UNIVERSITAT POLITÈCNICA DE CATALUNYA

Programa de Doctorat:

AUTOMÀTICA, ROBÒTICA I VISIÓ

Tesi Doctoral

EFFICIENT APPROACHES FOR OBJECT CLASS DETECTION

Michael Alejandro Villamizar Vergel

Directors:

Alberto Sanfeliu

Juan Andrade Cetto

Maig de 2012

To my parents María Helena and Reinaldo, and
my brothers José and Cristian.

Agradecimientos

Me gustaría agradecer profundamente al Prof. Alberto Sanfeliu por guiarme a través del doctorado y su apoyo en momentos difíciles. Sus preciados consejos, su continua ayuda y constante motivación han dado como fruto esta tesis. Me gustaría agradecer también al Dr. Juan Andrade por introducirme en el mundo de la visión por ordenador y por su valiosa ayuda en mis primeros pasos por este campo de investigación. Al Dr. Francesc Moreno por ser una parte importante en el último trayecto de mi doctorado. Su colaboración y dedicación conllevó a nuevas ideas y excelentes resultados, gracias también por aquellas jugadas magistrales impartidas los jueves en las tardes en nuestra sesión semanal de fútbol.

To Prof. Luc Van Gool and Helmut Grabner for their hospitality and their wisdom council during my research visit to K.U Leuven and ETHZ.

Por otro lado, me gustaría dar gracias a mis amigos Anaís, Dianita, Ernesto, Javier N., Javier V., Jorge, Leonel, Lidia, María, Nicolás, Oscar y Rafa, por su cálida amistad, sus consejos y la compañía en muchas fiestas de la noche barcelonesa. A mis amigos del despacho 7: Agustín, Attila, Bernat, Diego, Farzad, Mauricio, Oscar, Stephan, Valeria y Vanesa, por su grata compañía y comprensión durante el desarrollo de mi doctorado, especialmente, por mi música a niveles exasperantes, mis palabrotas y mis pies sobre la mesa. En general, muchas gracias a mis amigos en el Institut de Robòtica i Informàtica Industrial (CSIC-UPC). Quiero dar gracias también a aquellas personas cercanas y de lejos cuyos consejos y comentarios han contribuido positivamente en mi vida personal y académica. A special mention to Kade for sharing with me her time, her affection, her space and her smile.

Es importante dar mis agradecimientos también a la Universitat Politècnica de Catalunya y a los proyectos URUS (No. IST FP6 STREP 045062) y MIPRCV Consolider Ingenio 2010 por el soporte económico durante estos años de estudio e investigación. De igual forma agradecer al Ministerio Español de Educación por el soporte económico recibido durante mi estancia de estudios en K.U Leuven.

Finalmente, mi gratitud eterna a mi querida familia, en especial, a mis padres Reinaldo y María Helena por su inagotable cariño, su apoyo férreo y su ferviente labor como padres y educadores. A mis hermanos, José y Cristian por su apreciado afecto y colaboración. Al resto de familia muchas gracias por su inmenso e incondicional amor. Este logro personal es también vuestro. Gracias.

Abstract

Computer vision and more specifically object recognition have demonstrated in recent years an impressive progress that has led to the emergence of new and useful technologies that facilitate daily activities and improve some industrial processes. Currently, we can find algorithms for object recognition in computers, video cameras, mobile phones, tablets or websites, for the accomplishment of specific tasks such as face detection, gesture and scene recognition, detection of pedestrians, augmented reality, etc.

However, these applications are still open problems that each year receive more attention in the computer vision community. This is demonstrated by the fact that hundreds of articles addressing these problems are published in international conferences and journals annually. In a broader view, recent work attempts to improve the performance of classifiers, to face new and more challenging problems of detection and to increase the computational efficiency of the resulting algorithms in order to be implemented commercially in diverse electronic devices. Although nowadays there are robust and reliable approaches for detecting objects, most of these methods have a high computational cost that make impossible their application for real-time tasks. In particular, the computational cost and performance of any recognition system is determined by the type of features, the method of recognition and the methodology used for localizing objects within images. The main objective of these methods is to produce not only effective but also efficient detection systems.

Through this dissertation different approaches are presented for addressing efficiently and discriminatively the detection of objects in diverse and difficult imaging conditions. Each one of the proposed approaches are especially designed and focus on different detection problems, such as object categorization, detection under rotations in the plane or the detection of objects from multiple views. The proposed methods combine several ideas and techniques for obtaining object detectors that are both highly discriminative and efficient. This is demonstrated experimentally in several state-of-the-art databases where our results are competitive with other recent and successful methods. In particular, this dissertation studies and develops fast features, learning algorithms, methods for reducing the computational cost of the classifiers and integral image representations for speeding up feature computation.

Resumen

La visión por computador y más específicamente el reconocimiento de objetos han demostrado en los últimos años un impresionante progreso que ha llevado a la aparición de nuevas y útiles tecnologías que facilitan nuestras actividades diarias y mejoran ciertos procesos industriales. Actualmente, nosotros podemos encontrar algoritmos para el reconocimiento de objetos en computadores, videocámaras, teléfonos móviles, tablets o sitios web para la realización de ciertas tareas específicas tales como la detección de caras, el reconocimiento de gestos y escenas, la detección de peatones, la realidad aumentada, etc.

No obstante, estas aplicaciones siguen siendo problemas abiertos que cada año reciben más atención por parte de la comunidad de visión por computador. Esto se demuestra por el hecho de que cientos de artículos abordando estos problemas son publicados en congresos internacionales y revistas anualmente. Desde una perspectiva general, los trabajos más recientes intentan mejorar el desempeño de clasificadores, hacer frente a nuevos y más desafiantes problemas de detección, y a aumentar la eficiencia computacional de los algoritmos resultantes con el objetivo de ser implementados comercialmente en diversos dispositivos electrónicos. Aunque actualmente, existen enfoques robustos y confiables para la detección de objetos, la mayoría de estos métodos tienen un alto coste computacional que hacen imposible su aplicación en tareas en tiempo real. En particular, el coste computacional y el desempeño de cualquier sistema de reconocimiento está determinado por el tipo de características, método de reconocimiento y la metodología utilizada para localizar los objetos dentro de las imágenes. El principal objetivo de estos métodos es obtener sistemas de detección eficaces pero también eficientes.

A través de esta tesis diferentes enfoques son presentados para abordar de manera eficiente y discriminante la detección de objetos en condiciones de imagen diversas y difíciles. Cada uno de los enfoques propuestos ha sido especialmente diseñado y enfocado para la detección de objetos en circunstancias distintas, tales como la categorización de objetos, la detección bajo rotaciones en el plano o la detección de objetos a partir de múltiples vistas. Los métodos propuestos combinan varias ideas y técnicas para la obtención de detectores de objetos que son tanto altamente discriminantes como eficientes. Esto se demuestra experimentalmente en varias bases de datos del estado del arte donde los resultados alcanzados son competitivos al ser contrastados con otros métodos recientes. En concreto, esta tesis estudia y desarrolla características rápidas, algoritmos de aprendizaje, métodos para reducir el coste computacional de los clasificadores y representaciones de imagen integral que permiten un mejor cálculo de las características.

Resum

La visiò per ordinador, i més específicament el reconeixement d'objetes, han demostrat en els últims anys un impresionant progrès que ha portat a l'aparició de noves i útils tecnologies que faciliten les nostres activitats diàries i que milloren certs processos industrials. Actualment, nosaltres podem trobar algoritmes per al reconeixement d'objectes en ordinadors, videocàmeres, telèfons mòbils, tablets o pàgines webs per a la realització de certes tasques concretes tals com la detecció de cares, el reconeixement de gestos i escenes, la detecció de vianants, la realitat augmentada, etc.

No obstant, aquestes aplicaciones continuen sent problemes oberts que cada any reben mès atenció per part de la comunitat de visió per computador. Això ve demostrat pel fet que centenars d'artícles tractant aquests problemes sòn publicats en congressos internacionals i revistes anualment. Des d'una perspectiva general, els treballs més recents intenten millorar el desenvolupament dels classificadors, fer front a nous i més desafiants problemes desafiants de detecció, i augmentar l'eficiència computacional dels algoritmes resultants amb l'objectiu de ser implementats comercialment en diferents dispositius electrònics. Tot i que actualment, existeixen enfocs fiables i robustos per a la detecció d'objectes, la majoria d'aquests mètodes tenen un alt cost computacional i fan impossible la seva aplicació en tasques en temps real. En particular, el cost computacional i el desenvolupament de qualsevol sistema de reconeixement està determinat pel tipus característica, mètode de reconeixement i la metodologia utilitzada per a localitzar objectes dins de les imatges. El principal objectiu d'aquests mètodes és obtenir sistemes de detecció eficaç però també eficients.

A través d'aquest tesi, diferents enfocaments sòn presentats per tal de tractar de manera eficient i discriminant la detecció d'objectes en condicions d'imatge diverses i difícils. Cadascun dels enfocaments proposats ha estat especialment dissenyat i enfocat per a la detecció d'objectes en diferents circumstàncies, tals com la categorització d'objectes, la detecció sota rotacions en el pla o la detecció d'objectes a partir de múltibles vistes. Els mètodes proposats combinen diferents idees i tècniques per a l'obtenció de detectors d'objectes que són tant àltament discriminants com eficients. Això es demostra experimentalment en diferents bases de dades del estat de l'art on els resultats assolits són competitius al ser contrastats amb altres mètodes recents.

En concret, aquesta tesi estudia i desenvolupa característiques ràpides, algoritmes d'aprenentatge, métodes per a reduir el cost computacional dels classificadors i representacions d'imatge integral que permeten un millor càlcul de les característiques.

Contents

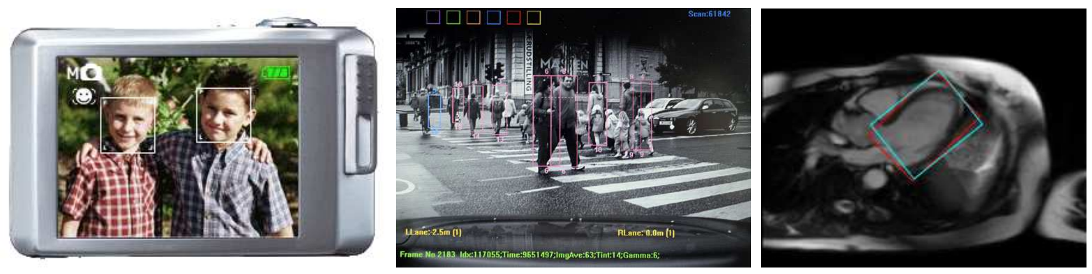
List of Figures

Chapter 1

Introduction

In the last years, computer vision has matured considerably and has achieved remarkable results in some fields such as object recognition, medical imaging, object tracking, scene reconstruction, etc. Nowadays, it is possible to find computer vision algorithms in diverse electronic devices that simplify our daily activities. For instance, personal cameras, webcams or cameras incorporated into cell phones have algorithms for face detection –Fig. 1.1(a)–, gesture recognition or landscape classification that improve the quality of our photos. These algorithms run in real-time and are robust to diverse scene conditions and object appearance changes. Other interesting and useful application is pedestrian detection in cars, see Fig. 1.1(b). In this case, the algorithms are incorporated in cars with the aim of informing us about the existence of people on the streets and to alert the driver about possible dangerous situations. In fact, the range of computer vision applications is wide. Medical image analysis –Fig. 1.1(c)–, surveillance in buildings and airports, car and robot navigation, human-machine interaction, image retrieval from large internet databases are some examples of current trends. Computer vision is, therefore, a very active research field that produces each year a very large number of articles published in diverse international conferences, journals and books, showing the great interest in this topic.

