register the training images followed by dense reconstruction [100, 101]. The resulting fine mesh is too detailed to be used so it is approximated by a coarse one containing much less detail. Despite its rough structure, this 3D model allows annotation of important parts of the object and the correct re-projection of this information onto images taken from arbitrary viewpoints as depicted by Figure 2.17.

2.5 Conclusion

In this chapter, we have introduced a naive Bayesian framework to recognize image patches around interest points that significantly improves upon the Randomized Tree based method of [63]. We also showed that its feature extraction step can be greatly simplified without altering its performance. Using this simplified structure, we highlighted its relation to existing techniques for efficient nearest neighbor classification to provide insights into both the original method and our modified version.

Our scheme builds strongly upon the pose invariant framework of Randomized Trees. This is essential to fast run-time performance but requires us to use a very large training set. As noted in Section 2.2.3, this means that recovering even an approximate nearest neighbor is very costly and we are forced to estimate only its class rather than its location. Therefore we can not use a distance ratio test to simply reject outliers [69].

One way to overcome this problem is to model the classifier response for each class using a Gaussian distribution. We can then label as outliers the keypoints with classification scores falling outside a predetermined range. This has been found to decrease detection time for real-world applications [1]. [63] also uses a similar strategy to threshold the response of Randomized Trees.

The memory requirements of Ferns can be greatly reduced by quantizing the distributions into less than 32 bits that is used by single-precision floating point representations since this does not degrade the run-time performance much. For example fixed point representations can profitably replace the floating point arithmetic we use in our implementation. Our scheme does not use much more memory than Nearest Neighbor classifiers using local descriptors at least in the context of most Augmented Reality applications [111]. Of course the descriptors themselves do not take up much space. But the data structures used in approximate nearest neighbor retrieval achieve their significant speed-up by exploiting space-time tradeoffs and as a result require large amounts of memory. However, the memory access pattern of our naive Bayesian scheme can be much more random than descriptor based methods and it is less suited to existing memory architectures of mobile devices.

Currently, our scheme can easily deal with hundreds of keypoints which is sufficient in most Augmented Reality applications. Since its memory requirements and run-time increases linearly with the number of keypoints, it does not scale to tens of thousands of keypoints required by image retrieval systems. This is a consequence of assigning an individual class for every single keypoint. This is inefficient and a hierarchical classification step is required to scale sub-linearly with the number of keypoints [84].

[1] Personal communication with Gerhard Reitmayr who suggested the use of a Gaussian classifier response model and kindly shared his results with us in the context of his work in [111].

CHAPTER

THREE

WEAKLY SUPERVISED LEARNING BY FEATURE HARVESTING

In many 3D object-detection and pose estimation problems ranging from Augmented Reality to Visual Servoing, run-time performance is of critical importance. For such applications, using only a recursive tracker is not enough since it requires initialization, it can drift and most importantly it can not recover from eventual tracking failure. Therefore an object detector needs to be combined with the tracker to solve these issues. Unfortunately, object detection usually takes much more time than simple tracking and can easily become a performance bottle-neck.

However, there usually is time to train the object detector before actually using it and in this way run-time detection speed can be increased substantially. [64] exploits this fact by using a textured 3D computer graphics model to generate synthetic images of the object. These are used to train a Randomized Tree based classifier that can perform wide-baseline matching between the keypoints on the image and the 3D features on the target object. This classifier can then be used to achieve robust real-time 3D object detection. In this section, we show that this approach extends naturally to the case where no *a priori* textured 3D model is available, thus removing one of the major limitations of the original method and yielding the behavior depicted by Fig. 3.1.

The key ingredient of our approach is a weakly supervised training algorithm that we refer to as *feature harvesting*: Assuming that we can first observe the target object moving slowly, we define an ellipsoid that roughly projects at the object's location in the first frame. We extract feature points inside this projection and use the image

patches surrounding them to train a first classifier, which is then used to match these initial features in the following frames. As more and more new frames become available, we discard features that cannot be reliably found and add new ones to account for aspect changes. We use new views of the features we retain to refine the classifier and, each time we add or remove a feature, we update it accordingly. Once all the training frames have been processed, we run a bundle-adjustment algorithm on the tracked feature points to also refine the model's geometry.

In short, starting from the simple ellipsoid shown in the top row of Fig. 3.1, we robustly learn both geometry and appearance. An alternative approach to initializing the process would have been to use a fully automated on-line Simultaneous Localization and Mapping (SLAM) algorithm [22]. We chose the ellipsoid both for simplicity's sake—successfully implementing a SLAM method is far from trivial—and because it has proved to be sufficient, at least for objects that can be enclosed by one. In case this is inappropriate, the recent Parallel Tracking and Mapping (PTAM) algorithm [59] can be used to build the 3D model during the training phase.

Note that the geometric construction and registration of the object model for feature harvesting employs the same approach that is typically used to perform SLAM. The feature harvesting extends these geometric concepts to model the image appearance of interest points. The appearance model is build from scratch and it is refined in a probabilistic manner as more frames are observed. The interest points included in the model are filtered to keep the best performing ones, not only in terms of repeatability but also according to the reliability of recognition.

The originality of our approach is to use exactly the same tracking and statistical classification techniques, first, to train the system and automatically select the most stable features and, second, to detect them at run-time and compute the pose. In other words, the features we harvest are those that can be effectively tracked by the specific wide-baseline matching algorithm we use. This contrasts with standard classification-based approaches in which classifiers are built beforehand, using a training set manually labeled and that may or may not be optimal for the task at hand. Hence our approach is similar in spirit to more recent on-line trackers based on Adaboost [36] that can be updated in a semi-supervised manner [4, 43] and the *P-N Learning* approach of [57] that trains and updates a set of Randomized Trees from unlabeled data.

In contrast to these more recent approaches that classify image patches directly as object or background, we use interest points as features and the 3D geometry to verify the detection. Integration of the 3D model allows us to recover 3D camera pose and provides robustness against changes in scale and aspect ratio without an explicit search over these parameters. As a result, our system is very easy to train by simply showing it the object slowly moving and, once trained, both very fast and very robust to a wide range of motions and aspect changes, which may cause complex variations of feature appearances.

In the next section, we present the relevant literature on object detection and tracking either by a sliding window or feature point based approach. In Section 3.2, we briefly review fast wide baseline matching for interest points using Randomized Trees and Ferns. We also show that they can both be updated on-line as more training samples are obtained. We present our *Feature Harvesting* framework in Section 3.3, and show its effectiveness using several image sequences in Section 3.4. Finally, we compare feature harvesting performance of Randomized Tree and to that of Ferns in Section 3.5.

3.1 Related Work

There are two dominant approaches to tracking-by-detection. Sliding window approaches use a binary object/background classifier and search for the object in the image exhaustively. Feature point based approaches use interest points to hypothesize potential object locations and verify object presence using geometric cues. We briefly review both approaches and discuss the performance trade-offs involved.

Sliding Window Approaches. Sliding window approaches [4, 43, 57, 117] train a binary classifier to distinguish the object from the background and do not require a geometrical model for verification. They exhaustively check object presence over all possible image locations and scales. Usually a classifier cascade is used to speed up this search to achieve real-time performance [34, 110].

Since the background will change during run-time the classifier needs to be updated on-line using the detected object locations on new frames. However since the classifier may not precisely locate the object, these updates can cause drift. This is

Figure 3.1: Our approach to 3D object detection applied to a toy car, a face, and a glass. In each one of the three cases, we show two rows of pictures. The first represents the training sequence, while the second depicts detection results in individual frames that are not part of the training sequence. We overlay the ellipsoid we use as our initial model on the images of the first row. The only required manual intervention is to position it in the very first image. To visualize the results, we attach a 3D referential to the center of gravity of the ellipsoid and use the estimated 3D pose to project it into the images. Note that, once trained, our system can handle large aspect, scale, and lighting changes. It can deal with the transparent glass as well as with the hand substantially occluding the car. And when a complete occlusion occurs, such as when the book completely hides the face, it simply returns no answer and recovers when the target object becomes visible again.

avoided using several strategies. [43] uses a semi-supervised approach that combines the human annotated training samples and the most recent set of detections. The classifier is allowed to capture variations in recent frames, but it can not drift too far away from the initial model. [4] trains the classifier using Multiple Instance Learning, which requires the classifier to label groups of patches as containing the object or not. The updates enforce that the classifier assigns a positive label to a bag of image patches collected around the most recent detection sites. Hence, individual patches do not need to be labelled. [57] introduces the *P-N Learning* framework that slowly updates the classifier with high confidence training samples that are generated by two processes as follows: Stable tracking generates the positive samples and ambiguous detections far from the tracked object form the negative sample set. They also allow other types of processes as long as the confidence in the labels is high enough.

Sliding window approaches can handle highly non-rigid objects and do not require strong texture cues to be present. However, lack of 3D information about the object forces these methods to explore the aspect ratio and scale ranges exhaustively, which is costly and less robust. Furthermore the camera pose is not recovered during object localization since the features used by the classifier do not provide the necessary spatial constraints. As a result, they are more suited for object detection and tracking in the 2D image plane and when the recovery of the 3D object pose is not required.

Feature Point Based Approaches. Feature point based approaches to object detection and pose estimation [59, 62, 73, 94, 99, 111] use matches between interest points detected on the image and 3D object features to localize the object. The presence of the object is usually verified using geometric constraints. Such geometric information includes the camera rotation and translation in case a 3D object model is available [64], a homography in case the object is planar [111], or the epipolar relation to a set of keyframes [89]. They are relatively insensitive to partial occlusions, cluttered backgrounds, and scale/aspect changes and they can simultaneously recover both the 2D object location and 3D camera pose.

The feature points used at detection time are often designed to be affine invariant [76]. Once they have been extracted, various local descriptors have been proposed to match them across images. Among these, SIFT [69] has been shown to be one of the most effective [75]. In [76], it is applied to rectified affine invariant regions to achieve

perspective invariance. In [87], a similar result is obtained by training the system using multiple views of a target object, storing all the SIFT features from these views, and matching against all of them. However, computing such descriptors can be costly. Furthermore, matching is usually performed by nearest-neighbor search, which tends to be computationally expensive if the number of training views used to build the detector is large, even when using an efficient data structure [8].

Another weakness of these descriptors is that they are predefined and do not adapt to the specific images under consideration. [73] addresses this issue by building the set of the image neighborhoods of features tracked over a sequence. Kernel PCA is then performed on this set to compute a descriptor for each feature. This approach, however, remains computationally expensive.

By contrast, [64] uses textured model of the target object to synthesize images from different viewpoints to train a set of Randomized Trees that can recognize the keypoints detected on these training views. This is effective because it allows the system to learn potentially complex appearance changes. However, it requires building a textured 3D model. This can be cumbersome if the object is either complex or made of a non-Lambertian material that makes the creation of an accurate texture-map nontrivial. If one is willing to invest the effort, it can of course be done but it is time consuming. The approach we introduce in this section completely does away with this requirement.

3.2 3D Wide Baseline Matching using Randomized Trees and Ferns

Let us consider a set of 3D object features $\{\mathbf{M}_i\}$ that lie on the target object and let us assume that we have collected a number of image patches $p_{i,j}$ centered on the projections of $\mathbf{M}_i$ into image j, for all available i and j. The $\{p_{i,j}\}$ constitute the training set we use to train the classifier $\hat{\mathscr{R}}$ to predict to which $\mathbf{M}_i$, if any, a given image patch p corresponds, in other words, to approximate as well as possible the actual classification rule $\mathscr{R}(p) = i$. At run-time, $\hat{\mathscr{R}}$ can then be used to recognize the object features by considering the image patch p around a detected image feature. Given the 3D position of the $\mathbf{M}_i$, this establishes 2D–3D correspondences that can be used to compute 3D pose.

In principle any kind of classifier could have been used. Randomized Trees, however, are particularly well adapted because they naturally handle multi-class problems, while being both robust and fast. We first review wide baseline matching using Randomized Trees and introduce a mechanism to perform online updates as new training samples become available. In Section 3.2.2, we show that Random Ferns can also be updated similarly.

As discussed in Section 1.2.2, multiple trees are grown so that each one yields a different partition of the space of image patches. The tree leaves contain an estimate of the posterior distribution over the classes, which is learned from training data. A patch p is classified by dropping it down each tree and performing an elementary test at each node, which sends it to one side or the other, and considering the sum of the probabilities stored in the leaves it reaches. We write

$$\hat{\mathscr{R}}(p) = \underset{i}{\text{argmax}} \sum_{T \in \mathcal{T}} \hat{P}_{L(T,p)}(\mathscr{R}(p) = i)\,, \tag{3.1}$$

where i is a label, the $\hat{P}_{L(T,p)}(\mathscr{R}(p) = i)$ are the posterior probabilities stored in the leaf $L(T,p)$ of tree T reached by p, and $\mathcal{T}$ is the set of Randomized Trees. Such probabilities are estimated during training as the ratio of the number n_i^L of patches of class i in the training set that reach L and the total number n_i of patches of class i that is used in the training. This yields

$$\hat{P}_L(\mathscr{R}(p) = i) \simeq \frac{n_i^L / n_i}{S_L}\,, \tag{3.2}$$

where $S_L = \sum_j \frac{n_j^L}{n_j}$ is a normalization term that enforces $\sum_i \hat{P}_L(\mathscr{R}(p) = i) = 1$. We normalize by the number of patches because the real prior on the class is expected to be uniform, while this is not true in our training population. Although any kind of test could be performed at the nodes, simple binary tests based on the difference of intensities of two pixels of Equation (2.4) have proved sufficient. In practice, classifying a patch involves only a few hundreds of intensity comparisons and additions per patch, and is therefore very fast.

3.2.1 Randomized Trees and On-line Training

The approach described above assumes that the complete training set is available from the beginning, which is not true in our case as object features may be added or re-

moved while the classifier is being trained. Here we show how to overcome this limitation by exploiting the random nature of the tree building algorithm.

As discussed before, in [64], the node tests are chosen so as to minimize leaf entropy, which is estimated according to the training set. Without the complete training set, this cannot be meaningfully done. Instead, we build the tree by randomly selecting the tests, that is to say the $\mathbf{d}_{j,1}$ and $\mathbf{d}_{j,2}$ locations of Equation (2.4). The training data is only used to evaluate the $\hat{P}_L$ posterior probabilities in the leaves of these randomly generated trees. Surprisingly, this much simplified procedure, which is going to allow us to iteratively estimate the $\hat{P}_L$ values, results in virtually no loss of classification performance [63]. Interestingly, a similar result has also been reported in the context of 2D object recognition [71].

We also introduce a mechanism for updating the tree when new views of an existing object feature are introduced or when an object feature is either added or removed, which the RT approach lets us do as follows.

- **Incorporating New Views of Object Features.** Recall that, during the initial training phase, patches are dropped down the tree and the number of patches reaching leaf L is plugged into Equation (3.2) to derive $\hat{P}_L$ for each class at leaf L. Given a new view, we want to use it to refine these probability estimates. To this end, we invert the previous step and compute the number of patches reaching leaf L as

$$n_i^L = \hat{P}_L(\mathscr{R}(p) = i) \times n_i \times S_L. \tag{3.3}$$

  This only requires storing the normalization terms S_L at each leaf L and keeping the n_i counters for each class. We then use newly detected patches to increment n_i^L and n_i. When all the new patches have been processed, we again use Equation (3.2) to obtain the refined values of $\hat{P}_L$. Note that we do not store the image patches themselves, which could cost a lot of memory for long training sequences.

- **Adding and Removing Object Features.** The flexible procedure outlined above can also be used to add, remove or replace the classes corresponding to specific object features during training. Removing class i and the corresponding object feature merely requires setting

$$n_i^L = n_i = 0. \tag{3.4}$$

We can then replace the i^{th} feature by a new one by simply changing the $\mathbf{M}_i$ 3D coordinates introduced at the beginning of Section 3.2 to be those of the new object feature and using patches centered around the new projections of $\mathbf{M}_i$ to estimate $\hat{P}_L$.

These update mechanisms are the basic tools we use to recursively estimate the RTs while harvesting features, as discussed in Section 3.3.

3.2.2 Random Ferns and On-line Training

Random Ferns can be updated in a similar manner by writing the probabilities of Equation (2.10) computed during Ferns training as

$$P(F = k \,|\, \mathscr{R}(p) = i) \simeq \frac{n_{k,i} + N_r}{\sum_{k=1}^{K}(n_{k,i} + N_r)} \tag{3.5}$$

$$= \frac{n_{k,i} + N_r}{n_i + K \times N_r}, \tag{3.6}$$

where Fern F takes on one of K values, N_r is the regularization prior, $n_{k,i}$ is the number of training samples from class i with Fern value k, and n_i is the total number of training samples from class i. Note that this is similar to Equation (3.2) except the lack of the normalization term S_L and the addition of the prior N_r.