One important research topic inside the computer vision community and the main topic of this dissertation is object recognition. It consists on localizing and identifying specific objects or categories inside images or video sequences. Despite the great advances in the recent years, this is still a challenging problem, mainly because the appearance of an object in an image can vary significantly from one instance to another.



(a) (b) (c)

Figure 1.1: Computer vision applications. (a) Face detection in digital cameras [www.gecameras.com.au]. (b) Pedestrian detection in cars [www.volvo.com]. (c) Left ventricle detection in 2D MRI [www.umiacs.umd.edu].

This variation is caused by various factors such as intra-class variation, camera viewpoint change, varying illumination conditions, scaling or deformations. Some example images illustrating this difficult problem are shown in Fig. 1.2. The images correspond to some public databases containing some object categories: motorbikes, cars or human faces. We see in these images different issues that difficult the learning and subsequent recognition of objects. Intra-class variability, for instance, is observed at the top row where different motorbike models are shown. At the second row, the object car appears at multiple scales and locations, whereas at the third row, the objects –frontal faces– suffer varying illumination conditions. Finally, the bottom row shows an object category seen from multiple views. In this case, both the size and appearance change according to the camera viewpoint.

The purpose of this dissertation is to propose new approaches to detect –localizing and recognizing– objects in different scenarios and varying imaging conditions using efficient techniques and algorithms for this goal. Through this dissertation, we cope with different object recognition problems by means of simple, time-saving and robust methods that achieve competitive detection rates against the state of the art with the advantage of a more straightforward and efficient computation. More precisely, we propose efficient methods by combining sinergically diverse techniques and approaches for speeding up the testing of object classifiers over images.

In the rest of this chapter, we briefly state the main contributions, overview and the derived publications, which had become the contents of the rest of monograph, set forth chapter by chapter.

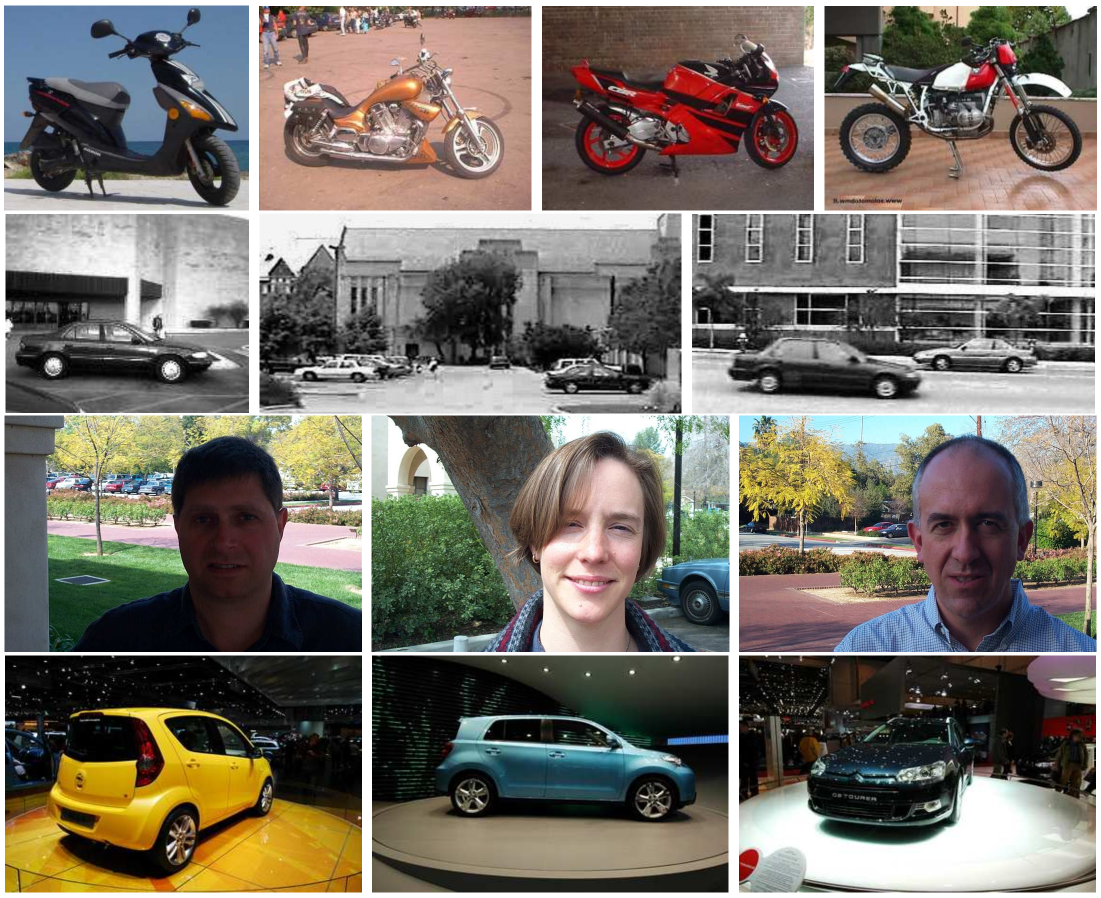


Figure 1.2: Object categories. Sample images containing four different categories: motorbikes –TUD motorbike dataset [25]–, cars –UIUC car dataset [1]–, faces –Caltech face dataset [17]– and multi-view car dataset –EPFL car dataset [72]–. The appearance of objects is affected by lighting, scale, intra-class and viewpoint changes.

1.1 Main Contributions

The main contributions of the dissertation may be summarized as follows:

1. A new approach for computing Haar-like features by means of an approximation of Steerable Filters is proposed. This yields a fast manner to extract image features such as edges or stripes at any given orientation. This approach overcomes the drawbacks of building additional integral images or more complex oriented features. Besides, our method is suitable for the description and detection of objects under rotations in the image plane.

2. We propose a method called Boosted Random Ferns –BRFs– for the computation of efficient and effective object classifiers. More precisely, this method combines fast features –Random Ferns calculated on histograms of oriented gradients– and AdaBoost in order to extract discriminative features and to obtain robust binary classifiers that allow to localize and recognize object categories inside images in a few seconds.

3. An efficient and robust method for detecting objects that may have rotations in the image plane is proposed. This method uses a two-step approach consisting of an orientation estimator and an object classifier which are efficiently calculated using Random Ferns –RFs–. This approach reduces drastically the computational cost since the object classifier is only evaluated on the reduced set of hypotheses –image locations and orientations– given by the estimator.

4. A detection method for recognizing multiple object categories in images is presented. Particularly, we propose to build discriminative object classifiers –BRFs– independently but sharing the same features –RFs– in order to reduce the inherent and expensive cost of testing several classifiers by separated. The proposed method is highly efficient since feature computation is common for all detectors and is done just once.

5. We present a 3D object detection approach that integrates fast features –RFs–, a pose estimator and a set of pose-specific classifiers which are trained via BRFs. This approach is very efficient and achieves high detection rates on some state-of-the-art datasets for detecting cars from multiples views. Besides, the estimator we propose, called Hough-RFs, is based on Random Ferns and the Hough transform in order to cope efficiently with size variations across object views. This estimator determines object candidates in images that are then verified by the set of pose-specific classifiers.

1.2 Thesis Overview

This monograph is organized according to the following chapters:

- **Chapter 2** reviews some techniques that are used through this thesis as ingredients for the computation and evaluation of efficient object classifiers. The techniques mentioned in this chapter refer mainly to learning algorithms, efficient features and integral image representations.

- **Chapter 3** presents an efficient method to recognize objects under arbitrary rotations in the image plane. Particularly, the proposed method combines fast and steerable features with a gradient-based estimator which gives hypotheses about the object orientation. This estimator allows to rotate the object classifier, trained discriminatively via AdaBoost [24], and reduces the high computational cost of evaluating the classifier for a vast range of orientations, or having a large set of classifiers trained at multiple orientations. This chapter also introduces Haar-like features which can be steered at any orientation using the principle of steerable filters [22].

- **Chapter 4** use Boosted Random Ferns –BRFs– to compute efficient and discriminative object classifiers. BRFs uses AdaBoost [24] to compute and combine, in an iterative and supervised process, the most relevant Random Ferns [64] over local histograms of oriented gradients –HOGs–. The chapter finishes with an extensive validation on standard object datasets, where the proposed BRFs yield high detection rates in spite of difficult imaging conditions such as occlusions, lighting changes and intra-class variations.

- **Chapter 5** introduces a robust and efficient approach for object detection and categorization under in-plane rotations. The proposed method makes use of BRFs –Chapter 4– and the two-step detection approach presented in Chapter 3. More concretely, the first step is an estimator that is tested in images to yield potential hypotheses about the object location and orientation. The second step is a very discriminative classifier –BRFs– which verifies the hypotheses given by the estimator. This approach reduces the number of false positives and speeds up the detection phase since the object classifier is tested only at specific poses. The chapter is concluded with an experimental validation over a novel and challenging dataset including motorbikes having planar rotations.

- **Chapter 6** focuses on the detection of multiple object categories in images in an efficient way. To this end, this chapter describes a new method to compute multiple object classifiers –BRFs– independently but sharing the same features –Random Ferns–. The benefit lies on feature computation, the most computationally expensive process, which is done once and is independent of the number of object categories. Therefore, this approach speeds up feature computation in runtime and reduces also the overall number of features. More precisely, the proposed approach learns a specific and discriminative configuration of features for each object category by means of AdaBoost [24]. Although such configurations differ, they share the same features.

- **Chapter 7** describes a novel 3D object detection approach. The proposed method integrates fast features –Chapter 4–, sharing features –Chapter 6– and a two-step approach consisting of a pose estimator and a set of pose-specific classifiers to deal with the localization of 3D objects from multiple views. In contrast to the estimator introduced in Chapter 5, this chapter proposes a new and efficient estimator based on the Hough transform. This estimator is used to determine object/pose hypotheses which are then verified by a set of pose-specific classifiers. The result is a very efficient and discriminative method for object detection that achieves high detection rates in public datasets while keeping efficiency.

- **Chapter 8** summarizes the dissertation and also provides the future work.

1.3 Derived Publications

The derived publications during the PhD are listed below. They correspond to articles submitted to relevant international and national journals and conferences.

1. M. Villamizar, A. Garrell, A. Sanfeliu and F. Moreno-Noguer. Online Human-Assisted Learning Using Random Ferns. International Conference on Pattern Recognition (ICPR), 2012 [submitted].

2. M. Villamizar, J. Andrade-Cetto, A. Sanfeliu and F. Moreno-Noguer. Bootstrapping Boosted Random Ferns for Discriminative and Efficient Object Classification. Pattern Recognition, 2012.

3. M. Villamizar, H. Grabner,J. Andrade-Cetto, A. Sanfeliu, L. Van Gool and F. Moreno-Noguer. Efficient 3D Object Detection using Multiple Pose-specific Classifiers. British Machine Vision Conference (BMVC), 2011 [oral].

4. M. Villamizar, F. Moreno-Noguer, J. Andrade-Cetto, A. Sanfeliu. Detection Performance Evaluation of Boosted Random Ferns. Iberian Conference on Pattern Recognition and Image Analysis (IBPRIA), 2011.

5. M. Villamizar, F. Moreno-Noguer, J. Andrade-Cetto, A. Sanfeliu. Efficient Rotation Invariant Object Detection using Boosted Random Ferns. Conference in Computer Vision and Pattern Recognition (CVPR), 2010.

6. J. Scandaliaris, M. Villamizar, A. Sanfeliu. Comparative Analysis for Detecting Objects Under Cast Shadows in Video Images International Conference on Pattern Recognition (ICPR), 2010.

7. M. Villamizar, F. Moreno-Noguer, J. Andrade-Cetto, A. Sanfeliu. Shared Random Ferns for Efficient Detection of Multiple Categories. International Conference on Pattern Recognition (ICPR), 2010.

8. M. Villamizar, A. Sanfeliu and J. Andrade-Cetto. Local boosted features for pedestrian detection. Iberian Conference on Pattern Recognition and Image Analysis (IBPRIA), 2009.

9. M. Villamizar, J. Scandaliaris, A. Sanfeliu and J. Andrade-Cetto. Combining color-based invariant gradient detector with HoG descriptors for robust image detection in scenes under cast shadows. International Conference on Robotics and Automation (ICRA), 2009.

10. J. Scandaliaris, M. Villamizar, J. Andrade-Cetto and A. Sanfeliu. Robust color contour object detection invariant to shadows. Iberoamerican Congress on Pattern Recognition (CIARP), 2007.

11. M. Villamizar, A. Sanfeliu and J. Andrade-Cetto. Unidimensional multiscale local features for object detection under rotation and mild occlusions. Iberian Conference on Pattern Recognition and Image Analysis (IBPRIA), 2007.

12. J. Andrade-Cetto and M. Villamizar. Object recognition. In Wiley Encyclopedia of Electrical and Electronics Engineering, 1–28. John Wiley and Sons, 2007.

13. M. Villamizar, A. Sanfeliu and J. Andrade-Cetto. Computation of rotation local invariant features using the integral image for real time object detection. International Conference on Pattern Recognition (ICPR), 2006.

14. M. Villamizar, A. Sanfeliu and J. Andrade-Cetto. Orientation invariant features for multiclass object recognition. Iberoamerican Congress on Pattern Recognition (CIARP), 2006.

15. M. Villamizar, A. Sanfeliu and J. Andrade-Cetto. Computo de características invariantes a la rotación para el reconocimiento de distintas clases de objetos. Jornadas de Automática, 2006.

Chapter 2

Background

In this chapter some techniques are described in order to contextualize the contributions of this dissertation. These techniques have been used in the past years for constructing successful approaches for the detection of objects in images. In this thesis, we propose detection approaches that use and extend some of these techniques with the aim of computing effective and efficient methods for recognizing and localizing object categories in challenging scenes. The addressed techniques are fast and discriminative features such as Random Ferns [64] and Haar-like features [98]; boosting algorithms [23, 24]; and image representations for speeding up feature computation.

2.1 Random Ferns

Random Ferns (RFs), proposed by Ozuysal *et al.* [64], were designed for very fast and effective keypoint matching, where each RF consists of a set of random and simple binary features that are calculated over pixel intensities in the neighborhood of interest points. By means of RFs, a semi-naïve classifier can be constructed to recognize keypoints very fast in spite of having image distortions such as rotations in the image plane.

More specifically, given a set of keypoint classes which are built offline by introducing image distortions on keypoint patches, the objective is to assign to a test patch, surrounding a detected keypoint, the most likely keypoint class. This is done, firstly, by modeling the class conditional probabilities of a set of N binary features in a training phase, and then by calculating, in run-time, these binary features f in the test patch.

The class for the test keypoint $\hat{C}$ is selected as that class with highest probability given the feature outputs. This can be formulated as:

$$\hat{C}(x) = \arg\max_{c_i} P(C = c_i | f_1(x), f_2(x), ...f_N(x)), \quad i = 1, 2, ...N_k, \tag{2.1}$$

where x is the test keypoint, N_k is the number of keypoint classes, and C is the random class variable.

Using the Bayes rule and assuming that all classes have the same prior probabilities $P(C)$, and removing the evidence factor $P(f_1(x), f_2(x), ...f_N(x))$ since it does not depend on the class, the keypoint classification can be written by means of its likelihood:

$$\hat{C}(x) = \arg\max_{c_i} P(f_1(x), f_2(x), ...f_N(x) | C = c_i), \quad i = 1, 2, ...N_k. \tag{2.2}$$

Since computing the complete joint probability for a large feature set $(P(f_1, f_2, ...f_N))$ is not feasible, it is split into R subsets $(\mathcal{F}_r = \{f_1^r, f_2^r, .., f_M^r\})$, with $M = N/R$ and $r = 1, 2, .., R$. These feature subsets are known as Ferns, and assuming they are independent, this joint probability conditioned to classes is calculated as:

$$P(f_1(x), f_2(x), ...f_N(x) | C = c_i) = \prod_{r=1}^{R} P(\mathcal{F}_r(x) | C = c_i), \tag{2.3}$$

and the class of the test keypoint is then selected by

$$\hat{C}(x) = \arg\max_{c_i} \prod_{r=1}^{R} P(\mathcal{F}_r(x) | C = c_i), \quad i = 1, 2, .., N_k. \tag{2.4}$$

We see the last equation is a semi-naïve classifier because it does not compute neither the complete joint feature probability nor a naïve classifier where independence among all features is assumed. By contrast, it models only some dependencies among features.

Random Ferns have proved to be competitive with the SIFT descriptor [49] but demanding less computation time because the processing steps to achieve insensitivity to image distortions are removed. Instead, Random Ferns are trained with a synthesized set of feature images in order to obtain robustness to viewpoint and lighting changes. In comparison to Randomized Trees [45], the presented approach is simpler, more powerful and more scalable in terms of the number of classes it can handle. The authors have

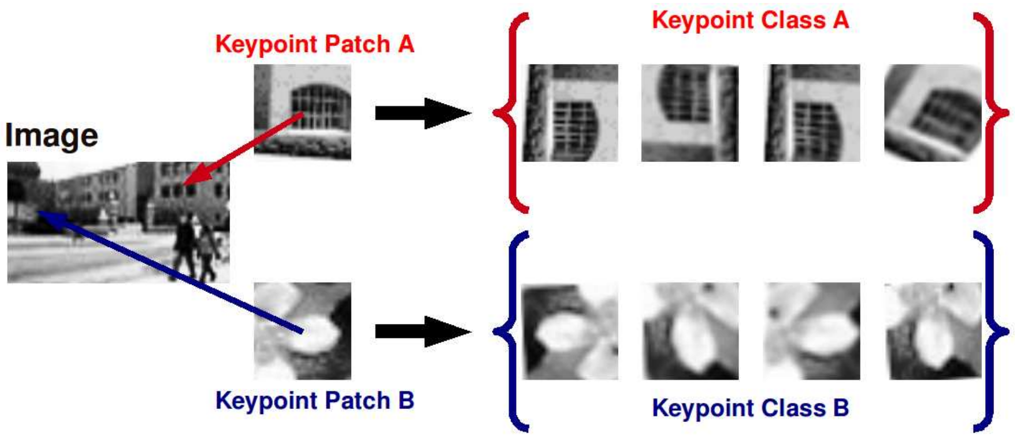
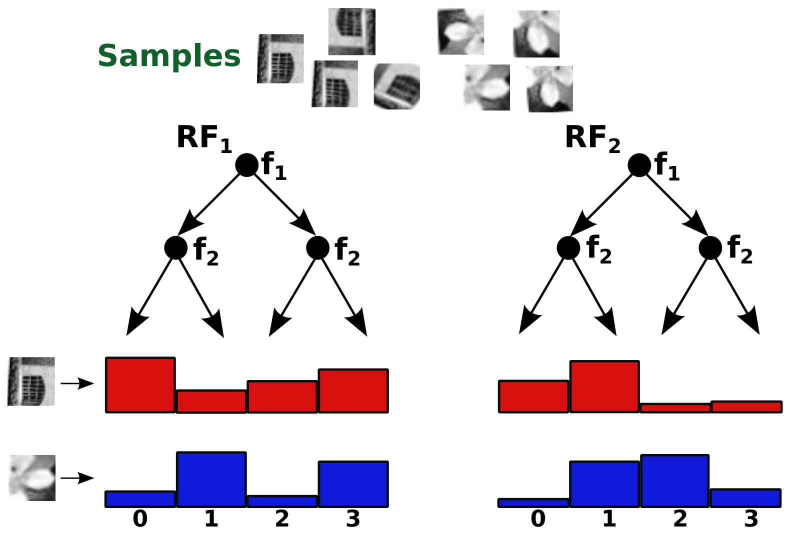


Figure 2.1: Construction of keypoints classes. Given detected keypoints in the input image, the keypoints classes are constructed by synthesizing a set of new images by means of applying affine deformations to the keypoints patches.

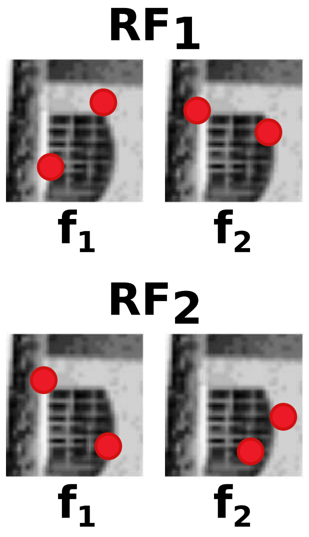
reported that with M around 10 and R between 30 and 50, good recognition rates are achieved for keypoint classification. Furthermore, the method is so simple that it can be implemented in ten lines of code [64]. Because of its efficiency, this method has been extended to other computer vision tasks such as object detection [91, 92, 93] and tracking [37].

Training the Random Ferns. To train the RFs, first, the keypoint classes are constructed given a set of initial keypoints that are extracted from images using an interest point detector [31, 81, 53, 54]. For each keypoint patch a set of new images are synthesized by introducing image distortions such as affine deformations to the initial patches. These collections of images represent the set of possible keypoint appearances under different view conditions. To illustrate this procedure, Fig. 2.1 shows how two keypoint classes are generated after adding image deformations to the original keypoint patches.

The training of RFs is exemplified in Fig. 2.2 where two RFs are trained for classifying two keypoint classes –image patches–. Training samples are evaluated over the Random Ferns in order to compute the distributions of Fern observations for each class. In this example, two binary features per Fern are considered ($M = 2$). These binary features are intensity comparisons between two random pixels –frame (b) in the figure–.



(a) Random Ferns



(b) FernFeatures

Figure 2.2: Training the Random Ferns. (a) Random Ferns are trained for the classification of two keypoints classes. Each RF consists of two binary features whose co-occurrence determines the the distribution of Fern observations for each class. (b) Features are basically random pixel comparisons over the keypoint patches.

Although the features are chosen randomly during training, they remain fixed in the classification phase. Given that the output of each feature is a binary value, the size of the Fern distribution is 2^M, with M the number of binary features per Fern.

Random Ferns, unlike Randomized Trees, do not have a hierarchical structure and features at the same level are equal. This means that the Fern distributions only measure the co-occurrence of features belonging to the Fern in question. Each bin in the histogram –distribution– represents a specific feature co-occurrence whose value is the probability of such output.

Keypoint Classification. Keypoint classification proceeds as follows: first, the Random Ferns are calculated in the test keypoint –patch– so that feature co-occurrence can be computed. Then, given the Fern outputs and the Fern distributions, the Fern probabilities for each class are combined into a final distribution. The class of the test keypoint is finally selected as that class with highest probability in the resulting class distribution, refer to Eq. 2.4.

2.2 Discrete AdaBoost Algorithm

Discrete AdaBoost –DAB– is a machine learning algorithm used to improving the performance of any given learning algorithm [24]. In general, it allows to construct a robust or strong classifier using a set of weak classifiers. DAB improves the performance of the strong classifier by the selection and weighted combination of discriminative features in an iterative process where the weights of training samples are updated in each iteration. AdaBoost has some interesting properties such as having a good generalization and proved convergence provided all weak hypotheses have less than 50% classification error.