Given a training set $\mathfrak{T}^T = \bigcup_{t=1}^{T} \mathfrak{T}_t$ that contains T subsets, and denoting the number of training samples from subset t and class i that evaluates to Fern value k as $n_{k,i}^t$, the training estimates the posterior distribution of feature values for each class,

$$P(F = k \,|\, \mathscr{R}(p) = i, \mathfrak{T}^T) \simeq \frac{N_r + \sum_{t=1}^{T} n_{k,i}^t}{\sum_{k=1}^{K}\left(N_r + \sum_{t=1}^{T} n_{k,i}^t\right)} \tag{3.7}$$

$$= \frac{N_r + \sum_{t=1}^{T} n_{k,i}^t}{K \times N_r + \sum_{t=1}^{T}\left(\sum_{k=1}^{K} n_{k,i}^t\right)} \tag{3.8}$$

$$= \frac{N_r + \sum_{t=1}^{T} n_{k,i}^t}{K \times N_r + \sum_{t=1}^{T} n_i^t} \tag{3.9}$$

$$= \frac{n_{k,i}^T + \left(N_r + \sum_{t=1}^{T-1} n_{k,i}^t\right)}{n_i^T + \left(K \times N_r + \sum_{t=1}^{T-1} n_i^t\right)} \tag{3.10}$$

$$= \frac{n_{k,i}^T + N_{k,i}^{T-1}}{n_i^T + D_i^{T-1}}. \tag{3.11}$$

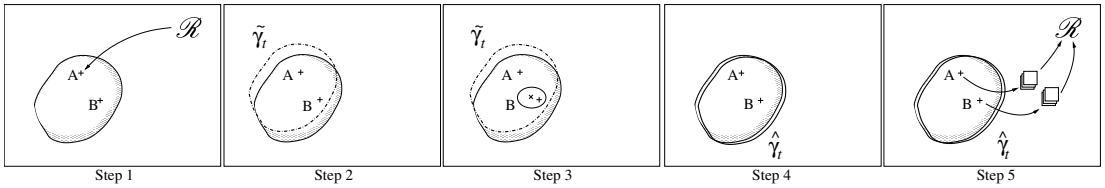


Figure 3.2: The five steps of feature harvesting introduced at the beginning of Section 3.3.1.

Since the feature probability estimated using the first $T-1$ training subsets is

$$P(F = k \,|\, \mathscr{R}(p) = i, \mathfrak{T}^{T-1}) \simeq \frac{N_{k,i}^{T-1}}{D_i^{T-1}}, \tag{3.12}$$

storing the denominator terms D_i^{T-1}, one for each class, is enough to be able to update the probabilities. In fact for Ferns this can be done more efficiently than for Trees since there is no need to store and update the normalization terms S_L.

Such incremental training assumes that the classifier trained using partial data will be effective in object detection from viewpoints that are not in the range used to obtain the training set. Of course we do not expect the classifier to generalize to wildly varying viewpoints but to those that are close to at least some training examples. In Section 3.5, we compare the generalization ability of Trees and Ferns to such test cases and assess their suitability for *Feature Harvesting*.

3.3 From Harvesting to Detection

In this section, we show that standard frame-to-frame tracking and independent 3D detection in each individual frame can be formalized similarly and, therefore, combined seamlessly as opportunity dictates. This combination is what we refer to as tracking-by-detection. The originality of our approach is to use exactly the same image feature recognition technique at all stages of the process, first, to train the system and automatically select the most stable features and, second, to detect them at runtime.

We first give an overview of our method. We then explain how the tracking is performed *without* updating the classifier, and conclude with the complete "feature harvesting" framework.

3.3.1 Overview

As shown in the top row of Figure 3.1, to initialize the training process, we position the ellipsoid that we use as an initial 3D model so that it projects on the target object in the first frame. We then extract a number of image features from this first image and back-project them to the ellipsoid, thus creating an initial set of the $\{\mathbf{M}_i\}$ object features of Section 3.2. By affine warping lightly the image patches surrounding the image features, we create the $p_{i,j}$ image patches that let us instantiate a first set of randomized trees.

During training, new features detected on the object are integrated into the classifier. Because the number of such features can become prohibitively large when dealing with long training sequences, it is desirable to keep the ones that are successfully detected and recognized by the classifier most often, and remove the other ones. More precisely, given the set of trees trained using the first frame or more generally all frames up to frame $t-1$, we handle frame t using the five-step feature-harvesting procedure described below and illustrated by Figure 3.2:

1. We extract image features from frame t and use the classifier to match them, which, in general, will only be successful for a subset of these features.

2. We derive a first estimate $\widetilde{\gamma}_t$ of the camera pose from these correspondences using a robust estimator that lets us reject erroneous correspondences.

3. We use $\widetilde{\gamma}_t$ to project unmatched image features from frame $t-1$ into frame t and match them by looking for the image features closest to their projections.

4. Using these additional correspondences, we derive a refined estimate $\widehat{\gamma}_t$.

5. We use small affine warping of the patches around image features matched in frame t to update the classifier as discussed in Section 3.2.1. Features that have not been recognized often are removed to be replaced by new ones.

At run-time, we use the exact same procedure, with one single change: We stop updating the classifier, which simply amounts to skipping the fifth step.

3.3.2 3D Tracking by Detection

Let us first assume that the classifier $\mathscr{R}$ has already been trained. Both tracking and detection can then be formalized as the estimation of the camera pose Γ_t from image features extracted from all previous images that we denote $I_{s\leq t}$. In other words, we seek to estimate the conditional density $P(\Gamma_t \,|\, I_{s\leq t})$.

A camera motion model —appearing as the term $P(\Gamma_t \,|\, \Gamma_{t-1})$ in the following derivations— should be chosen. It often assumes either constant velocity or constant acceleration. This is fine to regularize the recovered motion but can also lead to complete failure. This tends to occur after an abrupt motion or if Γ_{t-1} is incorrectly estimated, for example due to a complete occlusion. Γ_t can then have any value no matter what the estimate of Γ_{t-1} is. In such a case, we should consider the density of Γ_t as uniform and write $P(\Gamma_t \,|\, \Gamma_{t-1}) \propto \lambda$, which amounts to treating each frame completely independently. In our implementation, we use a mixture of these two approaches and take the distribution to be

$$P(\Gamma_t \,|\, \Gamma_{t-1}) \propto m(\Gamma_{t-1}, \Gamma_t) = \exp\Big(-\left(\Gamma_t - \Gamma_{t-1}\right)^\top \mathbf{\Sigma}^{-1} \left(\Gamma_t - \Gamma_{t-1}\right)\Big) + \lambda\,. \qquad (3.13)$$

This lets us both enforce temporal consistency constraints and to recover from tracking failures by relying on single-frame detection results. In our implementation, the respective values of $\mathbf{\Sigma}$ and λ were chosen manually.

Unfortunately, introducing the term λ process precludes the use of standard particle filtering techniques. Our camera pose space has six dimensions, and the required number of particles, which grows exponentially with the number of dimensions, would be too large to make particle filters tractable. Therefore we have to restrict ourself to the estimation of the mode $\widehat{\gamma_t}$ of this density:

$$\widehat{\gamma_t} = \operatorname*{argmax}_{\gamma} P(\Gamma_t = \gamma \,|\, I_{s\leq t})\,,$$

in which the expression of $P(\Gamma_t = \gamma \,|\, I_{s\leq t})$ can be found using the standard Bayesian tracking relation:

$$P(\Gamma_t = \gamma \,|\, I_{s\leq t}) \propto P(I_t \,|\, \Gamma_t = \gamma) P(\Gamma_t = \gamma \,|\, \Gamma_{t-1} = \widehat{\gamma_{t-1}}) P(\Gamma_{t-1} = \widehat{\gamma_{t-1}} \,|\, I_{s<t})\,. \qquad (3.14)$$

Once we obtain the set $\widetilde{n_t}$ of correspondences using the classifier, we apply a RANSAC based approach to remove false matches and to derive a first estimate $\widetilde{\gamma_t}$

for the camera pose. A new set $\widehat{n}_t$ is then made of the inliers of $\widetilde{n}_t$, and completed by projecting the unmatched object features with $\widetilde{\gamma}_t$ and matched each of them with the closest image feature. A numerical optimization is then performed to find $\widehat{\gamma}_t$ by minimizing the log-likelihood of $m(\widehat{\gamma_{t-1}}, \gamma)P(I_t \,|\, \Gamma_t = \gamma)$:

$$\widehat{\gamma}_t = \underset{\gamma}{\operatorname{argmin}} \sum_{n \in \widehat{n}_t} \|\mathbf{P}(\gamma)\mathbf{M}(n) - \mathbf{m}(n)\|^2 + \rho\left(\left(\gamma - \widehat{\gamma_{t-1}}\right)^\top \mathbf{\Sigma}^{-1} \left(\gamma - \widehat{\gamma_{t-1}}\right)\right) \tag{3.15}$$

where $\mathbf{P}(\gamma)$ is the projection matrix for the camera pose γ, ρ is the Tukey robust estimator that approximates the logarithm of (3.13), and $\mathbf{M}(n)$ and $\mathbf{m}(n)$ are respectively the object feature and the image feature for correspondence n.

The advantage of the classifier is that there is no need for the previous pose. However, this procedure can result in some jittering on the estimated pose over a sequence. To enforce temporal consistency and reduce the effect, when $\widehat{\gamma_{t-1}}$ is valid, we also consider transient object features which projections can be matched across I_{t-1} and I_t using standard cross-correlation. Their 3D positions can be estimated from the rough model and $\widehat{\gamma_{t-1}}$ by back-projection. According to our experience, over two consecutive frames, this position is accurate enough to improve the recovered displacement. These additional correspondences are integrated in Equation (3.15) for pose estimation exactly in the same way as the correspondences established with the classifier. Note that these correspondences are not required by our method, but they are useful to reduce the jittering effect.

3.3.3 Feature Harvesting

During training we use the same process but now the classifier is not initially available and we want to create it incrementally by "feature harvesting." This implies keeping or discarding object features such as those shown in Figure 3.3. Let us first denote by r_t^* the best classifier obtained with the images $I_{s \leq t}$ and the feature correspondences computed using the poses $\gamma_{s \leq t}$:

$$r_t^* = \underset{r}{\operatorname{argmax}}\, P(\mathscr{R} = r \,|\, \Gamma_{s \leq t} = \gamma_{s \leq t},\, I_{s \leq t})\,. \tag{3.16}$$

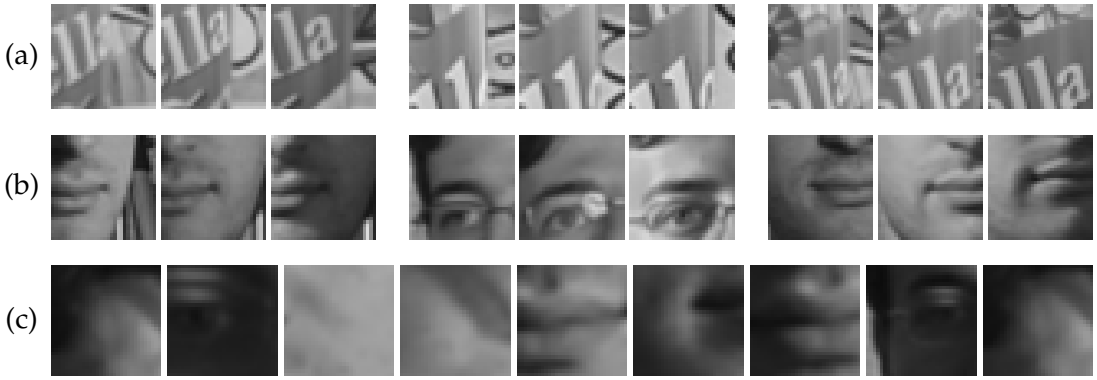


Figure 3.3: The harvest. **(a)** Three sample patches for three distinct features on the glass. Note that the foreground is relatively constant while the background changes drastically. **(b)** Three sample patches for three distinct face features, obtained under changing light and orientation. **(c)** Patches corresponding to object features found to be unreliable and discarded during training.

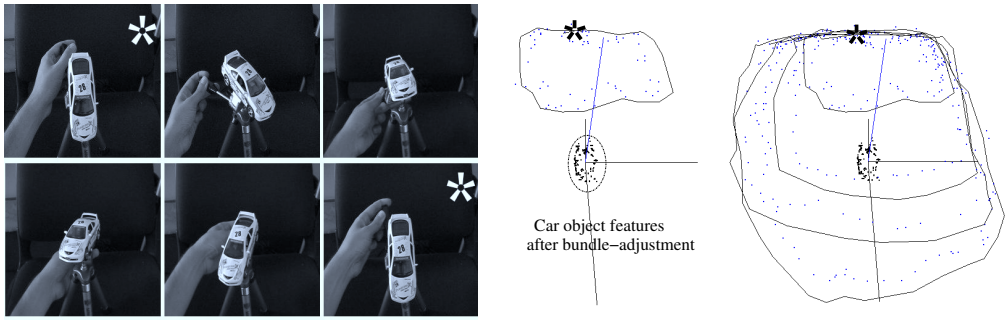
Here we show that r_t^* can be used to compute $\widehat{\gamma_{t+1}}$ under reasonable assumptions. We have:

$$\begin{aligned} & P(\Gamma_{s\leq t} = \gamma_{s\leq t}, I_{s\leq t}) \\ = & \sum_r P(\Gamma_{s\leq t} = \gamma_{s\leq t}, I_{s\leq t}, \mathscr{R} = r) = \sum_r P(\Gamma_t = \gamma_t, I_t, \Gamma_{s<t} = \gamma_{s<t}, I_{s<t}, \mathscr{R} = r) \\ = & \sum_r P(\Gamma_t = \gamma_t, I_t \,|\, \Gamma_{s<t} = \gamma_{s<t}, I_{s<t}, \mathscr{R} = r) P(\mathscr{R} = r \,|\, \Gamma_{s<t} = \gamma_{s<t}, I_{s<t}) \times \\ & P(\Gamma_{s<t} = \gamma_{s<t}, I_{s<t}) \,. \end{aligned}$$

All the classifiers have a negligible probability $P(\mathscr{R} = r \,|\, P_{s<t} = \gamma_{s<t},\, I_{s<t})$ except for those concentrated around $r = r_{t-1}^*$. Otherwise, that would mean that other classifiers than r_{t-1}^* constructed with $\gamma_{s<t}$, $I_{s<t}$ would be as good as r_{t-1}^*, which is not realistic since r_{t-1}^* has been built from these poses and images. Let us continue the derivation:

$$\begin{aligned} \simeq & \; P(\Gamma_t = \gamma_t,\, I_t \,|\, \Gamma_{s<t} = \gamma_{s<t},\, I_{s<t},\, \mathscr{R} = r_{t-1}^*) P(\Gamma_{s<t} = \gamma_{s<t},\, I_{s<t}) \\ = & \; P(I_t \,|\, \Gamma_t = \gamma_t,\, \Gamma_{s<t} = \gamma_{s<t},\, I_{s<t},\, \mathscr{R} = r_{t-1}^*) \times \\ & \; P(\Gamma_t = \gamma_t \,|\, \Gamma_{s<t} = \gamma_{s<t},\, I_{s<t},\, \mathscr{R} = r_{t-1}^*) P(\Gamma_{s<t} = \gamma_{s<t},\, I_{s<t}) \\ \simeq & \; P(I_t \,|\, \Gamma_t = \gamma_t,\, \mathscr{R} = r_{t-1}^*) P(\Gamma_t = \gamma_t \,|\, \Gamma_{s<t} = \gamma_{s<t}) P(\Gamma_{s<t} = \gamma_{s<t},\, I_{s<t}) \end{aligned}$$

because the incoming image does not depend on the poses except on the current one, and the current pose does not depend on the previous images neither on the classifier,



(a) (b) (c)

Figure 3.4: **(a)** Sample frames from a training sequence where the toy car is fixed to a tripod and rotated four times. The frames marked with a star show the reference position which is reached in all four loops. **(b)** Recovered relative camera motion with respect to the toy car after the first loop. The trajectory is shown in the referential of the ellipsoid. The dots represent the trajectory before bundle-adjustment, the plain curve after. **(c)** Camera motion for all four loops. As can be seen, there is no drift. Note that all four loops go through the star.

which is reasonable. By applying the Bayes' theorem on the terms $P(\Gamma_{s\leq t} = \gamma_{s\leq t}, I_{s\leq t})$ and $P(\Gamma_{s<t} = \gamma_{s<t},\ I_{s<t})$, we get:

$$P(\Gamma_{s\leq t} = \gamma_{s\leq t} \,|\, I_{s\leq t}) \simeq$$
$$\tfrac{P(I_{s<t})}{P(I_{s\leq t})} P(I_t \,|\, P_t = \gamma_t, \mathscr{R} = r^*_{t-1}) P(\Gamma_t = \gamma_t \,|\, \Gamma_{t-1 s<t} = \gamma_{s<t}) P(\Gamma_{s<t} = \gamma_{s<t} \,|\, I_{s<t})$$

And under standard probabilistic tracking hypotheses, we finally obtain:

$$P(\Gamma_t = \gamma_t \,|\, I_{s\leq t}) \propto$$
$$P(I_t \,|\, P_t = \gamma_t, \mathscr{R} = r^*_{t-1}) P(\Gamma_t = \gamma_t \,|\, \Gamma_{t-1} = \gamma_t) P(\Gamma_{t-1} = \gamma_{t-1} \,|\, I_{s<t})$$

which is the same expression as Equation (3.14) used for tracking, except that the classifier r^*_{t-1} appears in the observation model. That means that the same method as in Section 3.3.2 can be used to estimate $\widehat{\gamma}_t$. Once this pose is found, r^*_{t-1} is updated using correspondences between object features and image features to give r^*_t as explained in Section 3.2.1.

To validate this training procedure, we performed the experiment depicted by Figure 3.4, which clearly shows that the recovered camera trajectory does not drift.

3.4 Results

In this section we demonstrate the effectiveness and generality of our approach using three very different objects, a toy car, a face, and a partially-textured transparent glass. In all three cases, we follow the same procedure: We show the system the training sequence depicted by the top rows of Figure 3.1, which is used to harvest features as discussed in Section 3.3.3. When all the training frames have been processed, we freeze the set of RTs we have built and proceed with the tracking-by-detection approach of Section 3.3.2. Our non-optimized implementation runs at 5Hz during tracking, and 1Hz during training. About 20% of the time is devoted to extracting and recognizing the features, and the remaining 80% by the pose estimation procedure. This could be considerably sped-up by using more efficient strategies [18].

Figures 3.5, 3.6, and 3.7 show a number of frames extracted from test sequences of several hundreds frames—the toy sequence is made of about 1500 frames—in which our target objects translate and rotate. Because the object is re-detected in every frame, the algorithm is robust to abrupt motion and complete occlusion. For example, after the third frame of Figure 3.5, the car falls on the ground and has to be picked up. As soon as it becomes visible again, the system re-acquires it. The same happens in the example of Figure 3.6 after the subject hides his face behind the book. These examples highlight some of the strengths of our algorithm:

- **Robustness to cluttered background.** Once trained, the classifier is feature-specific enough so that it does not get confused by cluttered background as shown in Figure 3.5.

- **Insensitivity to scale changes.** Thanks to the multi-scale interest point detector, the algorithm can handle a very broad range of scales, including scales that were not part of the training sequence. As shown in several of the examples of Figures 3.5 and 3.6, the system keeps on successfully detecting even though the target object moves both much closer and much further.

- **Robustness to complex illumination effects.** In the case of the face, we deliberately changed the lighting when acquiring the training sequence of Figure 3.1 to build lighting invariance into the classifier. As can be seen in the bottom rows of Figure 3.6, this was successful and gives the system robustness to very

Figure 3.5: Detecting the car in a sequence that involves abrupt motions, large scale and lighting changes, and very substantial occlusions. To visualize the results, we attach a 3D referential to the center of gravity of the initial ellipsoid and use the estimated 3D pose to project it into the images. We also overlay the projections of the harvested object feature points. The toy car is successfully detected in all frames except those where it is almost entirely occluded. And, because the object is re-detected in every frame, the system easily recovers after such a failure.

Figure 3.6: Face results. Note that by contrast with previous face detection approaches, the face pose can be retrieved under **(a)** large rotations, **(b)** scale and lighting changes, and **(c)** different facial expressions. **(d)** After the occlusion by the book, the algorithm automatically recovers.

Figure 3.7: Detecting a transparent object with partial texture. The squares in the first three images outline the patches around the features detected at three different scales in a test frame. The straight line segments connect the feature with the corresponding one in a frame of the training sequence. Since during training the system learned which parts of the patches are meaningful as shown in Figure 3.3, the image features can be recognized even if the patch overlaps the background or the transparent parts. As shown in the fourth frame, the glass is successfully detected.

marked lighting changes. While it was not necessary for the toy car because it has a simple shape, it experimentally appeared that a training sequence with such variations greatly improve the results.

- **Handling transparencies.** Finally, we can also handle the partially-textured transparent glass of Figure 3.6 by using a suitable training sequence with a complex background. It lets the classifier learn that the parts of the patches surrounding feature points that overlap the transparent parts or the background are not relevant for classification purposes. Our algorithm can automatically reject feature points on transparent parts. At run-time features can thus be successfully recognized even if the background has changed.

3.5 Comparison of Feature Harvesting using Trees and Ferns

The feature harvesting relies on two important assumptions: the classifier trained using partial data will be effective in the detection of the object from new viewpoints and it can be updated on-line without the necessity of remembering the initial training set. This latter requirement ensures the efficiency of updates and the scalability to long training videos. In Section 3.2, we have already shown that both Randomized Trees and Ferns satisfy this requirement by virtue of their probabilistic representations.

So far we have not quantified the efficiency of the initial classifier. We have performed another set of experiments to measure how well it generalizes to wider range of viewpoints than used to generate the training set. We use the images of Section 2.3 in these tests as follows. In the first test we apply a random set of 2D perspective deformations to the images of Figure 2.5 and we limit the in-plane rotations to ± 5 degrees. We use 150 training samples generated in this way to train the initial classifier. The test set is obtained in the same way but it covers a wider range of ± 30 degrees of in-plane rotation with the addition of image noise. We measure the recognition rate on the test set and repeat this experiment each time with an increased range of in-plane rotations during training. We plot the results in Figure 3.8 that show Ferns are affected less by the limited range of training data than Randomized Trees. To understand this better we have included the prior term into the posterior distributions used by Randomized Trees. This resulted in a boost of their performance despite yielding lower performance than Ferns. This can be explained by noting that the prior term

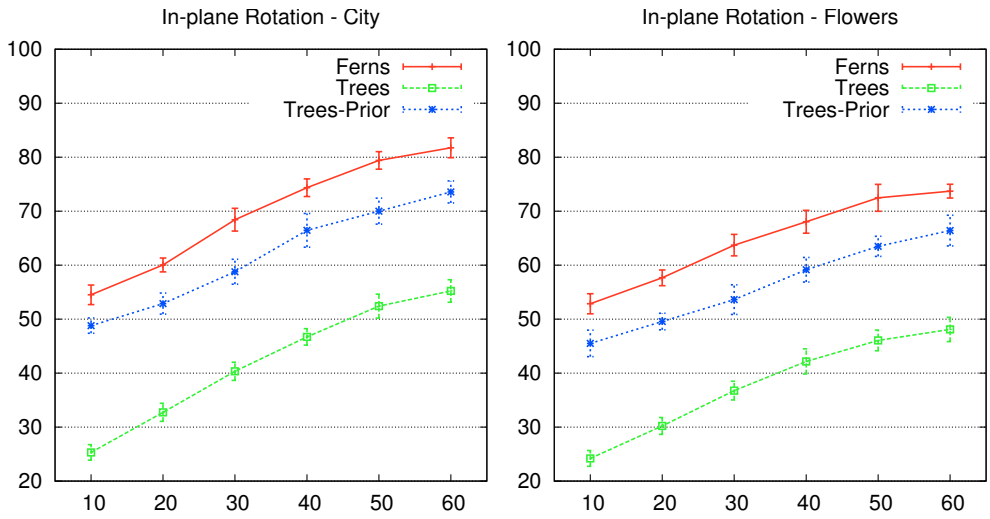


Figure 3.8: Recognition rate of Ferns and Trees as the range of in-plane rotations present in training data varies. The range of rotations used to generate the test data is ± 30 degrees in each case. As the training range is decreased recognition rate decreases but less so for Ferns than Trees. This robustness is due to the prior term in Equation (2.10) since adding a similar term also boosts the performance of Trees. **(a)** *City image*. **(b)** *Flowers* image.

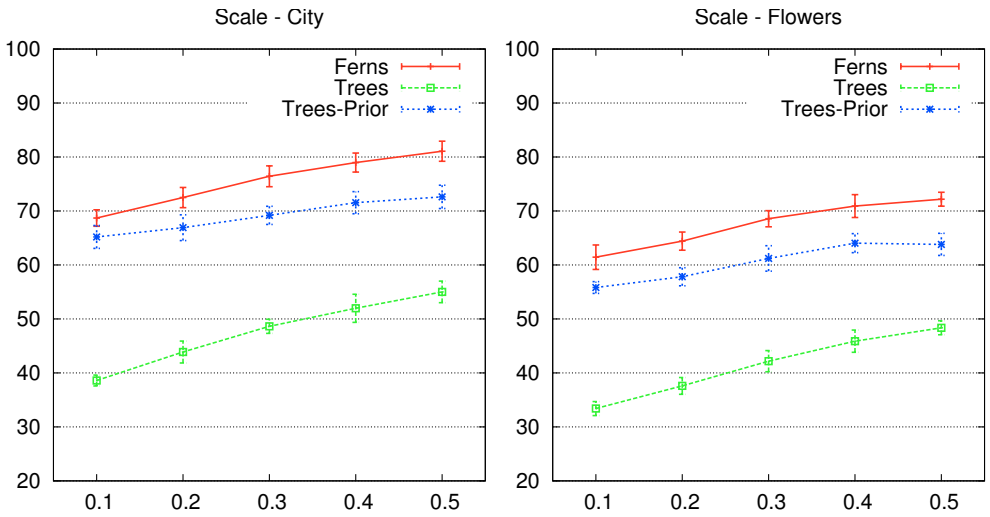


Figure 3.9: Recognition rate of Ferns and Trees as the range of scale changes during training is varied. Starting from ± 0.1 octave and increasing till ± 0.5 octave, which is the range used to generate test data, we observe a steady increase in recognition performance. Note that this set of fine scale changes does not affect the performance as much as in-plane rotations of Figure 3.8 and the Ferns still yield higher recognition rates. **(a)** *City image*. **(b)** *Flowers* image.

reduces the effect of leaves that receive a small number of samples during training and increases the weight of the more confident ones in the final classification. Without this regularization the naive Bayesian combination does not work at all. Our experiments show that it also increases the generalization of the additive combination when a small number of training samples are used.

Figure 3.9 depicts the results of a repetition of the above experiment, but this time using a limited range of scale changes during training. The test set covers one octave of the scale space and the training range is slowly increased from ± 0.1 to ± 0.5 octaves. Our conclusions above for in-plane rotations apply equally well to the case of scale changes.

3.6 Conclusion

In this chapter, we have shown that it is possible to train Randomized Trees and Ferns in a weakly supervised manner and the 3D rigid geometric constraints is effective to

remove the false matches generated by the classifiers. As a result, we can remove the stringent requirement of a textured 3D model and use ordinary image sequences during training with minimal supervision for the first frame. This significantly improves both the quality and the simplicity of training for 3D objects.

Considering the increased training data available in the form of video sequences, weaker supervision is an essential step towards faster and better object tracking-by-detection. Recent developments in this area show very promising results [4, 43, 57]. However, there are still barriers to successful application of these ideas. Perhaps the most important of these is the determination of object boundaries under 3D perspective changes. In a fully supervised setting the object is delineated by a human supervisor with correct boundaries or features.

To completely remove such supervision requires identifying the features that become visible during training as belonging to either the background or the object. This is equivalent to solving a segmentation problem in each frame that will be used to update the classifier. One possible research direction is to guide this segmentation task using category-level boundary and feature information that is collected from a database of objects of the same category as the tracked object. Indeed, the 3D ellipsoid and cylinder we have used in training exactly represent such category-level geometric prior information. Despite their effectiveness, these models are not very flexible and require careful initialization at the beginning of training. They are also very rigid and purely geometric. Replacing these with a statistical prior on both appearance and shape would increase both the quality and ease of training and consequently the performance at run-time.

CHAPTER

FOUR

POSE COVARIANT OBJECT DETECTION

In Chapter 3, we have shown that using 3D geometric constraints, a single object can be detected in images without performing an explicit search over scale and aspect ranges. In contrast, most state-of-the-art approaches to detecting and localizing objects of a particular category rely on searching over all possible image windows and on using a pose invariant classifier to decide whether or not the object is present in individual windows. This raises two difficult issues: First, for most objects, the bounding box aspect ratio and size can vary significantly, thus forcing the algorithm to explore a whole range of location and size parameters. Second, to achieve good localization performance, the classifier must be able to reject windows that only partially overlap with the object while at the same time being insensitive to object pose.

The first problem severely increases the computational burden of these approaches. The second is potentially even more serious because good performance rests on two conflicting demands: Good localization requires sensitivity to errors in bounding box location while robustness to viewpoint changes requires insensitivity to the changing feature statistics. As a result, even though standard histogram-based approaches offer some measure of pose invariance, their localization performance is often poor.

In this chapter, we will argue that both problems are due to the pose invariant nature of the classifier. Specifically, this classifier is used both to reject background patches and to estimate object dimensions and location by filtering out bounding boxes with incorrect dimensions or misplaced boxes of the correct size. These are in fact two separate problems and it is hard to efficiently solve for both, especially

independent of object pose, by using a single monolithic classification step. Instead, we propose a three step approach that is pose covariant. The first two steps concern estimating the correct bounding box dimensions and object pose, while the final step performs the object versus background classification. The information recovered in the first two steps guides and greatly simplifies the final classification.

To reliably estimate object bounding box and pose, we need to model their effect on the image features. Unfortunately, the rigid 3D geometric constraints that we have utilized to perform tracking-by-detection can not be obtained for an object category due to within class variance. However, object instances of the same category often share significant image characteristics such as edge orientations and feature locations that can be used to constrain the pose and scale at which the object is imaged. We will show that given appropriate training data, standard machine learning techniques can be used to train viewpoint and scale classifiers to extract such geometric information. Note that these estimators work on image patches regardless of whether they contain an object or not. In case of a background image patch, their output is almost random. In fact, they are trained with only pose and size annotated positive training samples. As a result they are not confused by complex background feature statistics and can focus on selecting only features that model viewpoint and scale variations. This greatly simplifies their training. Indeed, in Section 4.2 we show that even a simple naive Bayes classifier is sufficient when coupled with a powerful feature selection method such as Conditional Mutual Information Maximization (CMIM) [33].

Once we obtain such classifiers for object scale and pose, we can verify the existence of an object in the estimated area of interest using a classifier that is trained with only images of objects with similar viewpoints to the estimated one. The positive training sets for these view-tuned classifiers contain much less feature variation compared to that of a pose invariant approach. Therefore the feature statistics are much easier to model and they can be more efficiently separated from background clutter. Note that this is similar in spirit to the one used in keypoint descriptors such as SIFT [69] to achieve scale and rotation covariant keypoint recognition. SIFT uses maxima of Laplacian and the gradient orientation histogram to estimate the scale and rotation of the keypoints to be matched, both of which can be directly computed from image features. Our scale and pose estimators for object categories replace this direct

computation with a probabilistic one. Furthermore, as in the case of keypoint descriptors, we do not require these estimates to be perfect. Approximate values are sufficient because we rely on histogram based feature representations that are largely invariant to small changes in bounding box size and view angle.

Overall, we propose a layered approach to covariant object detection that greatly increases localization performance. First, we train an estimator for the bounding box dimensions, which then allows us to run our classifier only on windows with the estimated size instead of looping through ranges of different sizes. We then achieve view invariance by training a second estimator to return the viewpoint under which the object was imaged, which allows us to use a classifier trained for that viewpoint. Section 4.2 presents the details of each step.

To be able to train the pose and scale estimators and to quantify their performance, we introduce a database of images acquired at a car show. They were taken as the cars were rotating on a platform and cover the whole 360 degree range with a sample every 3 to 4 degrees. There are around 2000 images in the database belonging to 20 very different car models and Figure 4.1 depicts some sample frames together with detection and pose estimation results. In Section 4.3, using the first 10 sequences for training purposes and the rest for testing purposes, we show that our approach results in improvement in both detection and localization performance.

4.1 Related Work

Object detection and localization from multiple views has recently gained more attention with the adoption of more challenging datasets containing images of objects seen from arbitrary views [10, 19, 28, 29, 38]. To handle the increased variance in the object appearance and to close the gap between classification and localization, recent approaches either integrate stronger part location statistics [2, 28, 35] or rely on more complex classification machinery [10, 82]. However these approaches do not handle the 3D nature of the problem and rely on the classifier to discover an invariant representation using a training set that contains different objects of the same category seen from disparate views.