Formally, given a set of training samples $(x_1, y_1)...(x_n, y_n)...(x_N, y_N)$, where x_n refers to one sample from the sample space X, and y_n is a class label, $Y = \{+1, -1\}$, that indicates the positive and negative classes, respectively, DAB extracts at each iteration t a weak classifier h that best discriminates positive from negative training samples. The boosted combination of weak classifiers yields an efficient and robust classifier which is commonly known as the strong classifier H. Each weak classifier at iteration t maps the samples to a binary space, $h_t : X \rightarrow \{+1, -1\}$, and gives a classification error ϵ_t that is calculated as the sum of weights of the misclassified samples under the current weight distribution D_t. This error can be computed by

$$\epsilon_t = \sum_{n:h_t(x_n) \neq y_n} D_t(x_n), \quad n = 1, 2, .., N. \tag{2.5}$$

where N is the number of training samples.

For every iteration, the algorithm selects the most discriminative weak classifier and its contribution α_t in classifying the entire training set as a function of the classification error ϵ_t,

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right). \tag{2.6}$$

The algorithm also updates the distribution of weights over the training samples. Initially, all weights are set equally, but on each round, the weights of misclassified samples are increased so that the algorithm is forced to focus on such hard samples in the training set that were previously missed by the classifier.

Algorithm 1 Discrete AdaBoost.

```
1: Given a number $T$ of weak classifiers, a pool containing $K$ weak classifiers, and $N$
   samples $(x_1, y_1)...(x_n, y_n)...(x_N, y_N)$, where $y_n \in \{+1, -1\}$ is the label for positive
   and negative classes, respectively.
2: Initialize sample weights $D_1(x_n) = \frac{1}{N}$, with $n = 1, 2, .., N$.
3: for $t = 1$ to $T$ do
4:    for $k = 1$ to $K$ do
5:       Compute the weak classifier $h_k : X \to \{+1, -1\}$.
6:       Calculate the classification error $\epsilon_k$.
         $\epsilon_k = \sum_{n:h_k(x_n) \neq y_n} D_t(x_n)$
7:    end for
8:    Select the weak classifier $h_t$ that minimizes $e$.
9:    Calculate the weight of the weak classifier.
      $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$
10:   Update the sample weights.
      $D_{t+1}(x_n) = \frac{D_t(x_n) \exp[-y_n h_t(x_n)]}{\sum_{n=1}^{N} D_t(x_n) \exp[-y_n h_t(x_n)]}$
11: end for
12: Final strong classifier.
    $H(x) = sign \left( \sum_{t=1}^{T} \alpha_t h_t(x) - \beta \right)$
```

Finally, the strong classifier formed by T weak classifiers is defined as:

$$H(x) = sign \left(\sum_{t=1}^{T} \alpha_t h_t(x) - \beta \right), \tag{2.7}$$

where β refers to the classifier threshold. This classifier H is a weighted majority vote of the T weak hypotheses. Pseudocode for an AdaBoost implementation is described in Alg. 1.

2.3 Real AdaBoost Algorithm

Real AdaBoost –RAB– proposed by Schapire and Singer [73] is an extension to the discrete AdaBoost algorithm –DAB–. Unlike its discrete version whose weak classifiers yield Boolean predictions, RAB deals with confidence-rated weak classifiers. Similarly to DAB, Real AdaBoost is a machine learning algorithm used for improving the performance of any given learning algorithm. It computes a robust classifier as the weighted combination, in an iterative and supervised procedure, of weak classifiers. The algorithm also updates, in each iteration, a distribution of weights over the training samples

Algorithm 2 Real AdaBoost algorithm.

```
1: Given a number $T$ of weak classifiers, a pool containing $K$ weak classifiers, and a set of $N$ samples $(x_1, y_1)..(x_n, y_n)..(x_N, y_N)$, where $y_n \in \{+1, -1\}$ is the label for positive and negative classes, respectively.
2: Initialize sample weights $D_1(x_n) = 1/N$, with $n = 1, 2, .., N$.
3: for $t = 1$ to $T$ do
4:   for $k = 1$ to $K$ do
5:     For the weak classifier $h_k$, the space $X$ is divided into $J$ disjoint blocks.
       $X = \{X_1, X_2, .., X_j, .., X_J\}$
6:     Calculate the probability distributions for positive and negative samples.
       $W^j_{\pm 1} = P(x_n \in X_j, y_n = \pm 1) = \sum_{n:x_n \in X_j \wedge y_n = \pm 1} D_t(x_n)$
7:     Compute the weak classifier $h_k$.
       $h_t(x) = \frac{1}{2} \log \frac{P(W^j_{+1} + \epsilon)}{P(W^j_{-1} + \epsilon)}$
8:     Calculate the Bhattacharyya coefficient.
       $Q = 2 \sum_{j=1}^{J} \sqrt{W^j_{+1} W^j_{-1}}$
9:   end for
10:  Select the weak classifier $h_t$ that minimizes $Q$.
11:  Update the sample weights.
     $D_{t+1}(x_n) = \frac{D_t(x_n) \exp[-y_n h_t(x_n)]}{\sum_{n=1}^{N} D_t(x_n) \exp[-y_n h_t(x_n)]}$
12: end for
13: Final strong classifier.
    $H(x) = sign\left(\sum_{t=1}^{T} h_t(x) - \beta\right)$
```

in order to focus the algorithm's effort into the hard samples, that is, the misclassified samples at previous iterations.

In contrast to discrete AdaBoost, in RAB each weak classifier maps the sample space X to real-valued space R, $h : X \to R$. That is, the weak classifier divides the sample space into J disjoint blocks, where J is the number of partitions,

$$X = \bigcup_{j=1}^{J} X_j. \tag{2.8}$$

Then, each weak classifier is defined by

$$h_t(x) = \frac{1}{2} \log \frac{P(W^j_{+1} + \epsilon)}{P(W^j_{-1} + \epsilon)}, \quad j = 1, 2, .., J, \tag{2.9}$$

where ϵ is a smoothing parameter, and $W_{\pm 1}$ is the probability distribution for positive and negative samples. They are computed using histograms as follows:

$$W^j_{\pm 1} = P(x_n \in X_j, y_n = \pm 1) = \sum_{n: x_n \in X_j \wedge y_n = \pm 1} D_t(x_n) \quad j = 1, 2, .., J \quad \forall x_n \in X, \tag{2.10}$$

where D_t is the distribution of sample weights at current iteration t.

To select the most discriminative weak classifier for the current iteration, the Bhattacharyya coefficient is used. It measures the classification power of the weak classifier. The smaller this measure is, the more discriminative the weak classifier is. At each boosting iteration the weak classifier with the smallest value is chosen. This coefficient Q is calculated by

$$Q = \sum_{j=1}^{J} \sqrt{W^j_{+1} W^j_{-1}}. \tag{2.11}$$

After selecting the weak classifier h_t, the algorithm also updates the distribution of weights D over the training samples. Initially, all weights are set equally, but on each round, the weights of misclassified samples are increased so that the algorithm is forced to focus on such hard samples in the training set that were previously missed by the classifier.

Finally, the strong classifier is defined as:

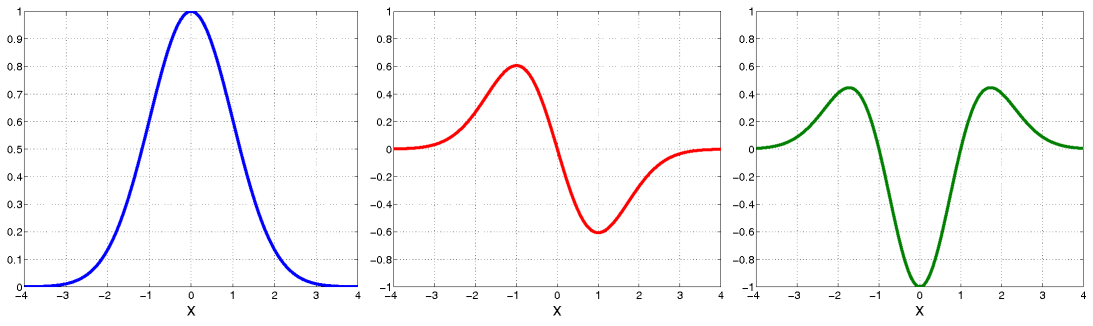
$$H(x) = sign\left(\sum_{t=1}^{T} h_t(x) - \beta \right), \tag{2.12}$$

where T is number of weak classifiers, and β is the classifier threshold. Pseudocode for Real AdaBoost is given in Alg. 2.

2.4 Gaussian-based Features

Gaussian-based functions are widely used in computer vision tasks because of their properties such as orientability, separability and self-similarity. Furthermore, they can be used for feature extraction since some of them resemble some image structures such as edges.

First, we describe Gaussian derivative functions in 1D with the objective of seeing how image patterns emerge when spatial derivatives of the Gaussian function are taken.



(a) $G(x)$ (b) $G'_x(x)$ (c) $G''_x(x)$

Figure 2.3: 1D Gaussian-based functions. (a) Gaussian function. (b) First order Gaussian derivative. (c) Second order Gaussian derivative.

The Gaussian function G for a 1D signal x is defined by,

$$G(x) = e^{-\frac{x^2}{2\sigma^2}}, \tag{2.13}$$

where σ is the standard deviation that also refers to size of the Gaussian function.

Differentiating, the first and second order Gaussian derivatives $G'_x(x)$, $G''_x(x)$ become

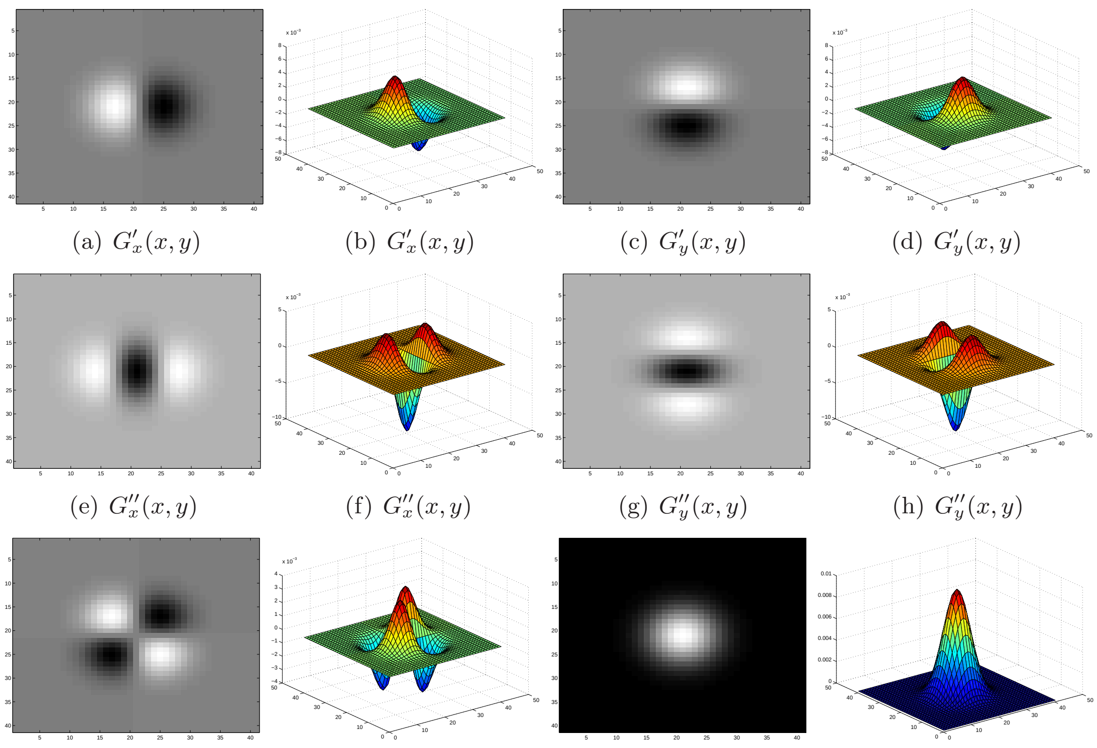
$$G'_x(x) = \frac{-x}{\sigma^2}e^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2}G(x), \tag{2.14}$$

$$G''_x(x) = \frac{x^2-\sigma^2}{\sigma^4}e^{-\frac{x^2}{2\sigma^2}} = \frac{x^2-\sigma^2}{\sigma^4}G(x). \tag{2.15}$$

These functions are shown in Fig. 2.3. We can see that they would respond to step and peak changes.