An alternative approach is to directly model the 3D viewpoint [65, 104, 118, 119]. Recently, [91] showed that it is possible to learn a 3D part based representation that

explicitly includes the viewpoint, which is also recovered as part of the detection process. [102] improved this approach by automatically learning a set of key frames that are registered on a view sphere and extracting the part based model using the geometric constraints imposed by the set of key frames. Although we share the same goal, our multi-step approach is more flexible since we have a much simpler statistical model and do not require rigid 3D constraints. Note that we use a manually annotated training set in our experiments, however we could recover the pose information using 3D constraints as [102] does and use it as input to the classifiers. Furthermore our classifiers can be used in conjunction with any existing method for object classification, which can be used to perform the final step. We demonstrate that decoupling the multi-view aspect of the problem from object classification yields better object localization.

Improved localization performance also depends on rejecting windows that only partially overlap with the object. [10] addresses this problem by training an object detector that learns a mapping from input features to the output label and bounding box. However this approach is dependent on the ability to compute a bound on the classification score for rectangle sets, which is a restrictive assumption. By contrast, we learn a separate mapping for bounding box estimation hence do not need to impose constraints on the form of the classification score.

4.2 Three-Step Object Localization

We present an object localization framework inspired by viewpoint invariant interest point descriptors and show that it leads to improved object localization. Our framework involves three steps. The first two assume that an object is present in the vicinity of the test location and estimate the bounding box size and object pose under this assumption. The final step confirms the existence of an object within the estimated bounding box, which is done using a single classifier tuned to the estimated viewpoint.

We first introduce the joint feature space for all three steps and give the details of our approach to viewpoint and bounding box estimation.

Figure 4.1: Sample detections from the test set. The green rectangle depicts the recovered bounding box and the estimated viewpoint is indicated inside the green circle at the top-right corner. A front facing car is indicated by a downward pointing line. Despite the challenging lightning conditions and changing backgrounds our approach can correctly localize cars and estimate their pose.

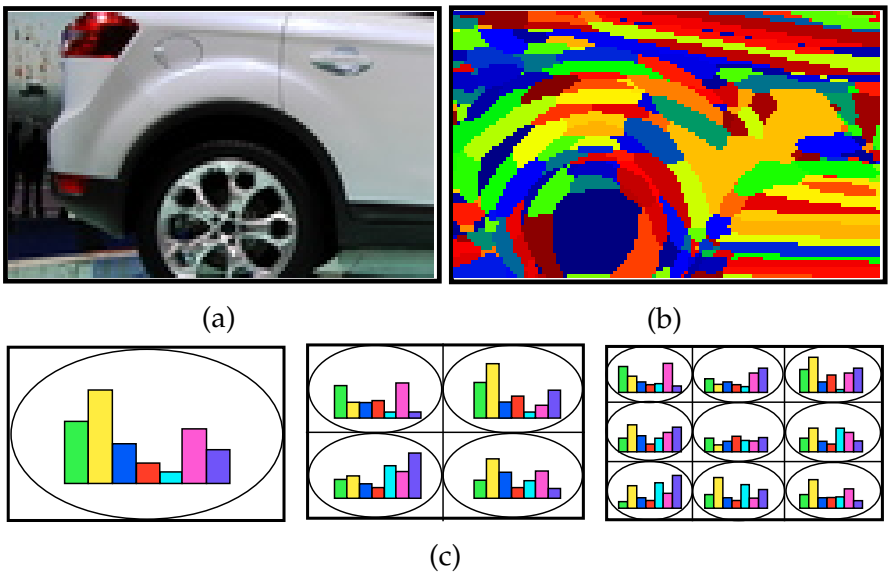


Figure 4.2: Image features. **(a)** Original image. **(b)** Cluster label map. A descriptor is computed at every point and assigned a cluster number. **(c)** Histogram pyramid. The cluster map is divided into increasingly finer regions and for each region a histogram of cluster numbers is built. The contribution of each pixel is weighted by a Gaussian kernel to achieve invariance to small translations.

4.2.1 Image Features

Given a bounding-box that defines an image window, we describe it in terms of histogram-based features, which have become the norm in object detection due to their ability to handle large intra-class variation and to provide robustness against errors in bounding-box size and location. In practice, to create these features, we first compute at every pixel a SIFT-like descriptor that has recently been introduced and is designed for dense computation [105]. We then assign to each pixel a cluster number to create label maps such as the one of Figure 4.2(b). The clusters centers are estimated in the training phase using K-Means. Finally, we create a spatial pyramid of histograms [60] that represents the label frequencies in smaller and smaller regions. We provide the precise parameters we used in Section 4.3.

Given these features, the final step of our algorithm is to train Support Vector Machines (SVMs) to decide whether or not a specific object is present within the bounding-box. We show below that these features are also effective for bounding box size and pose estimation.

4.2.2 Viewpoint Estimation

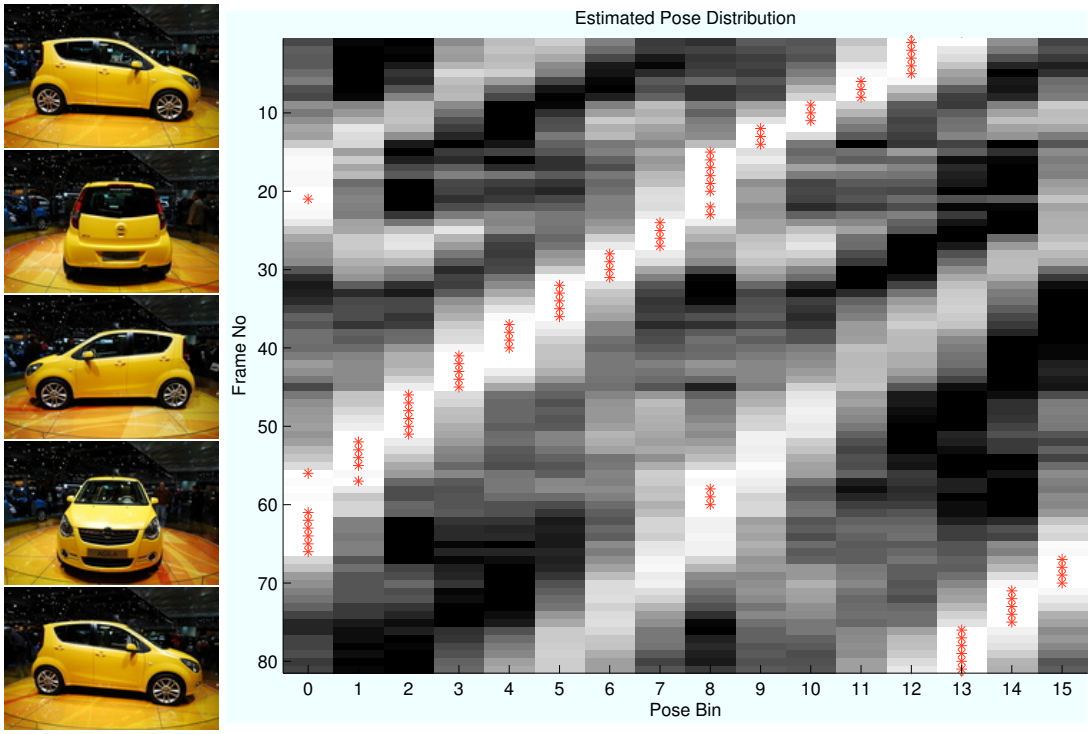
We model the viewpoint by a single angle representing the rotation parallel to the ground plane as it is the dominant factor as far as feature statistics are concerned. It is quantized into 16 pose bins. The i^{th} bin is denoted by $\mathcal{P}_i$ and $\mathcal{P}_0$ represents a front facing object. We assume that the cars in our database rotate at constant angular velocity and recover its value by using the time of capture of a full rotation. Using this information we compute the rotation angle for each image with respect to the front facing reference pose, to be used in the training and also as ground truth for testing.

We then use a Naive Bayes classifier to learn the mapping from spatial pyramid histograms to the probability of each pose bin,

$$P(\mathcal{P}_i \,|\, \mathcal{H}), \tag{4.1}$$

where $\mathcal{H}$ represents the spatial pyramid histograms computed in the given bounding box. It is obtained by concatenating the histograms from all regions inside the bounding box,

$$\mathcal{H} = [\mathcal{H}^1, \mathcal{H}^2, \mathcal{H}^3, \ \cdots, \mathcal{H}^{N_k}], \tag{4.2}$$



(a) (b)

Figure 4.3: Pose estimation. **(a)** From top to bottom, frames 1, 20, 40, 60 and 81 of an image sequence from the test set depicting a slowly rotating car.**(b)** Estimated pose distributions for all frames of the sequence. The red stars indicate the pose bin with maximum probability. The estimated pose values is mostly in sync with the motion of the car. Note the ambiguity of the estimated pose between the front and back facing object pose bins $\mathcal{P}_0$ and $\mathcal{P}_8$, which is the sole source of wrong estimates that are off the diagonal that represents the true car motion.

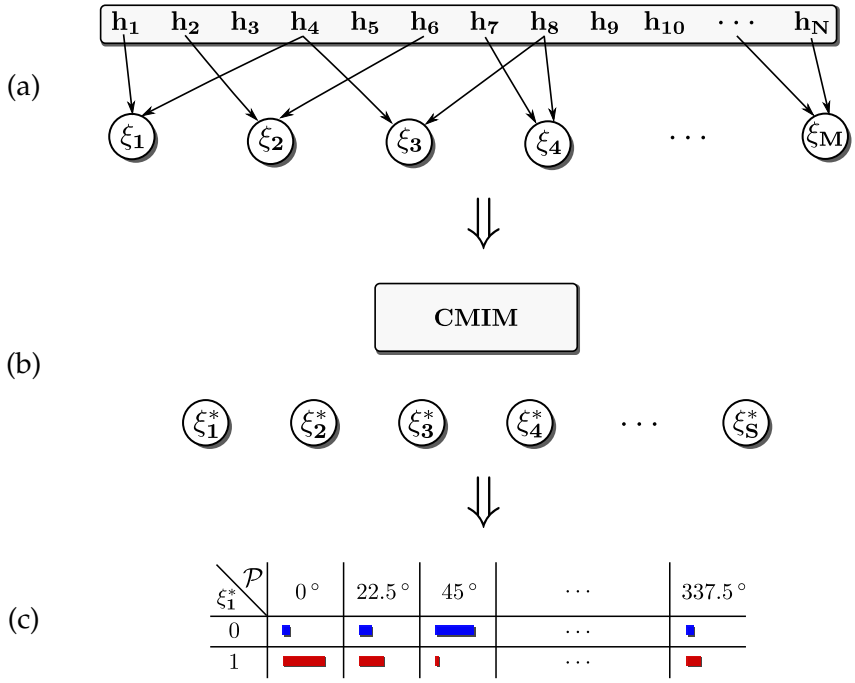


Figure 4.4: Feature selection for the pose estimator. **(a)** Initially, we generate ten thousand binary features ξ_i that compare two random bins of the histograms of Equation (4.2). **(b)** CMIM selects a subset of these features that have high mutual information with the object pose. **(c)** We build probability tables for each selected feature independently of the others. CMIM ensures that these tables will be discriminative when determining the object pose p.

where $\mathcal{H}^1$ is the histogram covering the whole bounding box, $\mathcal{H}^2$ to $\mathcal{H}^5$ are the four histograms that are computed on the second level, and so on.

Since some of the information present in the histograms is irrelevant for viewpoint estimation, we first define binary features on the histograms and select the ones that carry high mutual information with the pose bin value. We use binary features that compare two cluster label frequencies within the same region in the pyramid. We take the feature value to be

$$\rho_{i,j}^{k}(\mathcal{H}) = \begin{cases} 0 & \text{if } \mathcal{H}_i^k < \mathcal{H}_j^k \\ 1 & \text{otherwise} \end{cases}, \tag{4.3}$$

where $\mathcal{H}_i^k$ denotes the frequency of the i^{th} cluster in the k^{th} region in the pyramid.

We generate an initial feature set, denoted by $\mathcal{F}$, that contains a large number of features with randomly chosen parameters. Then a much smaller feature set is selected to be used in the pose estimation and we denote it by $\mathcal{F}_{\text{S}}$. In practice, there are

10000 features in $\mathcal{F}$ and 150 in $\mathcal{F}_{\mathrm{S}}$. The feature selection algorithm is based on conditional mutual information maximization [33], which sequentially picks features that carry high mutual information with the pose bin value, while avoiding features that are too similar to already picked ones. More exactly, we start with an empty set $\mathcal{F}_{\mathrm{S}}$ and select M features by repeatedly picking a feature $\hat{\xi}_i$ from $\mathcal{F}$ in the i^{th} selection round, removing it from $\mathcal{F}$ and adding to $\mathcal{F}_{\mathrm{S}}$. Denoting candidate features in $\mathcal{F}$ by ξ, and already selected ones in $\mathcal{F}_{\mathrm{S}}$ by ξ^*, $\hat{\xi}_i$ satisfies

$$\hat{\xi}_i = \underset{\xi \in \mathcal{F}}{\operatorname{argmax}} \ \min \left\{ I(\mathcal{P}; \xi), \min_{\xi^* \in \mathcal{F}_{\mathrm{S}}} I(\mathcal{P}; \xi \,|\, \xi^*) \right\}, \tag{4.4}$$

where $I(\mathcal{P}; \xi)$ is the mutual information between the object pose and a feature considered for selection, and $I(\mathcal{P}; \xi \,|\, \xi^*)$ is the value of the same quantity conditioned an already selected feature. They are both estimated using the training set.

We can visualize the relative importance of the different histograms for pose estimation by comparing the number of selected features that use each histogram as shown in Figure 4.6. The selected features almost never use the single histogram on the first level since it is too coarse. The remaining 3 levels contain 15, 44, and 38 percent of the features, respectively.

Since the selection process ensures only weak dependency between features, we approximate the mapping between the pyramid histograms and the object pose by

$$P(\mathcal{P}_i \,|\, \mathcal{H}) \approx P(\mathcal{P}_i \,|\, \mathcal{F}_{\mathrm{S}}(\mathcal{H})) \tag{4.5}$$

$$\approx \prod_{j}^{M} P(\mathcal{P}_i \,|\, \xi_j^*(\mathcal{H})), \tag{4.6}$$

where $\mathcal{F}_{\mathrm{S}}(\mathcal{H})$ represents the binary values of the features in $\mathcal{F}_{\mathrm{S}}$ and $\xi_j^*(\mathcal{H})$ the value of the j^{th} feature, all computed from the pyramid histograms. The feature probabilities $P(\mathcal{P}_i \,|\, \xi_j^*(\mathcal{H}))$ are again estimated from the training set. Figure 4.4 depicts the feature selection process and the naive Bayes modelling.

At run-time, given a bounding box in the image we compute the pyramid histograms and then use the learned mapping to estimate a distribution on the pose bins. For simplicity, we take the object pose to be the one that maximizes the probability for the corresponding bin. However, it would be straightforward to extend our approach to include multiple pose hypotheses using the pose probability distribution. Figure 4.3

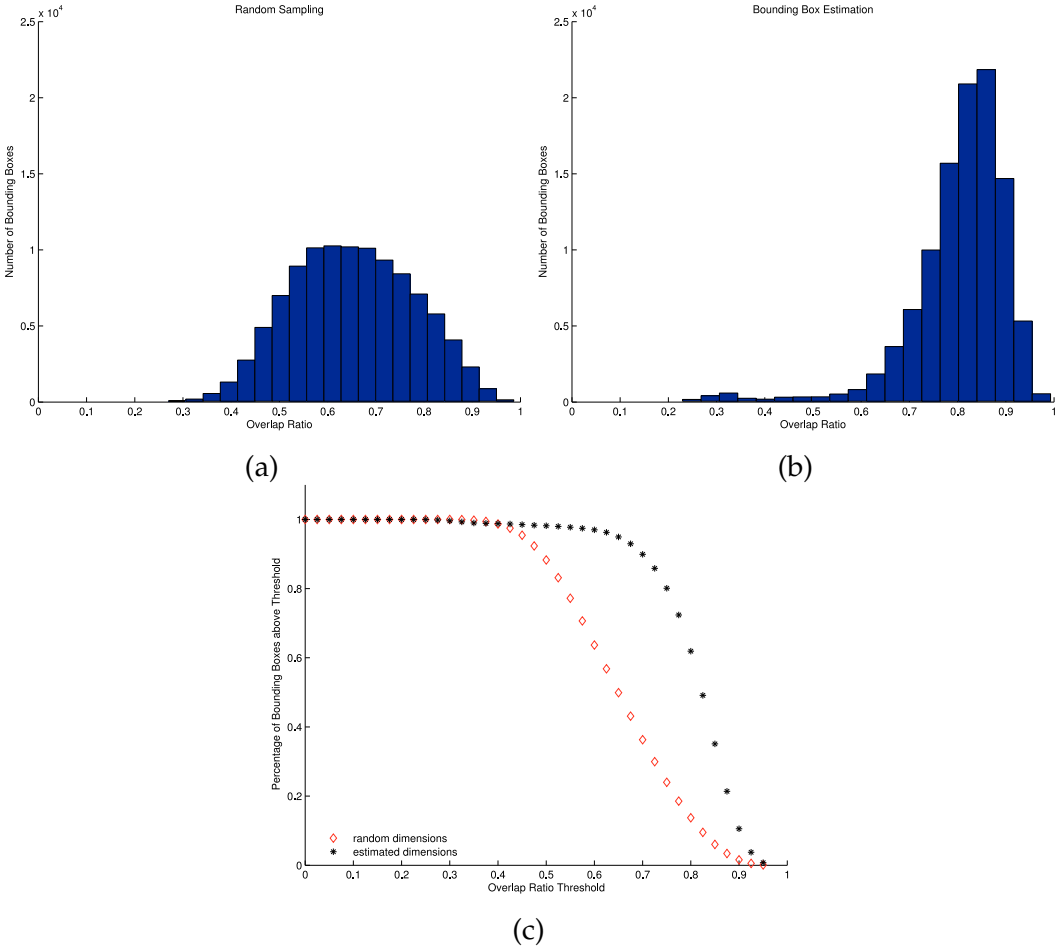


Figure 4.5: Bounding box estimation. **(a)** Histogram of the overlap ratios of the randomly sampled windows in the vicinity of the correct top left corner. **(b)** Histogram of the overlap ratios after bounding box estimation. The overlap ratio with the object has been greatly increased. **(c)** Ratio of windows that have larger overlap than a threshold, as the threshold is varied. Note that even for a conservative ratio of 0.7, most of the estimated windows can be considered as positive samples.

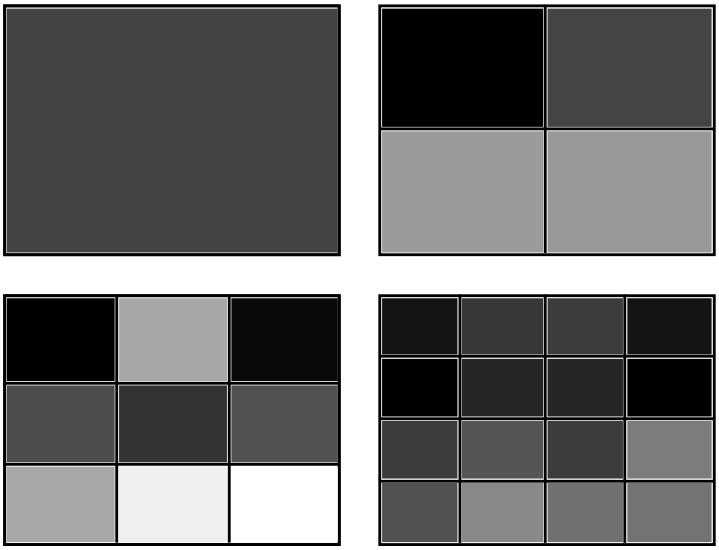


Figure 4.6: Histogram relevance for pose estimation. The brighter regions in the histogram pyramid denote higher importance. Feature selection has captured the importance of the lower parts of the bounding box. By contrast, the coarse first level and the corners in the upper part do not contribute to pose estimation.

depicts the estimated distributions over pose bins for an image sequence from the test set. The pose estimation is performed on the ground truth boxes. In Section 4.3, we show that partial overlap with ground truth is sufficient for reliable pose estimation.

4.2.3 Bounding Box Estimation

Bounding box estimation follows the same philosophy as pose estimation but involves estimating two variables, the bounding box aspect ratio and area. A straightforward approach would be to quantize their joint space into bins and estimate the correct bin from image features, exactly as above. However, the size of the joint space is large, which can bias the estimation in regions that receive a small number of training examples. To avoid these problems, we take a two step approach and treat the aspect ratio and area independently. We learn the distributions for both using the training set bounding boxes and divide the obtained value ranges into 20 equal bins.

We first learn an estimator for the aspect ratio using pyramid histograms from windows of fixed size, 150×150 pixels in our experiments. These windows are placed in the image so that their top left corners coincide with that of the training bounding

boxes. The estimator for the bounding box area is trained in the same way but using windows with the same aspect ratio as the training bounding boxes and of fixed height, taken to be 150 pixels in the experiments.

During testing, we use the trained estimators to select a single bounding box size with higher degree of overlap with the object than can be obtained by random sampling. To illustrate this, for each one of the 1000 test images, we sample 100 windows with random dimensions and top left corners within ± 10 pixels of the ground truth. The sampling distribution for the window size is computed from dimension statistics of the training set bounding boxes, and offset of the top left corner is uniform. The quality of a sampled window is measured by its overlap ratio (r) with the ground truth bounding box, which is computed in the standard way as

$$r = \frac{|\mathcal{B}_{\mathrm{G}} \cap \mathcal{B}|}{|\mathcal{B}_{\mathrm{G}} \cup \mathcal{B}|}, \tag{4.7}$$

where $\mathcal{B}$ represents the region covered by the sampled window and $\mathcal{B}_{\mathrm{G}}$ by the ground truth. We then measure the overlap ratio after resizing the windows to the dimensions obtained by bounding box estimation. First a fixed sized window is used to infer the aspect ratio. We then scale the width of the window to match the estimated value and update the pyramid histograms. The final window dimensions are obtained by estimating the area and by resizing the window to the estimated value. Figure 4.5 shows that the quality of the estimated dimensions is much better than random sampling and the bounding box estimator can reliably replace the exhaustive evaluation of all possible dimensions since it almost always finds a box of adequate size.

4.3 Results

We compare our estimators against a baseline implementation that uses a single SVM. The classifier uses spatial pyramid histograms that are built as follows. We extract DAISY descriptors[105] at every pixel in the training images. We randomly sample 100000 descriptors from the training images that are inside the object bounding boxes and obtain 100 cluster centers using K-Means. For each training image we compute 4 levels of spatial pyramid histograms as described in Section 4.2.1. Each pyramid contains 30 histograms, adding up to a 3000 dimensional representation.

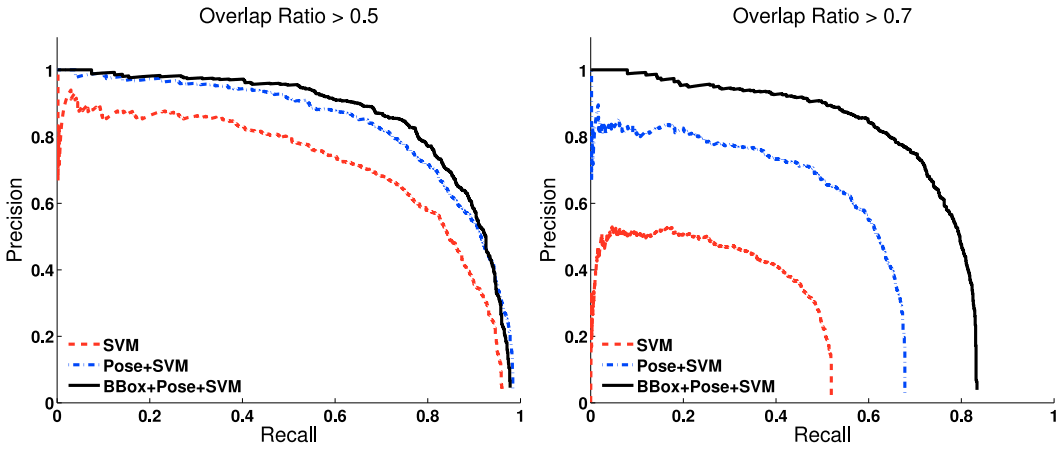
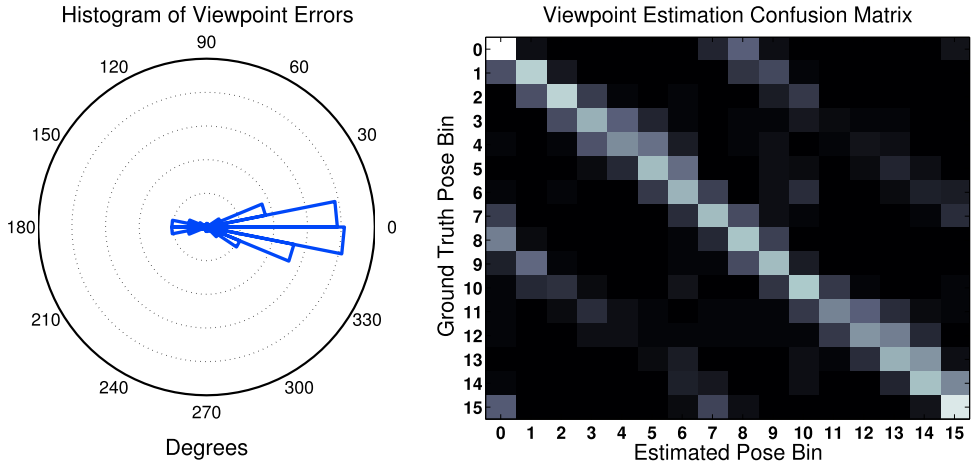


Figure 4.7: Precision/Recall curves comparing localization using only SVMs, using viewpoint estimation followed by a view-tuned SVM (Pose+SVM), and finally with the addition of bounding box estimation (BBox+Pose+SVM). **(a)** Curves when 0.5 bounding box overlap is accepted as positive detection, which is the standard threshold used in the literature. Adding viewpoint estimation improves the results and bounding box estimation leads to improved precision. **(b)** Curves when 0.7 bounding box overlap is required to be considered as a positive detection, which entails increased localization accuracy. Since boxes with smaller overlap can receive higher classification scores than boxes with more than 0.7 overlap, all curves degrade. However the degradation is much less severe when pose and window size estimation are turned on. In this more demanding context, it therefore yields even more clearly superior performance.



(a) (b)

Figure 4.8: Accuracy of pose estimation. **(a)** Histogram that shows the distribution of the error in the estimated pose in degrees. The small peak around 180 degrees is caused by the similarity in car appearance when seen from exactly opposite sides. **(b)** The confusion matrix showing the errors separately for each pose bin. As evidenced by the pose distributions from Figure 4.3, the pose errors are mostly due to the similarity of the front and back facing cars rather than due to confusion of side views. This is what produces the off diagonal terms in the confusion matrix.

Figure 4.9: Detections from the car show environment. We show correct detection results except in the last column which contains some false positives.

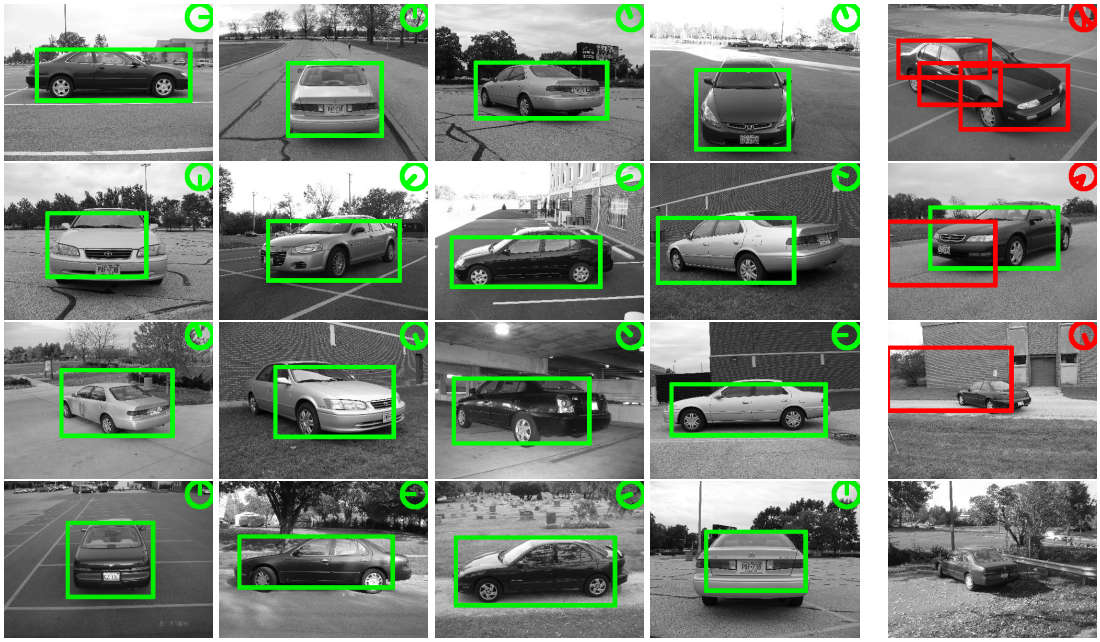


Figure 4.10: Detections on the database of [91]. The last column again contains false detections that can be attributed to failure in the bounding box estimation or to the fact that the scale of the cars is very different from the ones we used to train our SVMs.

The training set encompasses all images from the first 10 sequences, around 1000 images. In each one, we randomly pick 20 bounding boxes in addition to the ground truth box. These sampled boxes are labeled as positive or negative samples according to their overlap ratio defined by Equation (4.7). We further sample 9000 negative bounding boxes from 300 negative images that do not contain any cars. Using this training set, we train the viewpoint and bounding box estimators described in Section 4.2 and the baseline SVM. 16 view-tuned SVMs are then trained, each with positive samples only from a restricted viewpoint range but all the negative training set. In both cases, the SVM complexity parameters are found by cross-validation, training on 6 sequences and using the remaining 4 together with the 300 negative images as validation set.

Baseline approach. We detect cars in the test images by sliding windows and randomly sampling the window dimensions using the learned statistics from the training set. Each window is given a classification score by the baseline SVM and the non-maxima suppression removes windows that overlap with another window that received a higher score.

Our approach. To measure the performance of the viewpoint estimation we repeat the same process but this time we estimate the viewpoint for each sampled window using the Naive Bayes classifier and then compute the classification score with the selected view-tuned SVM. Finally, we also resize each sampled box to the dimensions given by the bounding box estimator and compute the score by view-tuned SVMs.

The bounding box estimator computes features within two extra windows compared to using the sampled box dimensions, one fixed size and another one with fixed height. Hence, to even out the amount of computation required by each experiment we sample three times as many windows when bounding box estimation is disabled.

Figure 4.7 depicts the precision/recall curves drawn for the test set containing 10 sequences of car images and 1000 images that do not contain any cars. The pose estimation yields a much improved curve compared to a single SVM and the bounding box estimation improves localization. The effect of bounding box estimation is more pronounced as higher accuracy is desired in bounding box dimensions.

We also test the accuracy of the pose estimation. During testing, for each bounding box that has overlap ratio with the ground truth greater than 0.5, we record the estimated pose value and compare it to the ground truth. Figure 4.8 shows the histogram of errors and the confusion matrix for the estimated pose bin. Table 4.1 lists the confusion matrix entries in percent.

By setting the threshold on the classification score to be the one that yields equal precision and recall, we obtain the detection results shown in Figure 4.1. We then ran our car detector on images acquired at the car show including cars not on rotating platforms and the results are depicted by Figure 4.9. We also tested our detector on the database provided by [91] and we show some representative detection results in Figure 4.10. On the binary car detection task, we achieve performances that are roughly equivalent to those reported in [91] even though we did not retrain our system for this case. This demonstrates that our estimators generalize well to images taken under much more generic conditions than those we trained for.

4.4 Conclusion

In this chapter, we have introduce a pose covariant object detection framework. Since pose covariance requires pose and scale estimators for an object category, we have

Table 4.1: Confusion matrix entries for object pose estimation in percent.