One important fact when we work with Gaussian derivatives is that their responses depend on the size of the filter, that is, the parameter σ. For instance, the maximum values of the functions $G'_x(x)$ and $G''_x(x)$ are determined by $\pm\frac{1}{\sigma}e^{-\frac{1}{2}}$ and $\frac{-1}{\sigma^2}$, respectively. Clearly, they rely upon the parameter σ. If the goal is to extract features in a multi-scale approach, Gaussian derivatives must be normalized to scale. This is done by adding an auxiliary normalization factor. To this aim, suppose we have a function f composed by one coordinate variable x and one scale variable σ, and that theses variables in the function f are related by $\frac{x}{\sigma}$. Then, it is possible to create a dimensionless variable $m = \frac{x}{\sigma}$ with which to obtain a scale invariant derivative. Using this new parametrization, we can write function derivatives of order n as:

$$\frac{\partial^n f}{\partial m^n} = \sigma^n \frac{\partial^n f}{\partial x^n}, \tag{2.16}$$


(a) $G'_x(x,y)$ (b) $G'_x(x,y)$ (c) $G'_y(x,y)$ (d) $G'_y(x,y)$

(e) $G''_x(x,y)$ (f) $G''_x(x,y)$ (g) $G''_y(x,y)$ (h) $G''_y(x,y)$

(i) $G''_{xy}(x,y)$ (j) $G''_{xy}(x,y)$ (k) $G(x,y)$ (l) $G(x,y)$

Figure 2.4: 2D Normalized Gaussian derivatives. First spatial order Gaussian derivatives resemble horizontal and vertical edges (a-d), while second order derivatives resemble lines (e-h). The second order Gaussian derivative in both axes G''_{xy} is useful for detecting diagonal edges and corners (i-j). The 2D Gaussian function is at bottom right.

where σ^n is the normalization factor and $\frac{\partial^n f}{\partial m^n}$ is the scale normalized derivative. The proof of this scale normalization was explained in [74, 86].

Adding this normalization factor to the previous Gaussian derivatives, the first and second order normalized Gaussian derivatives become

$$G'_x(x) = \sigma \frac{-x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma} G(x), \tag{2.17}$$

$$G''_x(x) = \sigma^2 \frac{x^2 - \sigma^2}{\sigma^4} e^{-\frac{x^2}{2\sigma^2}} = \frac{x^2 - \sigma^2}{\sigma^2} G(x). \tag{2.18}$$

The maximum values for these normalized functions are $\pm e^{-\frac{1}{2}}$ and -1, which are of course independent of the filter's scale.

For the 2D case, the Gaussian derivatives are normalized in the same way as commented previously, that is, by using a scale normalization factor. 2D spatial Gaussian derivatives are useful for the extraction of image structures given that they resemble edges, lines and some special contours –see Fig. 2.4–. For example, the first order Gaussian derivatives capture information about changes of the surface normal and then measure the intensity of edges. On the other hand, the second Gaussian derivatives can be used to extract image features such as bars, blobs and corners.

To illustrate the computation of 2D normalized Gaussian derivatives, the first and second order Gaussian derivatives in the axis x are calculated from the definition of 2D Gaussian function,

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{2.19}$$

Hence, the first and second order normalized derivatives can be formulated as:

$$G'_x(x,y) = \sigma \frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} = \frac{-x}{\sigma} G(x,y), \tag{2.20}$$

$$G''_x(x,y) = \sigma^2 \frac{x^2-\sigma^2}{\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} = \frac{x^2-\sigma^2}{\sigma^2} G(x,y). \tag{2.21}$$

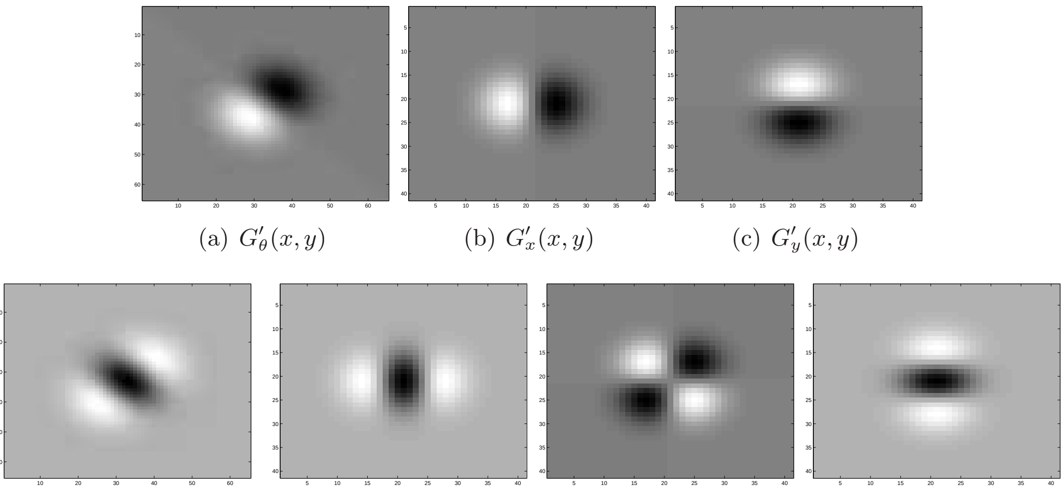
2.5 The Steerable Filters

In numerous applications it is necessary to convolve a filter to any given orientation. The simplest strategy would be to have a large set of filters, where each one is specialized to one particular orientation. However, this approach is a brute force implementation that has a high computational cost because each filter should be convolved with the image if the goal is to test the filter for many orientations.

Freeman and Adelson proposed the *steerable filters* for the rotation case [22]. They state that if a filter is steerable, it can be expressed as a linear combination of a fixed set of oriented filters. The best known steerable filters are the Gaussian derivatives, whose rotated versions can be computed from a filter basis consisting of $n+1$ oriented filters, where n is the derivative order.

For instance, the first order Gaussian derivative $G'_\theta(x,y)$ in an arbitrary orientation θ is given by

$$G'_\theta(x,y) = \cos(\theta) G'_x(x,y) + \sin(\theta) G'_y(x,y), \tag{2.22}$$


(a) $G'_\theta(x, y)$ (b) $G'_x(x, y)$ (c) $G'_y(x, y)$

(d) $G''_\theta(x, y)$ (e) $G''_x(x, y)$ (f) $G''_{xy}(x, y)$ (g) $G''_y(x, y)$

Figure 2.5: Steerable Filters. First order Gaussian derivative (a) to any orientation (i.e $\frac{7\pi}{4}$) can be synthesized by a linear combination of filter basis (b-c). In the same way, the second order Gaussian derivative to the same orientation (d) is calculated using a fixed set of basis filters (e-g).

where $\cos(\theta)$ and $\sin(\theta)$ are the basis coefficients, whereas $G'_x(x, y)$ and $G'_y(x, y)$ are the filter basis. Since convolution is a linear operation, the response of convolving the steered filter $G'_\theta(x, y)$ with an input image I can be expressed as:

$$G'_\theta(x, y) * I = \cos(\theta)G'_x(x, y) * I + \sin(\theta)G'_y(x, y) * I. \qquad (2.23)$$

We see that the basis filters $G'_x(x, y)$ and $G'_y(x, y)$ correspond to Gaussian filters computed at orientations 0 and $\frac{\pi}{2}$. That is, to rotated versions of the steered filter, and that only two basis filters are necessary to compute the filter to any orientation, and thus to avoid a large number of specialized filters for every orientation. This shows the great benefit of steerability, a derivative filter can be computed in any given orientation with minimal computational costs.

Similarly, the second order Gaussian derivative can be calculated as follows:

$$G''_\theta(x, y) = \cos^2(\theta)G''_x(x, y) + 2\cos(\theta)\sin(\theta)G''_{xy}(x, y) + \sin^2(\theta)G''_y(x, y). \qquad (2.24)$$

where basis filters $G''_x(x, y)$, $G''_{xy}(x, y)$ and $G''_y(x, y)$ are the spatial Gaussian derivatives. Fig. 2.5 shows the first and second order Gaussian derivatives steered to $\frac{7\pi}{4}$. They are computed as the linear combination of the responses of their basis filters.

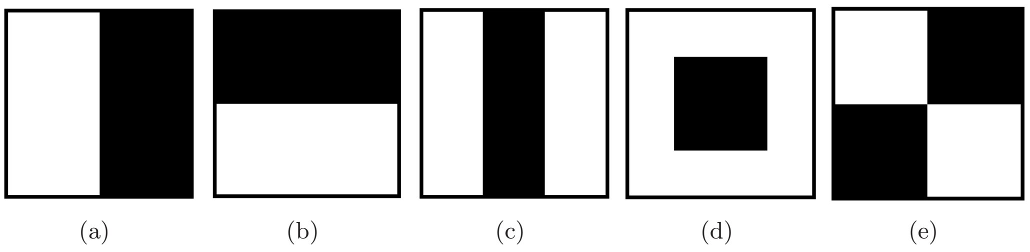


Figure 2.6: Haar-like features. (a,b) Features that respond to horizontal and vertical edges, respectively. (c) Haar-like feature used to extract horizontal stripe features. (d) Center-surround feature. It responds to image blobs and edges. (e) Special diagonal stripe feature.

There is a large number of applications for steerable filters where edge detection, texture and motion analysis are just some examples. Extensions to this idea allow to approximate and design orientation-selective features [67, 52, 79, 85, 80, 36]. Steerable filters are also used to compute rotationally-invariant descriptors which are calculated over interest points in order to describe locally object components. Ballard and Wixson [5], for instance, addressed object recognition using steerable filters as local descriptors. In this work, the descriptors achieve rotational invariance by steering all filters according to the local image orientation that is calculated from the first Gaussian derivatives. A similar work was presented by Yukono and Poggio [102] for building local descriptors consisting of oriented Gaussian filters up to the third order and evaluated for several scales. These descriptors are localized over corner points using the Harris corner detector [31].

2.6 Haar-like Features and the Integral Image

Rapid feature computation is an important issue when we attempt to perform object detection in real-time. Therefore, a detection system that uses simple features, with a low computational cost, is essential for achieving high detection rates.