Bin #	0	1	2	3	4	5	6	7
0	59.5	3.1	0.1	0.1	0.1	0.0	0.1	7.7
1	17.8	46.0	4.7	0.1	0.0	0.0	0.1	0.9
2	0.2	17.8	46.6	13.2	1.4	0.1	1.0	0.2
3	0.6	0.4	17.2	38.2	22.1	7.6	1.2	0.2
4	1.0	0.9	1.3	18.9	31.5	25.3	5.9	0.4
5	0.3	0.6	0.8	2.3	8.8	41.0	24.9	0.5
6	1.1	0.6	1.4	0.6	0.1	12.1	39.6	14.5
7	13.8	0.7	0.3	0.7	0.1	0.2	9.6	41.1
8	28.5	1.9	0.1	0.7	0.4	0.1	1.2	8.5
9	6.0	23.9	1.1	0.1	0.0	0.2	1.1	1.4
10	1.3	8.9	10.9	2.0	0.1	0.1	4.4	0.8
11	1.8	0.0	2.1	10.0	3.5	1.2	0.9	1.6
12	1.8	0.0	0.0	3.2	3.1	1.8	1.8	1.1
13	0.5	0.0	0.0	0.4	0.7	2.7	6.2	0.6
14	1.4	0.0	0.0	0.0	0.0	0.7	7.1	4.9
15	21.2	0.0	0.0	0.0	0.0	0.0	2.1	15.4
Bin #	8	9	10	11	12	13	14	15
0	21.9	3.4	0.0	0.0	0.0	0.1	0.0	3.8
1	11.9	16.6	1.5	0.0	0.0	0.1	0.0	0.0
2	0.5	5.6	12.6	0.6	0.1	0.1	0.0	0.0
3	1.2	1.0	5.4	2.3	1.7	1.0	0.0	0.0
4	1.6	3.4	0.9	1.2	4.3	3.5	0.0	0.0
5	0.6	3.0	2.8	0.2	2.1	8.0	3.6	0.4
6	0.9	3.4	9.6	0.1	0.0	2.4	7.1	6.4
7	17.0	3.0	1.5	0.3	0.2	0.1	1.3	10.1
8	43.0	14.9	0.2	0.1	0.1	0.1	0.1	0.3
9	17.2	41.3	5.7	0.0	0.0	0.3	0.3	0.3
10	0.3	11.3	43.8	12.2	1.3	0.6	0.4	1.6
11	1.5	1.0	12.6	28.9	21.7	9.6	2.3	1.3
12	1.2	1.2	1.8	12.1	33.1	28.3	8.9	0.6
13	0.3	0.4	3.3	1.5	9.6	38.2	32.6	2.8
14	0.1	0.4	2.9	0.9	0.5	7.9	42.5	30.6
15	3.7	0.3	0.1	0.2	0.3	0.0	4.0	52.7

acquired a dataset that contains continuous pose annotation and used it to train these estimators. By comparing their output to ground truth values obtained similarly, we showed that these estimators are accurate and stable. When used in conjunction with an object detector, they yield improved localization.

Moreover our scheme is able to simultaneously recover object pose at detection time. This is useful to provide contextual information since the orientations of objects play a fundamental role in scene understanding. For example the cars and pedestrians often move along a road in a single direction. Such consistent behavior can be used to filter out clutter and noise [39, 45, 48].

Whether pose estimation can be performed reliably for an object category depends on the variation of image features as a function of object pose. In case these features vary significantly and in a statistically meaningful way it is easier to design estimators to capture the pose information. The feature selection algorithm we have used in training (CMIM) relies on the existence of such features and picks the best ones to be included in pose estimation. On the other hand in case the features do not change at all then the object detection task can be performed easily using a pose independent classifier. Obviously in this case even defining a pose space for the object is not possible. Therefore automatic ways to determine the extend of pose variation and grouping of category instances with similar pose is a requirement for further developments in the field. The latent variables used in recent deformable part based models [29] can represent a change in object pose and each individual part model can represent appearance from a separate viewpoint. As a result they have a potential to be used in covariant pose detection and estimation of the viewpoint. However doing this automatically requires decoupling appearance changes due to pose from intra-category variations. The database we have collected and also the one of [91] exactly targets this kind of classifier training.

Overall, we argue that understanding the limitations of both pose invariant and covariant classifiers will allow us to tackle some fundamental issues in category-level object detection. Databases with continuous feature variation and pose annotation will be a requirement for this task. Using such voluminous data might require weaker supervision during training. We believe the experience gained from instance-level object detection can be instrumental in the design of novel training algorithms that require less supervision.

CHAPTER

FIVE

CONCLUSION

We have presented the frameworks we have developed to match feature points, to automatically train a 3D object detection system, and to detect objects of a single category together with pose annotations. In all cases, we exploit the presence of training data that densely covers a wide range of viewpoints.

We showed that it is possible to match keypoints fast by using a powerful naive Bayesian classifier that is easy to train. We tested our approach on various 2D and 3D datasets. It allows robust real-time Augmented Reality on various application domains. We also compared it to existing approaches using Randomized Trees steps or local descriptors. It outperforms similar methods and it is more suited to real-time applications than local descriptors.

The main disadvantage of using large training sequences is the amount of necessary manual labelling. We developed a weakly supervised training algorithm for our keypoint matching approach that greatly reduces the training requirements. This algorithm is also effective to remove keypoints that do not contribute consistently to reliable detection of the object. Such data filtering is essential to be able to handle larger training data and also has a significant effect on the run-time detection quality.

We have demonstrated the combination of these approaches on a 3D tracking-by-detection task. Using a training image sequence, we trained an object detector that can recover the 3D camera pose independently in every frame of a test video. Such a detector greatly improves the performance of object tracking by providing a means

of recovery and preventing drift. We showed that our system is robust to scale and lighting changes and it can handle occlusions.

Finally, we extended the use of viewpoint-dense training data to category-level object detection. This is not possible with readily available datasets, so we have collected a new one that contains image sequences that depict objects as the viewpoint varies slowly. It permits us to construct an estimator for the object pose. We showed that this estimator produces reliable and stable output and it can be utilized to improve detection performance. Although we have used a standard linear SVM baseline in our comparisons, any object detection algorithm can be integrated in our framework to improve its robustness to viewpoint variations.

Overall, we have demonstrated various ways to train classifiers to perform view independent object detection. In some cases, these classifiers employ invariant statistics to dampen the effects of viewpoint variation. Others estimate the extend of these variations with respect to a reference view. This estimate guides the analysis of image features and the way they are classified. We showed that given suitable training data both approaches can be used in novel ways to improve the run-time performance on several vision tasks.

Limitations and Future Work

Our keypoint classification system already satisfies the constraints of most Augmented Reality systems. The main drawback of this system is its memory consumption. Although several modifications have already been proposed, these only reduce the required amount by a relatively small integer factor. At the moment, we represent and store the learned probabilistic model as multinomial histograms independently for each keypoint. It should, at least in principle, be possible to either share some of these histogram data between keypoints or to replace the histograms with a parametric distribution with much fewer variables.

The most recent image retrieval systems can easily handle thousands of images that contain on the order of thousand keypoints each. Our approach currently does not generalize to this more demanding domain. This is a consequence of allocating a separate class for each keypoint. It might be possible to remedy this by clustering keypoints and by replacing the linear classification structure with a hierarchical one.

A possible research direction is to follow the combined statistical and descriptor based approach of [15], discussed in Section 2.1, that collects generic texture statistics to compute compact signatures for keypoints. This is beneficial since the dictionary used in the signature generation can be quite small and it is shared by all keypoints. All current work on signatures is based on learning pose invariant texture statistics, but a more pose-aware approach might be needed to go beyond current limitations on operational viewpoint range. More specifically, the signature will vary as a function of the viewpoint parameters. Even for fixed pose, there might be multiple alternative base keypoints to be included in the keypoint signature since Randomized Trees respond similarly for all of them. These alternatives can be learned to assign a set of signatures to each new keypoint instead of a single one. Each member of this set can be annotated with its valid viewpoint range to simultaneously recover keypoint identity and object pose information at run-time.

Our feature harvesting framework overcomes the need to manually label all training images. However this is achieved by relying on a rough geometric model of the object. This model is necessary to segment the object in novel images with very different viewpoints from the initial frame. Removing it from the training prerequisites would greatly increase the flexibility of our approach and ease of training for new objects. One possibility is to replace this stringent geometric constraint by a prior on object motion. Since we already require a restricted environment, we can also require the object and background features to have inconsistent motion vectors without severely encumbering the training process.

In Chapter 4, we have demonstrated that our category-level object detection pipeline is effective to detect cars from multiple viewpoints. The selection of cars as a category to test our approach is not an arbitrary one. Estimating object pose naturally requires a well defined pose space and image features that vary consistently with viewpoint, both of which are well satisfied by the cars object category. Similar arguments can be made for bikes, motorcycles, TV sets, and chairs. Note that all these categories have a well defined *front face* enforced by their function and edges in their images smoothly vary with viewpoint. On the other hand, categories like trees, glasses, and pedestrians either do not have a completely well defined pose or their dominant image features do not change much with viewpoint. These won't fit into our framework easily. This also means that they are much easier to recognize from multiple viewpoints using

image features only from a specific view. As a result, there is a need to handle pose variations in a more automated manner, using a combination of pose invariant and covariant approaches.

APPENDIX

A

SOLVING THE PERSPECTIVE-THREE-POINT PROBLEM FOR CAMERA POSE ESTIMATION

We present a brief overview of the Perspective-Three-Point (P3P) algorithm that we used to recover the 3D camera pose as described in Chapter 3. We refer the readers to [23, 40] for a more detailed account of solutions to P3P problem.

The goal in solving a P3P problem is to determine the location of three 3D points $\{\mathbf{M}_i^C\}$ in the camera coordinate system using their projections $\{\mathbf{m}_i\}$ in the image plane. The assumption is that we already know the 3D coordinates of these points $\{\mathbf{M}_i^W\}$ in a world coordinate system, they are part of a rigid scene, and they are not coplanar with the camera center $\mathbf{C}$. Consequently, 3D inter-point distances $\{d_{ij}\}_{i<j}$ between each pair will be the same in the camera coordinate system.

We also assume knowledge of camera internal calibration matrix $\mathtt{K}$. Therefore given a pair of 2D points $\mathbf{m}_i$ and $\mathbf{m}_j$, we can back-project rays $\mathbf{r}_i$ and $\mathbf{r}_j$ from $\mathbf{C}$. They are given by

$$\mathbf{r}_i = \mathtt{K}^{-1}\mathbf{m}_i \text{ and } \mathbf{r}_j = \mathtt{K}^{-1}\mathbf{m}_j. \tag{A.1}$$

We can also compute the angle between them Θ_{ij} as

$$\cos\Theta_{ij} = \frac{\mathbf{m}_i^T \omega \mathbf{m}_j}{\left(\mathbf{m}_i^T \omega \mathbf{m}_i\right)^{1/2}\left(\mathbf{m}_j^T \omega \mathbf{m}_j\right)^{1/2}}, \tag{A.2}$$

where $\omega = \left(\mathtt{K}\mathtt{K}^T\right)^{-1}$ is the image of the absolute conic [47].

Since we know that the 3D point $\mathbf{M}_i^C$ must lie on $\mathbf{r}_i$ and $\mathbf{M}_j^C$ on $\mathbf{r}_j$, the only unknowns are the distances between $\mathbf{C}$ and the 3D points $\mathbf{M}_i^C$ and $\mathbf{M}_j^C$, which we denote

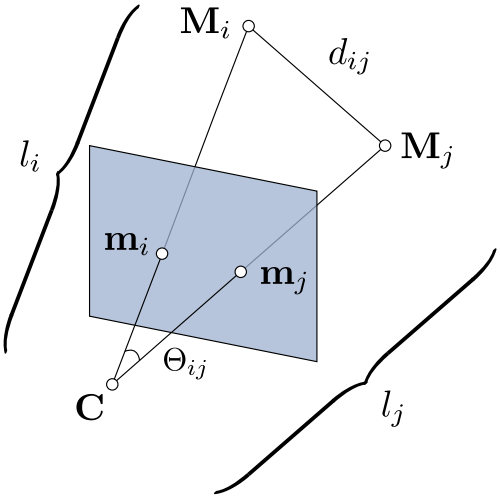


Figure A.1: Geometry of the perspective-three-point problem. Since the angle Θ_{ij} and the distance d_{ij} is known, it is possible to write a quadratic constraint on the 3D lengths l_i and l_j using the cosine law.

by l_i and l_j respectively. As shown by Figure A.1, these satisfy the law of cosines

$$d_{ij}^2 = l_i^2 + l_j^2 - 2l_i l_j \cos\Theta_{ij}. \tag{A.3}$$

Since we have three points, each pair generates an independent equation that can be stacked to form the P3P system of equations

$$\begin{aligned} d_{12}^2 &= l_1^2 + l_2^2 - 2l_1 l_2 \cos\Theta_{12} \\ d_{13}^2 &= l_1^2 + l_3^2 - 2l_1 l_3 \cos\Theta_{13} \\ d_{23}^2 &= l_2^2 + l_3^2 - 2l_2 l_3 \cos\Theta_{23}. \end{aligned} \tag{A.4}$$

[40] presents a complete set of solutions to this system by decomposing it into ten components and solving each of these polynomial systems independently. The first one of these is called the *main component* and the rest contain degenerate configurations. The solution involves computing the *resultant* of two polynomials which removes one of the variables but increases the degree of the remaining system.

We have used the solution to the main component in our implementation, which we found to be robust enough for our purposes. It may contain up to four solutions, but in practice most of the time one or two solutions will be found. Figure A.2 illustrates the solution space. Multiplicity of the solutions is not a problem since we use

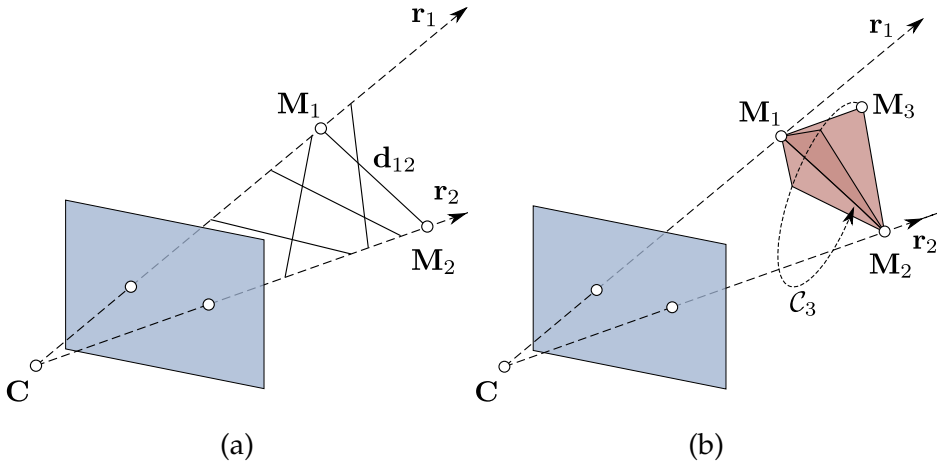


Figure A.2: Solution space of the P3P problem [114]. **(a)** The locations of $\mathbf{M}_1^C$ and $\mathbf{M}_2^C$ are determined by fitting a line segment of length d_{12} between rays $\mathbf{r}_1$ and $\mathbf{r}_2$. **(b)** The points $\mathbf{M}_1^C$, $\mathbf{M}_2^C$, and $\mathbf{M}_3^C$ define a 3D triangle that fits on the baseline defined by $\mathbf{M}_1^C$ and $\mathbf{M}_2^C$. For a fixed baseline $\mathbf{M}_3^C$ lies on a circle $\mathcal{C}_3$. Then the family of possible locations for the baseline in **(a)** defines a family of circles. The set of P3P solutions contains the intersections of ray $\mathbf{r}_3$ with this circle family.

RANSAC [32] to select the best one. Moreover, since P3P requires only three 2D-to-3D correspondences, RANSAC can converge very fast even when the outlier ratio is relatively high.

Solution to P3P gives the lengths $\{l_i\}$ and this is enough to compute the locations of the 3D points in the camera coordinate system $\{\mathbf{M}_i^C\}$. To find the camera pose with respect to the world coordinate system, we need to compute the rotation $\mathtt{R}$ and translation $\mathbf{t}$ between the point sets $\{\mathbf{M}_i^C\}$ and $\{\mathbf{M}_i^W\}$. This is called the *absolute orientation* and [52] gives a closed form algorithm to solve for it.

A. SOLVING THE PERSPECTIVE-THREE-POINT PROBLEM FOR CAMERA POSE ESTIMATION

APPENDIX

B

HOMOGRAPHY PARAMETRIZATION FOR TRAINING FERNS

For planar objects, the training for Ferns and Randomized Trees involves generating random homography deformations representing transformations of a model image in 3D. In Section 2.3, we have sampled from parameters of affine transformations to obtain these deformations. We represent affine image deformations in the form

$$A = R_\theta R_{-\phi} \mathrm{diag}(\lambda_1, \lambda_2) R_\phi, \tag{B.1}$$

where $\mathrm{diag}(\lambda_1, \lambda_2)$ is a diagonal 2×2 scaling matrix and R_γ represents a rotation of angle γ. $R_{-\phi}$ controls the degree of skew deformation.

While this representation is suitable for many applications, it suffers from several limitations. First, a uniform sampling of the skew angle $R_{-\phi}$ does not result in a uniform sampling of skew deformations. This is due to the effect of scaling parameters λ_1 and λ_2). When the values of these are close to each other the skew angle has little effect. As a result, training set reflects scale changes stronger than skew. Second, it is difficult to translate constraints on viewpoint into restrictions on the parameter values. For example, we might restrict viewpoints to no more than ± 20 degrees of out of plane rotations. Affine parameterization does not easily allow this. Although *rejection sampling* can be used to simulate these constraints, it is not very efficient.