For instance, Papageorgiou *et al.* proposed in [66] simple yet powerful features in order to capture contour-based features for detecting humans within images. These rectangular features are reminiscent to Haar basis functions and defined by the difference between adjacent image regions, being each rectangular region the sum of pixel intensities within that region. Some Haar-like features are depicted at Fig. 2.6. These

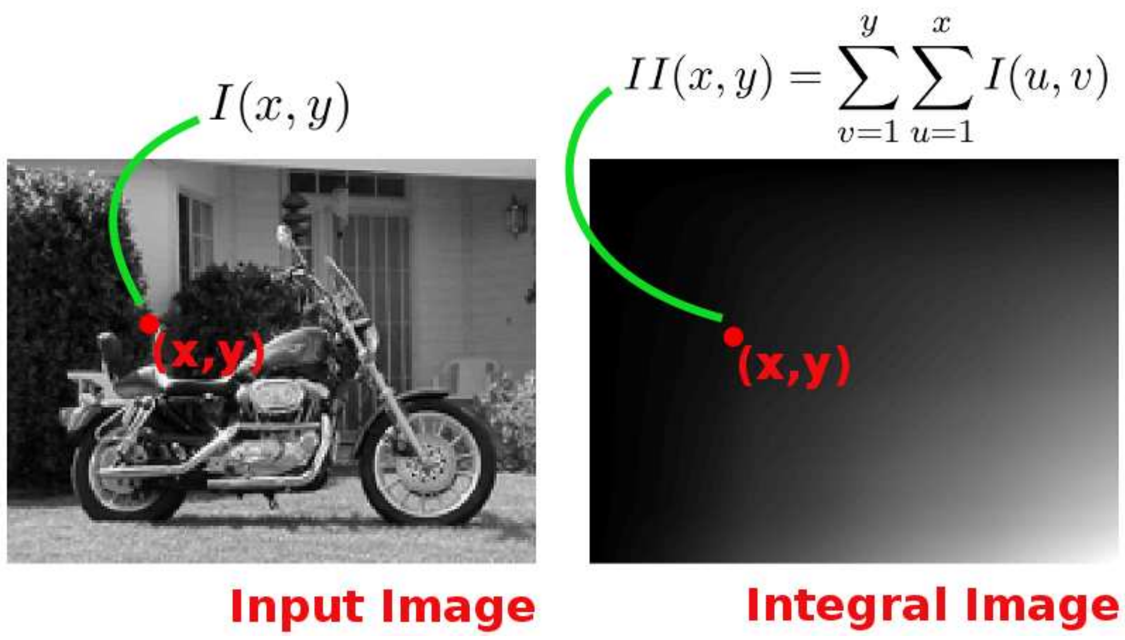


Figure 2.7: Integral Image. The value at coordinates (x, y) corresponds to the sum of pixel intensities from all pixel locations above and to the left of (x, y).

features are sensitive to the presence of edges, stripes and points.

With the aim of evaluating Haar-like features in an efficient way, the *integral image* was introduced by Crow in [12] and popularized by Viola and Jones in [98]. This image is useful for computing the response to rectangular features, such as Haar-like features, because it is an image representation that once it is computed it enables to calculate this type of features in a few pixel operations. In this image representation, the value at coordinates (x, y) contains the sum of pixel intensities above and to left of (x, y), inclusive,

$$II(x,y)=\sum_{v=1}^{y}\sum_{u=1}^{x}I(u,v), \tag{2.25}$$

where I is the input image. This computation is illustrated in Fig. 2.7. Furthermore, the integral image can be computed recursively using its previous calculated values,

$$II(x,y)=II(x-1,y)+II(x,y-1)-II(x-1,y-1)+I(x,y). \tag{2.26}$$

Once the integral image is computed, a rectangular region –sum of pixel intensities– can be calculated only from its corner values in the new image representation. For

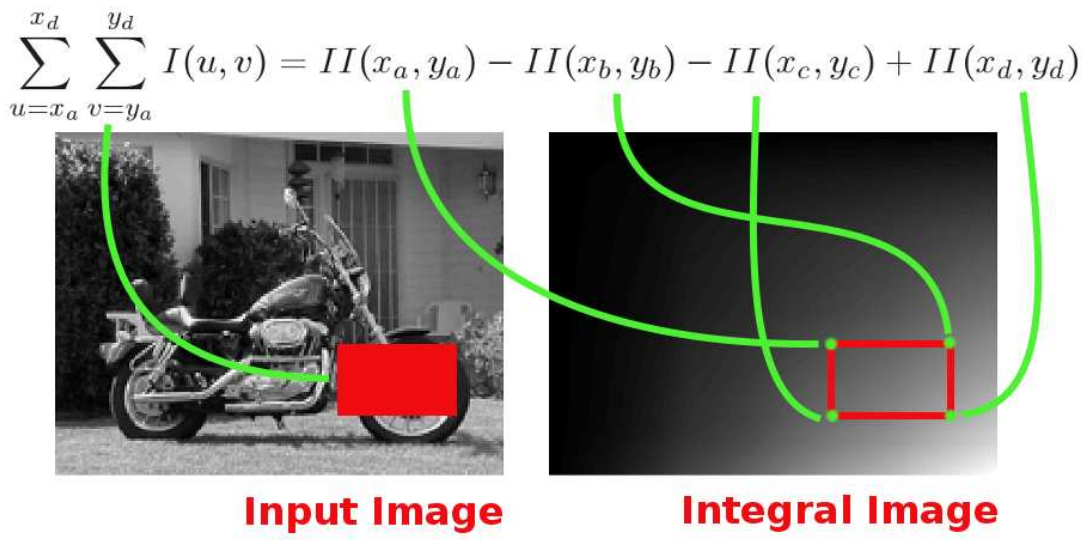

$\sum_{u=x_a}^{x_d} \sum_{v=y_a}^{y_d} I(u,v) = II(x_a,y_a) - II(x_b,y_b) - II(x_c,y_c) + II(x_d,y_d)$
Input Image
Integral Image

Figure 2.8: Rectangular feature computation. The sum of pixel intensities within a rectangular image region can be calculated by means of its corner values at the integral image.

example, if we wish to compute the sum of intensity values inside the rectangular region defined by corners a, b, c, d –Fig. 2.8– it can be obtained easily by

$$\sum_{u=x_a}^{x_d} \sum_{v=y_a}^{y_d} I(u,v) = II(x_a,y_a) - II(x_b,y_b) - II(x_c,y_c) + II(x_d,y_d). \tag{2.27}$$

Naïvely the left-hand side of equation implies $n_1 \times n_2$ operations. Contrary, the right-hand side of the same equation only takes four operations using the integral image. This is the advantage of using rectangular features in combination with an integral image. Features can be calculated at any location and scale in constant time provided the integral image is available from an initial processing step. The end result is that Haar-like features can be evaluated using few pixel operations.

With the objective of improving the detection results of the method presented by Viola and Jones [98], new types of Haar-like features were proposed by Lienhart in [47]. These features enrich the simple feature pool commented above –Fig. 2.6– by adding rotated Haar-like features. In Fig. 2.9 some of these new features are shown. With these extended features, any detection system is able to extract more meaningful features with which to describe objects. This is particularly convenient when the objects exhibit

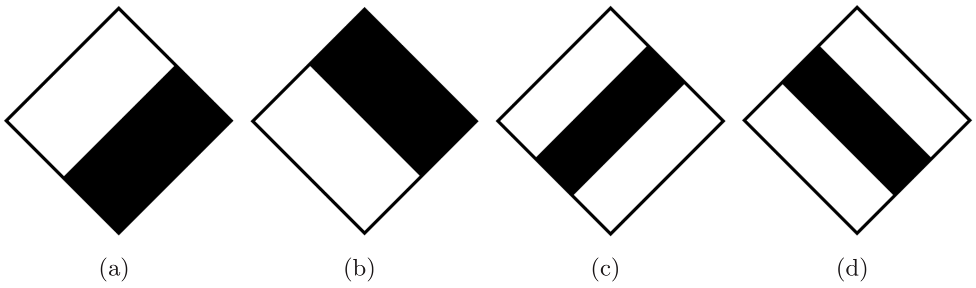


Figure 2.9: Extended Haar-like features. (a,b) Rotated edge features. (c,d) Rotated stripe features. These features are calculated by means of an auxiliary integral image that increases their computational cost.

diagonal structures. However, such features are only oriented to 45 degrees and for their computation an additional rotated integral image is needed. This is an important disadvantage because the computational cost is increased by computing two integral images, and because it partially solves the problem of extracting features for multiple orientations.

Some extensions to the integral image and Haar-like features have been proposed and used in the last years with the goal of performing fast feature computation and to obtain efficient detection methods. For example, methods based on Gaussian derivatives have replaced these functions by Haar-like features to speed up their computation [7, 95]. On the other hand, the integral image has been extended for computing image moments or integral histograms of image features [76, 68, 90].

2.7 Histogram of Oriented Gradients

Histogram of Oriented Gradients –HOG– is a very popular technique in the computer vision community that has been used to describe the appearance of image regions. Its popularity lies in its robustness to small geometric distortions thanks to the quantization of the gradient location and orientation. The overall idea is to capture the spatial and orientation distribution of gradients, computed in a local image region, in order to form a robust descriptor with which to encode the appearance of image regions.

Several types of HOG-based descriptors have been proposed in the past [49, 13, 10, 40]. They differ in how and where these descriptors are computed. Basically, they can

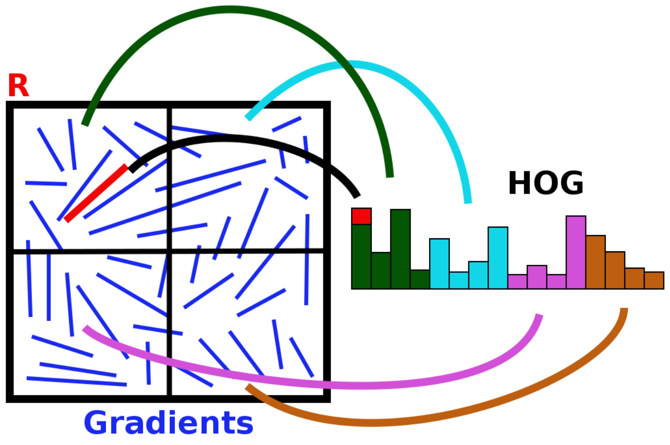


Figure 2.10: HOG computation. The HOG is a histogram where each gradient in the region R casts a vote for one spatial and orientation bin according to its location (x, y) and its orientation $\psi(x, y)$. Each vote is weighted using the gradient magnitude. For this example, the HOG consists of 2×2 cells and 4 orientation bins.

be calculated using different spatial and orientation configurations, or calculated in a sparse or dense manner over images.

The SIFT descriptor proposed by Lowe [49], for example, is calculated in a sparse manner by means of an interest point detector –DoG detector [49, 89]– that yields image locations where the descriptors are computed subsequently. The descriptor rotationally invariance is obtained by determining the orientation of the interest point and rotating the gradients according to this orientation. This type of HOG-based descriptor has a fixed configuration of 4×4 spatial bins and eight gradient orientation bins. This yields a feature vector with 128 elements. The SIFT descriptor is similar to the *shape context* descriptor proposed by Belongie *et al.* [8]. However, the latter is a histogram of edge point locations that is computed using a log-polar configuration.

Because of the successful results given by the SIFT descriptor, some extensions have been proposed in order to improve its performance. For example, the RIFT descriptor, created by Lazebnik *et al.* [40], is a rotationally invariant descriptor that generalizes the SIFT. In contrast to this descriptor, the RIFT avoids to find the dominant orientation of interest points for its construction. Basically, this descriptor is a histogram of oriented gradients that is constructed using concentric rings of equal width, where in each one of them a local gradient orientation histogram is computed. For its construction, four rings and eight gradient orientations are used. It yields a feature vector of 32 elements. To achieve rotationally invariance the orientation of each gradient –inside the support region– is measured relative to the direction from the region center.

Another similar descriptor is the GLoH –Gradient Location and Orientation Histogram– proposed by Mikolajczyk and Schmid [57]. This descriptor can be seen as a combination between the shape context descriptor and the SIFT, given that it computes a HOG using a log-polar configuration. This log-polar configuration captures the layout of gradients that are spatially quantized into three radial bins and eight angular ones. For each spatial bin a local histogram of gradient orientations is calculated. For this descriptor 16 orientation bins have been chosen. The result is a feature histogram with 272 elements that is reduced via PCA to 128 elements, equal to the SIFT.

Inspired from the work of Lazebnik *et al.* [41] for image pyramid representations of scenes, Bosch *et al.* [10] introduced a Pyramidal Histogram of Oriented Gradients –PHOG– where histograms are computed for multiple resolutions. This allows to have a compact representation where the pyramidal descriptor encodes features and their spatial layout at several resolution levels and gives robustness to small feature shifts. In this representation, finer histogram levels are weighted more than coarser ones because finer levels have more detailed feature shape information.

Unlike previous works in which the descriptors are computed over interest points, Dalal and Triggs [13] proposed to compute HOGs densely over the whole image with the aim of detecting pedestrians. For their computation, the image is first divided into cells consisting of image regions of 8×8 pixels, to then build overlapping blocks formed by the concatenation of 2×2 cells. For every cell a histogram of gradient orientations with 9 elements –orientations– is calculated. Given these blocks, the object classifier is trained via SVM. To gain certain invariance to illumination changes and shadows the authors utilize an overlapping local contrast normalization in this work.

In order to show the construction of a HOG-based descriptor, we proceed to build one simple descriptor consisting of 2×2 spatial bins or cells and four gradient orientation bins in an image region R, see Fig 2.10. For this example, each gradient's vote is only weighted by its magnitude.

First, image gradients are computed using differential operators, namely Prewitt operators, over the input image $I(x, y)$ by

$$\lambda_y(x, y) = I(x, y) * P_y, \quad \lambda_x(x, y) = I(x, y) * P_x, \tag{2.28}$$

where λ_y and λ_x are the gradient image maps, P_y and P_x are the Prewitt operators for axes y and x, respectively, and $*$ denotes convolution. From the gradient maps, the orientation ψ is calculated for each image gradient at coordinates (x, y) according to:

$$\psi(x, y) = \arctan \frac{\lambda_y(x, y)}{\lambda_x(x, y)}, \quad \forall (x, y) \in R. \tag{2.29}$$

Subsequently, each gradient casts a vote for one spatial and orientation bin according to its location (x, y) inside the region R and its orientation $\psi(x, y)$. For the orientation case, the orientation $\psi(x, y)$ is discretized into m orientations bins –in this example $m = 4$–, while for the spatial case the image region is divide into 2×2 adjacent cells or bins. Therefore, each gradient casts a vote for its spatial and orientation bin using its magnitude $mag(x, y)$ as weight. This procedure is computed by

$$mag(x, y) = \|(\lambda_y(x, y), \lambda_x(x, y))\|_2, \tag{2.30}$$

and

$$HOG_R(s, b) = \sum mag(x, y) \quad \forall (x, y) \in R, \tag{2.31}$$

where HOG_R is the histogram of oriented gradients computed in the image region R, and, b and s are the orientation and spatial bins for the gradient at coordinates (x, y). The result is a histogram of 16 elements that contains the layout of gradients. This final histogram can also be seen as a concatenation of local histograms of oriented gradients computed in each spatial cell.

2.8 Integral Histogram

Following the same idea of the integral image and its great computational benefits for computing fast features, a new representation called *integral histogram* was proposed by Porikli [68]. It shares the same idea but with the difference of computing histograms instead of accumulative pixel intensities. Once the integral histogram is constructed in a previous step, histograms over arbitrary rectangular image regions can be calculated in constant time.

Thanks to this histogram representation, descriptors based on histograms of pixel intensities, HOGs [13], or multidimensional histograms built of any type of features [74], can be computed efficiently. For instance, in [104] the authors used this representation to speed up the computation of HOG-based descriptors. This work reported results for human detection comparable with the Dalal's work [13], but with improved efficiency.

The construction of this image is carried out as follows. Suppose we have an input image I from where a set of features is extracted for every pixel location (x, y). These features, namely gradient orientation, intensity, Gaussian derivatives, etc, form a new image consisting of C channels that correspond to the specific feature outputs. From this new image, the integral histogram is built as:

$$IH(x, y, f_c) = \sum_{v=1}^{y} \sum_{u=1}^{x} I(x, y, f_c), \quad c = 1, 2, .., C, \tag{2.32}$$

where f_c denotes the feature channel from the image I. We see that this equation is pretty similar to the computation of the integral image but adding an extra dimension. It means that computing an integral histogram can be seen as computing C consecutive integral images.

Similar to the integral image case, the integral histogram can be constructed iteratively using its previous values or histograms,

$$IH(x, y, f_c) = IH(x-1, y, f_c) + IH(x, y-1, f_c) - IH(x-1, y-1, f_c) + I(x, y, f_c). \tag{2.33}$$

Then, the computation of histograms over rectangular image regions is done using its corner values, that in this case, are accumulative vectors or histograms. Suppose we wish to calculate the feature histogram for an image region R defined by its corners a, b, c, d, the histogram is obtained by

$$Hist_R = IH(x_d, y_d, f_c) - IH(x_a, y_a, f_c) - IH(x_b, y_b, f_c) + IH(x_c, y_c, f_c), \quad c = 1, .., C. \tag{2.34}$$

where a is the top-left corner, b the top-right, c the bottom-left and d the bottom-right one.

As a consequence, the integral histogram allows to compute any rectangular histogram in $4 \times C$ operations, independently of its size and location. This fact shows its great computational benefit in contrast to computing a histogram without the integral representation which requires $n_1 \times n_2 \times C$ operations, being $n_1 \times n_2$ the region size.

Chapter 3

Efficient Detection of Specific Objects Under In-plane Rotations

In this chapter we present an object detection method for the case of in-plane rotated objects. The efficiency of the proposed method lies in the use of Haar-like features in combination with an orientation estimator with the aim of reducing the computational cost of evaluating the object classifier for multiple orientations. In addition, a boosting algorithm is used to carry out feature selection and to build a robust and discriminative object classifier. This classifier can be rotated to any given orientation by a simple procedure based on steering the Haar-like features via steerable filters. The work in this chapter has been presented in [95].

3.1 Introduction

For object detection, most methods commonly scan the image using a sliding window where at each image location an object classifier is evaluated to determine whether the current location contains an object instance or not. This approach can also be seen as a binary classification problem that is carried out over the entire image. It allows detecting and localizing spatially the objects within the input image and copes with scale changes if this procedure is repeated for multiple image scales. However, this method is computationally costly requiring thousands or even millions of evaluations.

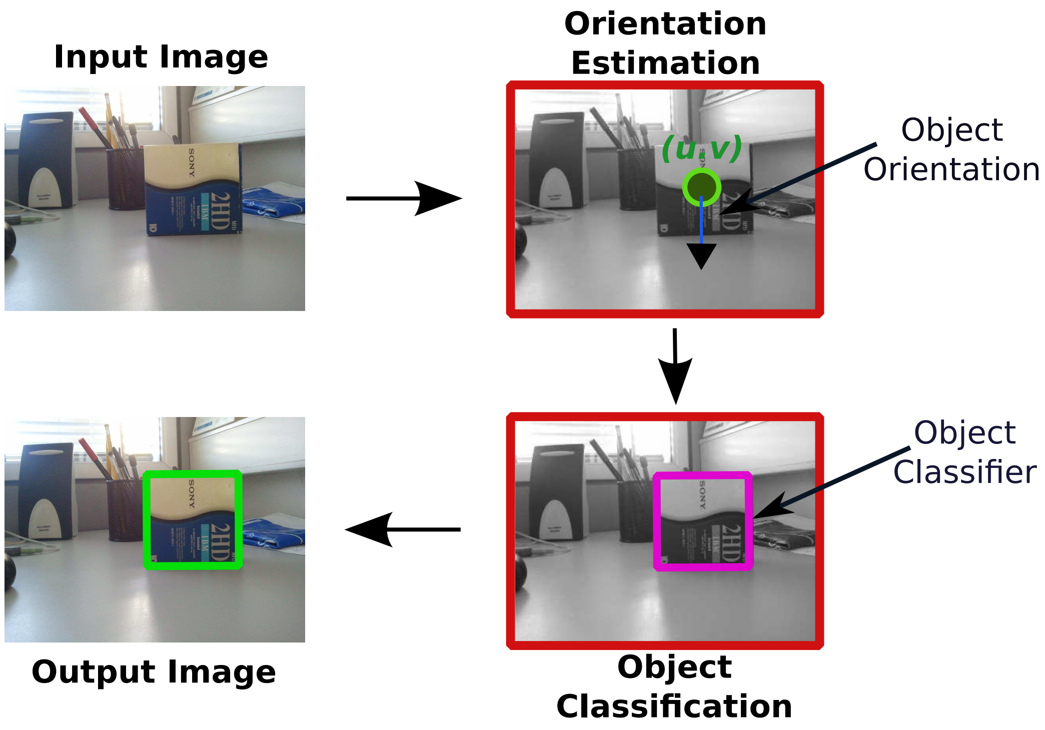


Figure 3.1: The proposed approach. Given an input image, an estimator of local orientation is tested for every image location (u, v). Then, a trained object classifier is steered according to such predictions of local orientation and tested. This step is carried out by steering the set of local features using approximated steerable filters.

To reduce the computational cost, some ideas have been introduced like using low-cost features or using cascades of classifiers to speed up the detection process.

A well-known and seminal method including the aforementioned ideas is the work proposed by Viola and Jones [98] for detecting frontal faces in real-time. More precisely, this method makes use of Haar-like features, AdaBoost and a cascade of classifiers. Because of the successful results and the computational efficiency of this method, recent methods have done extensions to focus on more challenging applications such as detection of faces from several views [34, 99], detection of pedestrian [104] or hand and gesture recognition [11, 58].

In this chapter, we follow the previous work for a fast detection of objects. However, we incorporate new ideas in order to consider objects having orientation changes in the image plane. To this end, we propose a local orientation estimator and fast oriented features, see Fig. 3.1. They allow having an efficient strategy for detecting rotated objects which is based on two consecutive steps: orientation estimation and object classification. The estimation step determines locally the object orientation using the distribution of image gradients, whereas the second step steers and tests the object classifier according to the estimations of orientation given by the estimator. The proposed method, therefore, avoids the testing of the object classifier for many orientations.

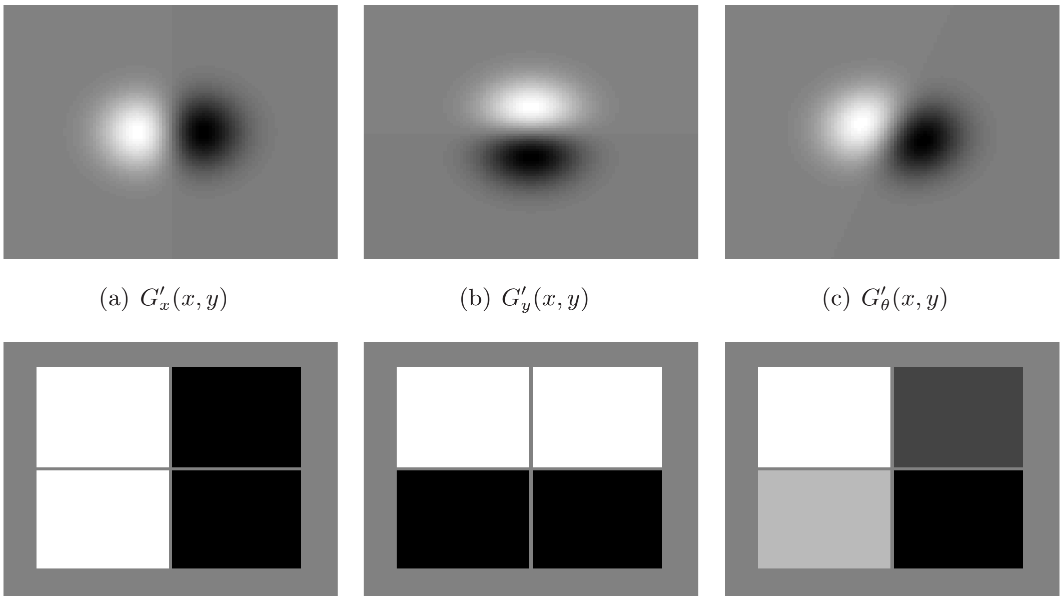
On the other hand, the proposed features can be computed and steered to any given orientation in a fast way. This is done by steering the Haar-like features via steerable filters. Although Lienhart and Maydt [47] proposed oriented Haar-like features, these features require a more complex procedure for their computation including an auxiliary integral image.