In this appendix, we derive a parameterization of the homography space that better reflects viewing conditions. We assume that the model image for training Ferns

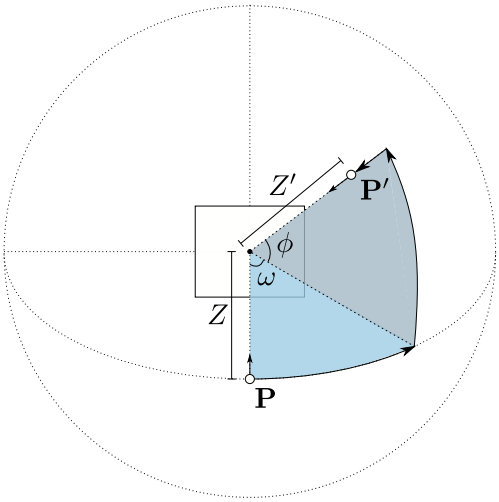


Figure B.1: Geometry of the homography parameterization.

have been captured from a fronto-parallel viewpoint. The camera is placed at a distance Z from the planar target as depicted by Figure B.1 and we write its projection matrix as

$$\mathbf{P} = \mathtt{K}\,[\mathtt{I} \mid \mathbf{0}]\,. \tag{B.2}$$

Therefore, model camera coordinate system coincides with the world coordinate system.

We define another coordinate system whose origin is at the center of the planar object and its xy-plane is parallel to the object surface. The z-axis points away from the plane towards the model image camera. We refer to this system as the *object coordinate system*. The model image camera center is at $(0, 0, Z)^T$ in this coordinate system. To generate the training images, we move the camera to a location given by the spherical object coordinates $(Z', \omega, \phi)^T$. The training camera z-axis points towards the object center. Its y-axis is in the direction of the normal vector $(-\sin\phi\sin\omega, \cos\phi, -\sin\phi\cos\omega)^T$ in the object coordinate system. We also rotate the training camera in its xy-plane by an angle θ to simulate in-plane rotations. This completely specifies its location and orientation using four parameters $(Z', \omega, \phi, \theta)^T$. Note that this is smaller than the six parameters required to specify a camera in 3D space. The missing two correspond to a translation of the camera in the x and y directions of the object coordinate system. Since this will result in a shift in the generated training

image, we do not need to explicitly set these. The placement and orientation scheme detailed above implicitly defines default values for these two remaining parameters.

We can write the training projection matrix in the world coordinate system as

$$\mathbf{P}' = \mathtt{K}' \left[\mathtt{R} \,|\, \mathbf{t} \right]. \tag{B.3}$$

Once we specify $\mathtt{R}$ and $\mathbf{t} = (t_0, t_1, t_2)^T$, we can write the transfer homography $\mathtt{H}$ between the model and training images [47] as

$$\mathtt{H} = \mathtt{K}' \left(\mathtt{R} - \begin{bmatrix} 0 & 0 & -t_0/Z \\ 0 & 0 & -t_1/Z \\ 0 & 0 & -t_2/Z \end{bmatrix} \right) \mathtt{K}^{-1}. \tag{B.4}$$

It is not easy to write the rotation matrix directly and we derive it using quaternions. Please refer to [52] for a detailed description. To follow the discussion here it is enough to know that a 3D rotation about a unit vector $\mathbf{u}$ by an angle δ can be represented by using a four element vector $\mathbf{q} = (\cos(\delta/2),\, \sin(\delta/2)\, \mathbf{u})^T$. The rotation matrix $\mathtt{R}$ is a composition of three consecutive rotations represented by the following set of quaternions

$$\begin{aligned} \mathbf{q}_\omega &= (\cos(\omega/2),\, \sin(\omega/2)\, \mathbf{n}_\omega)^T \\ \mathbf{q}_\phi &= (\cos(\phi/2),\, \sin(\phi/2)\, \mathbf{n}_\phi)^T \\ \mathbf{q}_\theta &= (\cos(\theta/2),\, \sin(\theta/2)\, \mathbf{n}_\theta)^T. \end{aligned} \tag{B.5}$$

The first one is a rotation about the y-axis and the corresponding normal is

$$\mathbf{n}_\omega = (0, 1, 0)^T. \tag{B.6}$$

The second one is a rotation about the x-axis, but x-axis has been transformed by the first rotation so the corresponding normal is

$$\mathbf{n}_\phi = (\cos\omega, 0, \sin\omega)^T. \tag{B.7}$$

Finally, the camera is rotated about its z-axis to perform an in-plane rotation and the corresponding normal can be written as

$$\mathbf{n}_\theta = \begin{bmatrix} -\sin\omega\cos\phi \\ \sin\phi \\ \cos\omega\cos\phi \end{bmatrix}. \tag{B.8}$$

The quaternion representing the combined rotation is given by the multiplication of the three quaternions $\mathbf{q}_{\mathtt{R}} = \mathbf{q}_\theta \cdot \mathbf{q}_\phi \cdot \mathbf{q}_\omega = (q_0, q_1, q_2, q_3)^T$ and the rotation matrix can be computed using the Rodrigues Formula

$$\mathtt{R} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2\left(q_1 q_2 + q_0 q_3\right) & 2\left(q_1 q_3 + q_0 q_2\right) \\ 2\left(q_1 q_2 + q_0 q_3\right) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2\left(q_2 q_3 + q_0 q_1\right) \\ 2\left(q_1 q_3 + q_0 q_2\right) & 2\left(q_2 q_3 + q_0 q_1\right) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix}. \tag{B.9}$$

The training camera center can be directly written as

$$\mathbf{C}' = \begin{bmatrix} Z' \cos\phi \sin\omega \\ -Z' \sin\phi \\ Z - Z' \cos\phi \cos\omega \end{bmatrix}. \tag{B.10}$$

The corresponding translation vector is

$$\mathbf{t} = -\mathtt{R}\mathbf{C}'. \tag{B.11}$$

REFERENCES

[1] Y. AMIT AND D. GEMAN. **Shape Quantization and Recognition with Randomized Trees**. *Neural Computation*, **9**(7):1545–1588, 1997. 13, 14

[2] Y. AMIT AND A. TROUVÉ. **POP: Patchwork of Parts Models for Object Recognition**. *Computer Vision and Image Understanding*, **75**(2):267–282, November 2007. 77

[3] A. ANDONI AND P. INDYK. **Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions**. *Communications of the ACM*, **51**(1):117–122, 2008. 17, 33

[4] B. BABENKO, M.-H. YANG, AND S. BELONGIE. **Visual Tracking with Online Multiple Instance Learning**. In *Conference on Computer Vision and Pattern Recognition*, 2009. 4, 54, 55, 57, 74

[5] R. BARANIUK. **Compressive Sensing**. *Journal of Machine Learning Research*, 2007. 26

[6] H. BAY, A. ESS, T. TUYTELAARS, AND L. VAN GOOL. **SURF: Speeded Up Robust Features**. *Computer Vision and Image Understanding*, **10**(3):346–359, 2008. 26

[7] J. BEIS AND D.G. LOWE. **Learning indexing functions for 3-D model-based object recognition**. In *Conference on Computer Vision and Pattern Recognition*, pages 275–280, Seattle, 1994. 1

[8] J. BEIS AND D.G. LOWE. **Shape Indexing using Approximate Nearest-Neighbour Search in High-Dimensional Spaces**. In *Conference on Computer Vision and Pattern Recognition*, pages 1000–1006, Puerto Rico, 1997. 12, 44, 58

[9] C.M. BISHOP. *Pattern Recognition and Machine Learning*. Springer, 2006. 32

[10] M.B. BLASCHKO AND C.H. LAMPERT. **Learning to Localize Objects With Structured Output Regression**. In *European Conference on Computer Vision*, pages 2–15, 2008. 4, 77, 78

[11] A. BOSCH, A. ZISSERMAN, AND X. MUNOZ. **Image Classification using Random Forests and Ferns**. In *International Conference on Computer Vision*, 2007. 26

[12] J. BRIAN BURNS, R. S. WEISS, AND E. M. RISEMAN. **View Variation of Point-Set and Line-Segment Features**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15**(1):51–68, 1993. 2

[13] M. CALONDER, V. LEPETIT, AND P. FUA. **Keypoint Signatures for Fast Learning and Recognition**. In *European Conference on Computer Vision*, 2008. 26

[14] M. CALONDER, V. LEPETIT, AND P. FUA. **Pareto-optimal Dictionaries for Signatures**. In *Conference on Computer Vision and Pattern Recognition*, 2010. 26

[15] M. CALONDER, V. LEPETIT, K. KONOLIGE, J. BOWMAN, P. MIHELICH, AND P. FUA. **Compact Signatures for High-speed Interest Point Description and Matching**. In *International Conference on Computer Vision*, 2009. 26, 97

[16] G. CAUWENBERGHS AND T. POGGIO. **Incremental and Decremental Support Vector Machine Learning**. In *Neural Information Processing Systems*, pages 409–415, 2000. 3

[17] C. CHOW AND C. LIU. **Approximating Discrete Probability Distributions with Dependence Trees**. *Information Theory, IEEE Transactions on*, **14**(3):462–467, 1968. 29

[18] O. CHUM AND J. MATAS. **Matching with PROSAC - Progressive Sample Consensus**. In *Conference on Computer Vision and Pattern Recognition*, pages 220–226, San Diego, CA, June 2005. 48, 68

[19] O. CHUM AND A. ZISSERMAN. **An Exemplar Model for Learning Object Classes**. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007. 4, 15, 77

[20] D. COMANICIU, V. RAMESH, AND P. MEER. **Kernel-Based Object Tracking**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **25**(5):564–575, 2003. 1

[21] PEGASE CONSORTIUM. **Helicopter and Aeronef Navigation Airborne System Experimentations**. `http://dassault.ddo.net/pegase/index.php`. 8

[22] A. J. DAVISON. **Real-Time Simultaneous Localisation and Mapping with a Single Camera**. In *International Conference on Computer Vision*, page 1403, 2003. 54

[23] D. DEMENTHON AND L.S. DAVIS. **Exact and Approximate Solutions of the Perspective-Three-Point Problem**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **14**(11):1100–1105, November 1992. 99

[24] P. DOMINGOS AND M. PAZZANI. **On the Optimality of the Simple Bayesian Classifier under Zero-One Loss**. *Machine Learning*, **29**(2-3):103–130, 1997. 24

[25] M. EVERINGHAM, L. VAN GOOL, C. K. I. WILLIAMS, J. WINN, AND A. ZISSERMAN. **The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results**. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html. 4

[26] L. FEI-FEI, R. FERGUS, AND P. PERONA. **One-Shot Learning of Object Categories**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**(4):594–611, 2006. 4, 25

[27] P. FELZENSZWALB AND D. HUTTENLOCHER. **Pictorial Structures for Object Recognition**. *Computer Vision and Image Understanding*, **16**(1), 2005. 15

[28] P. FELZENSZWALB, D. MCALLESTER, AND D. RAMANAN. **A Discriminatively Trained, Multiscale, Deformable Part Model**. In *Conference on Computer Vision and Pattern Recognition*, June 2008. 4, 77

[29] 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*, 2009. 4, 15, 77, 93

[30] V. FERRARI, F. JURIE, AND C. SCHMID. **From Images to Shape Models for Object Detection**. *International Journal of Computer Vision*, **87**(3):284–303, 2010. 15

[31] S. FIDLER AND A. LEONARDIS. **Towards Scalable Representations of Objects Categories: Learning a Hierarchy of Parts**. In *Conference on Computer Vision and Pattern Recognition*, 2007. 15

[32] M.A FISCHLER AND R.C. BOLLES. **Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography**. *Communications ACM*, **24**(6):381–395, 1981. 48, 101

[33] F. FLEURET. **Fast Binary Feature Selection with Conditional Mutual Information**. *Journal of Machine Learning Research*, **5**:1531–1555, November 2004. 76, 84

[34] F. FLEURET AND D. GEMAN. **Coarse-To-Fine Visual Selection**. *International Journal of Computer Vision*, **41**(1):85–107, January 2001. 55

[35] F. FLEURET AND D. GEMAN. **Stationary Features and Cat Detection**. *Journal of Machine Learning Research*, **9**:2549–2578, 2008. 15, 77

[36] Y. FREUND AND R. E. SCHAPIRE. **A Decision-theoretic Generalization of On-line Learning and an Application to Boosting**. *Journal of Computer and System Sciences*, **55**(1):119–139, 1997. 54

[37] J. H. FRIEDMAN AND U. FAYYAD. **On Bias, Variance, 0/1-loss, and the Curse-of-Dimensionality**. *Data Mining and Knowledge Discovery*, **1**:55–77, 1997. 24

[38] B. FULKERSON, A. VEDALDI, AND S. SOATTO. **Localizing Objects With Smart Dictionaries**. In *European Conference on Computer Vision*, 2008. 4, 77

[39] C. GALLEGUILLOS, A. RABINOVICH, AND S. BELONGIE. **Object Categorization Using Co-Occurrence, Location and Appearance**. *Journal of Machine Learning Research*, June 2008. 93

[40] X.-S. GAO, X.-R. HOU, J. TANG, AND H.-F. CHENG. **Complete Solution Classification for the Perspective-Three-Point Problem**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **25**(8):930–943, 2003. 48, 99, 100

[41] P. GEURTS, D. ERNST, AND L. WEHENKEL. **Extremely Randomized Trees**. *Machine Learning*, **36**(1):3–42, 2006. 14

[42] A. GIONIS, P. INDYK, AND R. MOTWANI. **Similarity Search in High Dimensions via Hashing**. In *VLDB'99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529, 1999. 23, 32

[43] H. GRABNER, C. LEISTNER, AND H. BISCHOF. **Semi-supervised On-line Boosting for Robust Tracking**. In *European Conference on Computer Vision*, 2008. 4, 54, 55, 57, 74

[44] G. GRIFFIN, A. HOLUB, AND P. PERONA. **Caltech-256 Object Category Dataset**. Technical Report 7694, California Institute of Technology, 2007. 4

[45] A. GUPTA AND L. DAVIS. **Beyond Nouns: Exploiting Prepositions and Comparators for Learning Visual Classifiers**. In *European Conference on Computer Vision*, 2008. 93

[46] C. HARRIS. *Tracking With Rigid Objects*. MIT Press, 1992. 1

[47] R. HARTLEY AND A. ZISSERMAN. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000. 99, 105

[48] G. HEITZ AND D. KOLLER. **Learning Spatial Context: Using Stuff to Find Things**. In *European Conference on Computer Vision*, 2008. 93

[49] G. E. HINTON AND R. R. SALAKHUTDINOV. **Reducing the Dimensionality of Data with Neural Networks**. *Science*, **313**(5786):504–507, July 2006. 12

[50] G.E. HINTON. **Training Products of Experts by Minimizing Contrastive Divergence**. *Neural Computation*, **14**:1771–1800, 2002. 24

[51] D. HOIEM, R. SUKTHANKAR, H. SCHNEIDERMAN, AND L. HUSTON. **Object-Based Image Retrieval Using the Statistical Structure of Images**. In *Conference on Computer Vision and Pattern Recognition*, **02**, pages 490–497, 2004. 28

[52] B. K. P. HORN. **Closed-form Solution of Absolute Orientation using Unit Quaternions**. *Journal of the Optical Society of America. A*, **4**(4):629–642, April 1987. 101, 105

[53] D.P. HUTTENLOCHER AND S. ULLMAN. **Object Recognition using Alignment**. In *International Conference on Computer Vision*, pages 102–111, 1987. 1

[54] K. IKEUCHI AND T. KANADE. **Automatic Generation of Object Recognition Programs**. *Proceedings of the IEEE*, **76**(8):1016–1035, August 1988. 1

[55] F. JURIE. **Tracking Objects With a Recognition Algorithm**. *Journal of Machine Learning Research*, **3-4**(19):331–340, 1998. 1

[56] F. JURIE AND C. SCHMID. **Scale-invariant Shape Features for Recognition of Object Categories**. In *Conference on Computer Vision and Pattern Recognition*, **2**, pages II–90–II–96 Vol.2, 2004. 15

[57] Z. KALAL, J. MATAS, AND K. MIKOLAJCZYK. **P-N Learning: Bootstrapping Binary Classifiers from Unlabeled Data by Structural Constraints**. In *Conference on Computer Vision and Pattern Recognition*, 2010. 4, 54, 55, 57, 74

[58] J. KITTLER, M. HATEF, R.P.W. DUIN, AND J. MATAS. **On Combining Classifiers**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(3):226–239, 1998. 2, 32

[59] G. KLEIN AND D. MURRAY. **Parallel Tracking and Mapping for Small AR Workspaces**. In *International Symposium on Mixed and Augmented Reality*, Nara, Japan, November 2007. 54, 57

[60] S. LAZEBNIK, C. SCHMID, AND J. PONCE. **Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories**. In *Conference on Computer Vision and Pattern Recognition*, 2006. 81

[61] B. LEIBE, A. LEONARDIS, AND B. SCHIELE. **Combined Object Categorization and Segmentation with an Implicit Shape Model**. In *European Conference on Computer Vision*, pages 17–32, 2004. 15

[62] V. LEPETIT AND P. FUA. **Monocular Model-Based 3D Tracking of Rigid Objects: A Survey**. *Foundations and Trends in Computer Graphics and Vision*, **1**(1):1–89, October 2005. 57

[63] V. LEPETIT AND P. FUA. **Keypoint Recognition using Randomized Trees**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**(9):1465–1479, September 2006. 2, 12, 13, 14, 18, 23, 26, 36, 51, 52, 60

[64] V. LEPETIT, P. LAGGER, AND P. FUA. **Randomized Trees for Real-Time Keypoint Recognition**. In *Conference on Computer Vision and Pattern Recognition*, San Diego, CA, June 2005. 53, 57, 58, 60

[65] J. LIEBELT, C. SCHMID, AND K. SCHERTLER. **Viewpoint-independent Object Class Detection using 3D Feature Maps**. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008. 77

[66] S. LIEBERKNECHT, S. BENHIMANE, P. MEIER, AND N. NAVAB. **A Dataset and Evaluation Methodology for Template-based Tracking Algorithms**. In *International Symposium on Mixed and Augmented Reality*, 2009. 26

[67] T. LIU, A. MOORE, A. GRAY, AND K. YANG. **An Investigation of Practical Approximate Nearest Neighbor Algorithms**. In *Neural Information Processing Systems*, 12 2004. 12

[68] D.G. LOWE. **Object Recognition from Local Scale-Invariant Features**. In *International Conference on Computer Vision*, pages 1150–1157, 1999. 14

[69] D.G. LOWE. **Distinctive Image Features from Scale-Invariant Keypoints**. *International Journal of Computer Vision*, **20**(2):91–110, 2004. 2, 10, 41, 51, 57, 76

[70] D.G. LOWE. **Demo Software: SIFT Keypoint Detector**, 2008. `http://www.cs.ubc.ca/~lowe/keypoints/`. 41, 45

[71] R. MARÉE, P. GEURTS, J. PIATER, AND L. WEHENKEL. **Random Subwindows for Robust Image Classification**. In *Conference on Computer Vision and Pattern Recognition*, 2005. 60

[72] J. MATAS, O. CHUM, U. MARTIN, AND T. PAJDLA. **Robust Wide Baseline Stereo from Maximally Stable Extremal Regions**. In *British Machine Vision Conference*, pages 384–393, September 2002. 2, 10

[73] J. MELTZER, M.-H. YANG, R. GUPTA, AND S. SOATTO. **Multiple View Feature Descriptors from Image Sequences via Kernel Principal Component Analysis**. In *European Conference on Computer Vision*, pages 215–227, May 2004. 57, 58

[74] K. MIKOLAJCZYK AND C. SCHMID. **An Affine Invariant Interest Point Detector**. In *European Conference on Computer Vision*, pages 128–142, 2002. 2, 10

[75] K. MIKOLAJCZYK AND C. SCHMID. **A Performance Evaluation of Local Descriptors**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**(10):1615–1630, 2004. 2, 57

[76] K. MIKOLAJCZYK, T. TUYTELAARS, C. SCHMID, A. ZISSERMAN, J. MATAS, F. SCHAFFALITZKY, T. KADIR, AND L. VAN GOOL. **A Comparison of Affine Region Detectors**. *International Journal of Computer Vision*, **65**(1/2):43–72, 2005. 2, 57

[77] F. MOOSMANN, B. TRIGGS, AND F. JURIE. **Fast Discriminative Visual Codebooks using Randomized Clustering Forests**. In *Neural Information Processing Systems*, pages 985–992. MIT Press, 2006. 12

[78] H. MORAVEC. *Robot Rover Visual Navigation*. UMI Research Press, 1981. 1

[79] P. MOREELS AND P. PERONA. **Evaluation of Features Detectors and Descriptors based on 3D Objects**. *International Journal of Computer Vision*, 2006. 45, 46, 48

[80] J. L. MUNDY. **Object Recognition in the Geometric Era: A Retrospective**. In J. PONCE, M. HEBERT, C. SCHMID, AND A. ZISSERMAN, editors, *Toward Category-Level Object Recognition*, **4170** of *Lecture Notes in Computer Science*, pages 3–28. Springer, 2006. 15

[81] J. L. MUNDY AND A. ZISSERMAN, editors. *Geometric Invariance in Computer Vision*. MIT Press, Cambridge, MA, USA, 1992. 2

[82] J. MUTCH AND D. G. LOWE. **Multiclass Object Recognition With Sparse, Localized Features**. In *Conference on Computer Vision and Pattern Recognition*, 2006. 4, 15, 77

[83] D. NISTER AND H. STEWENIUS. **Scalable Recognition With a Vocabulary Tree**. In *Conference on Computer Vision and Pattern Recognition*, 2006. 12

[84] S. OBDRZALEK AND J. MATAS. **Sub-linear Indexing for Large Scale Object Recognition**. In *British Machine Vision Conference*, 2005. 14, 52

[85] J. PHILBIN, O. CHUM, M. ISARD, J. SIVIC, AND A. ZISSERMAN. **Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases**. In *Conference on Computer Vision and Pattern Recognition*, 2008. 12

[86] J. PILET, V. LEPETIT, AND P. FUA. **Fast Non-Rigid Surface Detection, Registration and Realistic Augmentation**. *International Journal of Computer Vision*, **76**(2), February 2008. 10

[87] D. PRITCHARD AND W. HEIDRICH. **Cloth Motion Capture**. In *Eurographics*, **22(3)**, pages 263–271, September 2003. 58

[88] L.G. ROBERTS. **Machine Perception of 3-D Solids**. In *Optical and Electrooptical Information Processing*, pages 159–197, 1965. 1

[89] F. ROTHGANGER, S. LAZEBNIK, C. SCHMID, AND J. PONCE. **3D Object Modeling and Recognition Using Local Affine-Invariant Image Descriptors and Multi-View Spatial Constraints**. *International Journal of Computer Vision*, **66**(3):231–259, 2006. 14, 57

[90] M. SAHAMI, S. DUMAIS, D. HECKERMAN, AND E. HORVITZ. **A Bayesian Approach to Filtering Junk Email**. In *AAAI Workshop on Learning for Text Categorization*, July 1998. AAAI Technical Report WS-98-05. 24

[91] S. SAVARESE AND L. FEI-FEI. **3D Generic Object Categorization, Localization and Pose Estimation**. In *International Conference on Computer Vision*, 2007. iv, 77, 90, 91, 93

[92] C. SCHERRER, J. PILET, V. LEPETIT, AND P. FUA. **Souvenirs du Monde des Montagnes**. *Leonardo, special issue on ACM SIGGRAPH*, **42**(4):350–355, August 2009. 6, 8

[93] C. SCHMID AND R. MOHR. **Local Grayvalue Invariants for Image Retrieval**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19**(5):530–534, May 1997. 10

[94] S. SE, D. G. LOWE, AND J. LITTLE. **Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks**. *International Journal of Robotics Research*, **22**(8):735–758, 2002. 57

[95] T. SERRE, L. WOLF, AND T. POGGIO. **Object Recognition with Features Inspired by Visual Cortex**. In *Conference on Computer Vision and Pattern Recognition*, 2005. 15

[96] G. SHAKHNAROVICH. *Learning Task-Specific Similarity*. PhD thesis, MIT, 2005. 12

[97] E. SHECHTMAN AND M. IRANI. **Matching Local Self-Similarities across Images and Videos**. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007. 15

[98] J. SIVIC AND A. ZISSERMAN. **Video Google: Efficient Visual Search of Videos**. In *Toward Category-Level Object Recognition*, pages 127–144. Springer, 2006. 12

[99] I. SKRYPNYK AND D. G. LOWE. **Scene Modelling, Recognition and Tracking with Invariant Image Features**. In *International Symposium on Mixed and Augmented Reality*, pages 110–119, Arlington, VA, November 2004. 57

[100] C. STRECHA, R. FRANSENS, AND L. VAN GOOL. **Wide-baseline Stereo from Multiple Views: a Probabilistic Account**. In *Conference on Computer Vision and Pattern Recognition*, **2**, pages 552–559, 2004. 51

[101] C. STRECHA, R. FRANSENS, AND L. VAN GOOL. **Combined Depth and Outlier Estimation in Multi-View Stereo**. In *Conference on Computer Vision and Pattern Recognition*, 2006. 51

[102] H. SU, M. SUN, L. FEI-FEI, AND S. SAVARESE. **Learning a Dense Multi-view Representation for Detection, Viewpoint Classification and Synthesis of Object Categories**. In *International Conference on Computer Vision*, 2009. 4, 78

[103] M. PIETIKÄINEN T. OJALA AND T. MÄENPÄÄ. **Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**(7):971–987, 2002. 25

[104] A. THOMAS, V. FERRARI, B. LEIBE, T. TUYTELAARS, B. SCHIELE, AND L. VAN GOOL. **Towards Multi-View Object Class Detection**. In *Conference on Computer Vision and Pattern Recognition*, 2006. 77

[105] E. TOLA, V. LEPETIT, AND P. FUA. **DAISY: An Efficient Dense Descriptor Applied to Wide Baseline Stereo**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010. 81, 87

[106] A. TORRALBA, R. FERGUS, AND Y. WEISS. **Small Codes and Large Databases for Recognition**. In *Conference on Computer Vision and Pattern Recognition*, June 2008. 12

[107] A. TORRALBA, K.P. MURPHY, AND W.T. FREEMAN. **Sharing Visual Features for Multiclass and Multiview Object Detection**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**(5):854–869, May 2007. 15

[108] T. TUYTELAARS AND L. VAN GOOL. **Matching Widely Separated Views Based on Affine Invariant Regions**. *Computer Vision and Image Understanding*, **59**(1):61–85, 2004. 2, 10

[109] L. VACCHETTI, V. LEPETIT, AND P. FUA. **Stable Real-Time 3D Tracking Using Online and Offline Information**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**(10):1391–1391, 2004. 8

[110] P. VIOLA AND M. J. JONES. **Robust Real-Time Face Detection**. *International Journal of Computer Vision*, **57**(2):137–154, 2004. 55

[111] D. WAGNER, G. REITMAYR, A. MULLONI, T. DRUMMOND, AND D. SCHMALSTIEG. **Pose Tracking from Natural Features on Mobile Phones**. In *International Symposium on Mixed and Augmented Reality*, Cambridge, UK, September 2008. 8, 9, 11, 12, 51, 52, 57

[112] B. WILLIAMS, G. KLEIN, AND I. REID. **Real-time SLAM Relocalisation**. In *International Conference on Computer Vision*, 2007. 25

[113] S.A. WINDER AND M. BROWN. **Learning Local Image Descriptors**. In *Conference on Computer Vision and Pattern Recognition*, June 2007. 12

[114] J. WOLFE, W, D. MATHIS, C. W. SKLAIR, AND M. MAGEE. **The Perspective View of Three Points**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **13**(1):66–73, 1991. 101

[115] E. WOODS, P. MASON, AND M. BILLINGHURST. **MagicMouse: An Inexpensive 6-Degree-of-Freedom Mouse**. In *GRAPHITE*, pages 285–286. ACM, 2003. 8

[116] A. YILMAZ, O. JAVED, AND M. SHAH. **Object Tracking: A Survey**. *ACM Computing Surveys*, **38**(4):13, 2006. 1

[117] Q. YU, T. B. DINH, AND G. MEDIONI. **Online Tracking and Reacquisition using Co-trained Generative and Discriminative Trackers**. In *European Conference on Computer Vision*, pages 678–691, 2008. 55

[118] Q. YUAN, A. THANGALI, V. ABLAVSKY, AND B. SCLAROFF. **Multiplicative Kernels: Object Detection, Segmentation and Pose Estimation**. *Journal of Machine Learning Research*, June 2008. 77

[119] J. ZHANG, S.K. ZHOU, L. MCMILLAN, AND D. COMANICIU. **Joint Real-Time Object Detection and Pose Estimation Using Probabilistic Boosting Network**. *Journal of Machine Learning Research*, 2007. 77

[120] F. ZHENG AND G.I. WEBB. **A Comparative Study of Semi-naive Bayes Methods in Classification Learning**. In *Proceedings of the Fourth Australasian Data Mining Conference (AusDM05)*, pages 141–156, Sydney, 2005. 28

[121] A. ZISSERMAN, J. MUNDY, D. FORSYTH, J. LIU, N. PILLOW, C. ROTHWELL, AND S. UTCKE. **Class-based Grouping in Perspective Images**. In *Computer Vision*, pages 183–188, June 1995. 1

[122] E. ZITZLER AND S. KÜNZLI. **Indicator-Based Selection in Multiobjective Search**. In *Parallel Problem Solving from Nature (PPSN VIII)*, pages 832–842, 2004. 26

Mustafa Özuysal

EPFL IC ISIM CVLAB
BC 306 Station 14
CH-1015 Lausanne

mustafa.oezuysal@a3.epfl.ch
http://cvlab.epfl.ch/~oezuysal

EDUCATION

2005–Present **Ph.D. in Computer, Communication and Information Sciences**

Thesis Title: *Learning Pose Invariant and Covariant Classifiers from Image Sequences*
Computer Vision Laboratory (CVLAB), http://cvlab.epfl.ch
École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
Directed by Prof. Pascal Fua and Dr. Vincent Lepetit
Expected graduation: June, 2010

2002–2004 **M.Sc. in Electrical and Electronics Engineering**

Thesis Title: *Manual and Auto Calibration of Stereo Camera Systems*
METU-VISION, http://vision1.eee.metu.edu.tr/~vision
Middle East Technical University (METU), Ankara, Turkey
Directed by Prof. Uğur Halıcı

1998–2002 **B.Sc. in Electrical and Electronics Engineering**

Graduation Project: *A local positioning system using ultrasound transmitters*
Middle East Technical University, Ankara
Specialization: Computer & Telecommunication Systems

1995–1998 **Izmir Science High School**

WORK EXPERIENCE

2005–Present **Assistant, EPFL, Lausanne, Switzerland**

Supervised 4 Semester and 1 Master projects
C/Unix/Perl Course (One semester)
C++ Course (two semesters)
Foundations of Image Science Course (One semester)

2002-2004 **Teaching Assistant, METU, Ankara, Turkey**

Analog/Digital Electronics Laboratories (Four semesters)
Logic Design Recitation (One semester)
Circuit Theory Recitation (One semester)

2000, 2001 **Summer Internships, ASELSAN Inc., Ankara, Turkey**

Designed a CRC32 encoder/decoder on a Xilinx FPGA using VHDL, 2001

JOURNAL PUBLICATIONS

2010 **Fast Keypoint Recognition using Random Ferns**
M. Özuysal, M. Calonder, V. Lepetit, and P. Fua
IEEE Transactions on Pattern Analysis and Machine Intelligence

CONFERENCE PUBLICATIONS

2010 **Making Action Recognition Robust to Occlusions and Viewpoint Changes**
D. Weinland, M. Özuysal, and P. Fua
European Conference on Computer Vision, To appear

Combining Geometric and Appearance Priors for Robust Homography Estimation
E. Serradell, M. Özuysal, V. Lepetit, P. Fua, and F. Moreno-Noguer
European Conference on Computer Vision, To appear

2009 **Pose Estimation for Category Specific Multiview Object Localization**
M. Özuysal, V. Lepetit, and P. Fua
IEEE Conference on Computer Vision and Pattern Recognition

2007 **Fast Keypoint Recognition in Ten Lines of Code**
M. Özuysal, P. Fua, and V. Lepetit
IEEE Conference on Computer Vision and Pattern Recognition

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

AWARDS

2009 Co-inventor for the "Ferns" based object detection software sold by EPFL

2005–2006 Scholarship of Doctoral School of Computer & Communication Sciences, EPFL

1998 Bülent Kerim Altay Award for outstanding GPA

1998-2002 Scholarship of the Middle East Technical University
Awarded for ranking in the top 100 in the nationwide university entry examination

1997 First prize in the computer branch of high school project contest
Organized by TÜBİTAK (National Science Foundation of Turkey)
Project Title: *Finding Roots of Mathematical Expressions using Genetic Algorithms*

SKILLS

Programming C/C++, Matlab, Python, OpenGL, LaTeX

OS Linux, Windows XP, Mac OS X

Language English (Fluent), French (Basic)