3.2 Efficient Oriented Features

We propose first to compute efficient features for extracting contours in images by approximating Gaussian derivatives with Haar basis, and orienting the filter using steerable filters.

Haar features have demonstrated to be a good choice for fast feature computation when they are used in combination with an integral image representation, see Sec. 2.6. However, they cannot be computed apart from canonical horizontal and vertical forms; and to use an auxiliary integral image for oriented features would increase the computational cost and only diagonal features could be computed –i.e, 45 degrees–. This is a great disadvantage because the object description would rely entirely on vertical and horizontal features –edges and lines–, and disregard other rotated edges that might be discriminative for classification.

Nevertheless, we have seen that oriented Gaussian derivatives can be synthesized to any orientation using a combination of a fixed set of filters, see Sec. 2.5. From this point of view, we propose to approximate Gaussian derivative filters using Haar basis given their strong similarity. With this approximation, a more efficient and simpler way to evaluate Haar-like features to a given orientation is obtained without having to compute auxiliary integral images or having to construct a large set of features for several orientations. That is, we approximate the first and second order Gaussian derivatives with the goal of extracting edges and stripes over images. These approximations are



(a) $G'_x(x,y)$ (b) $G'_y(x,y)$ (c) $G'_\theta(x,y)$

(d) $H'_x(x,y)$ (e) $H'_y(x,y)$ (f) $H'_\theta(x,y)$

Figure 3.2: First order filter approximation. The filter is steered to an orientation of $\frac{\pi}{9}$ by interpolating two basis filters. (a,b) Gaussian basis filters. (c) Steered Gaussian derivative. (d,e) Haar basis filters. (f) Steered Haar filter.

visualized in Fig. 3.2 and Fig. 3.3. In these figures, we can see the strong similarity between Gaussian and Haar features.

Approximating Gaussian-based filters by Haar-like filters has been used in the past to speed up feature extraction. Bay *et al.* [7], for instance, used this approximation with the aim of extracting interest points. This method, coined as SURF –Speeded Up Robust Features–, extracts interest points using the determinant of the Hessian matrix, which consists of Gaussian derivatives that are replaced by their equivalent Haar filters. Then, a fast interest point detector is built whose performance is comparable with the discretized Gaussian-based filters.

Here, the Haar approximations of the first and second order Gaussian derivatives, calculated at any orientation θ, are computed with

$$H'_\theta(x,y) = \cos(\theta)H'_x(x,y) + \sin(\theta)H'_y(x,y), \tag{3.1}$$

$$H''_\theta(x,y) = \cos^2(\theta)H''_x(x,y) + 2\cos(\theta)\sin(\theta)H''_{xy}(x,y) + \sin^2(\theta)H''_y(x,y), \tag{3.2}$$

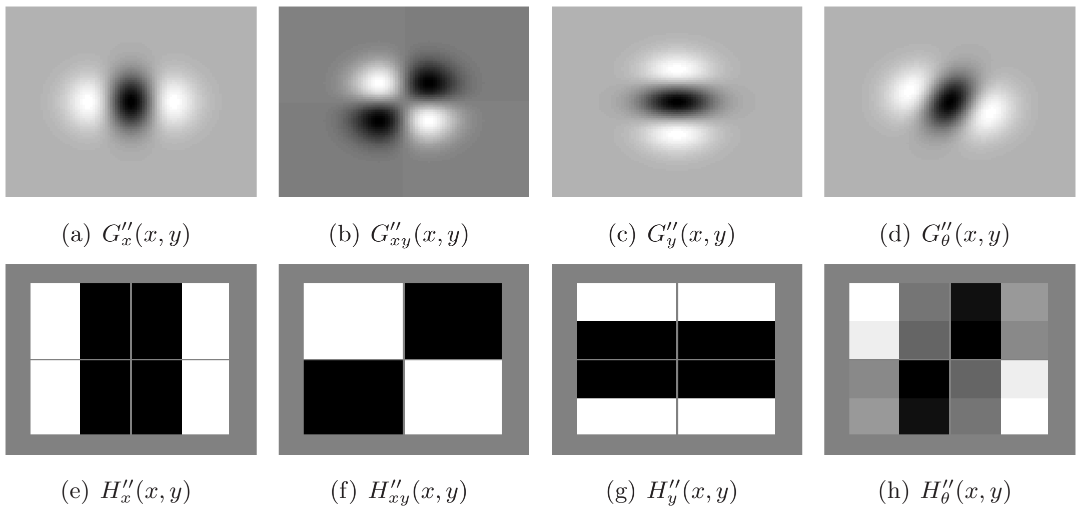
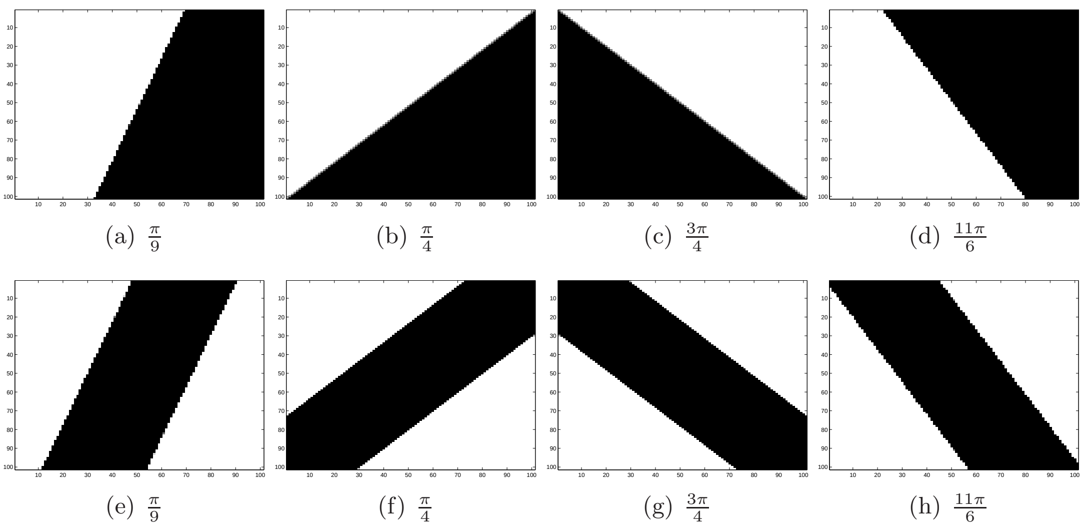


Figure 3.3: Second order filter approximation. The filter is steered to an orientation of $\frac{\pi}{9}$ by using three basis filters. (a-c) Gaussian basis filters. (d) Steered Gaussian derivative. (e-g) Haar basis filters. (h) Steered Haar filter.

being H'_x, H'_y, H''_x, H''_y and H''_{xy} the oriented basis filters. The main difference with steerable filters is the replacement of the Gaussian basis by their corresponding Haar versions. Fig. 3.2 illustrates one example where both the Gaussian derivative and the Haar features are steered to an orientation of $\frac{\pi}{9}$. Similarly, the second order Gaussian derivative and its efficient approximation are steered to the same orientation in Fig. 3.3.

To show how Haar-like features can be used for some computer vision tasks instead of Gaussian derivatives, two evaluations are carried out in this section. The first one consists on evaluating the accuracy of the steered filters. This is done by calculating the orientation of image features –edges and stripes– that are artificially rotated inside images, see Fig. 3.4. The objective is to compare the orientation estimation that both methods provide. This comparison offers a measure of similarity between the two addressed methods. The second evaluation refers to extracting relevant oriented features within images. To this end, the filters are steered to a given orientation and the number of most significant features are compared. This offers an idea to what extent both methods can extract the same image structures.

Feature Orientation. From the steerable filter equations –refer to Sec. 2.5–, we can derive equations to compute the local orientation of oriented edge and stripe features.



(a) $\frac{\pi}{9}$ (b) $\frac{\pi}{4}$ (c) $\frac{3\pi}{4}$ (d) $\frac{11\pi}{6}$

(e) $\frac{\pi}{9}$ (f) $\frac{\pi}{4}$ (g) $\frac{3\pi}{4}$ (h) $\frac{11\pi}{6}$

Figure 3.4: Artificial features computed for diverse orientations. (a-d) Edge features. (e-h) Stripe features.

Differentiating and equating to zero Eq. 2.22, the orientation with maximum response is achieved.

$$\frac{dG'_\theta(x,y)}{d\theta} = -\sin(\theta)G'_x(x,y) + \cos(\theta)G'_y(x,y) = 0 \tag{3.3}$$

$$\tan(\theta) = \frac{G'_y(x,y)}{G'_x(x,y)} \tag{3.4}$$

Hence, the maximum response orientation can be computed with:

$$\theta = \arctan\left(\frac{G'_y(x,y)}{G'_x(x,y)}\right). \tag{3.5}$$

The orientation of an edge within an input image I is the result of convolving the operator with I:

$$\theta_{edge} = \arctan\left(\frac{G'_y(x,y) * I}{G'_x(x,y) * I}\right). \tag{3.6}$$

It is important to point out that if the scale –σ– of the Gaussian derivative is small, the Sobel operators emerge and the last equation becomes the standard equation for computing the gradient orientation by means of the Sobel operators.

If the previous procedure is repeated for the second order Gaussian derivative, the orientation of a stripe feature can be computed. Differentiating Eq. 2.24,

$$\frac{dG''_\theta(x,y)}{d\theta} = -2\cos(\theta)\sin(\theta)G''_x(x,y) + 2G''_{xy}(x,y)(\cos^2(\theta) - \sin^2(\theta)) + 2\sin(\theta)\cos(\theta)G''_y(x,y), \tag{3.7}$$

$$\arctan(2\theta) = \frac{2G''_{xy}(x,y)}{G''_x(x,y) - G''_y(x,y)}, \tag{3.8}$$

the orientation of a stripe feature within an input image I is

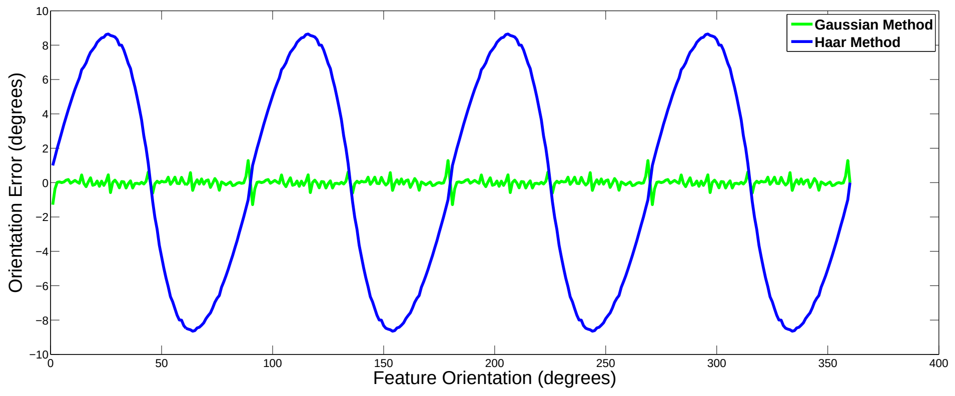
$$\theta_{stripe} = \frac{1}{2}\arctan\left(\frac{2G''_{xy}(x,y) * I}{G''_x(x,y) * I - G''_y(x,y) * I}\right). \tag{3.9}$$

Using Eq. 3.6 and Eq. 3.9, we can estimate the orientation of edge and stripe features present in images. For this evaluation, we have created a set of images containing artificial edges and stripes at different orientations, varying from 0 to 2π. Some sample images are shown in Fig. 3.4. These features represent ideal contours which might appear locally in images.

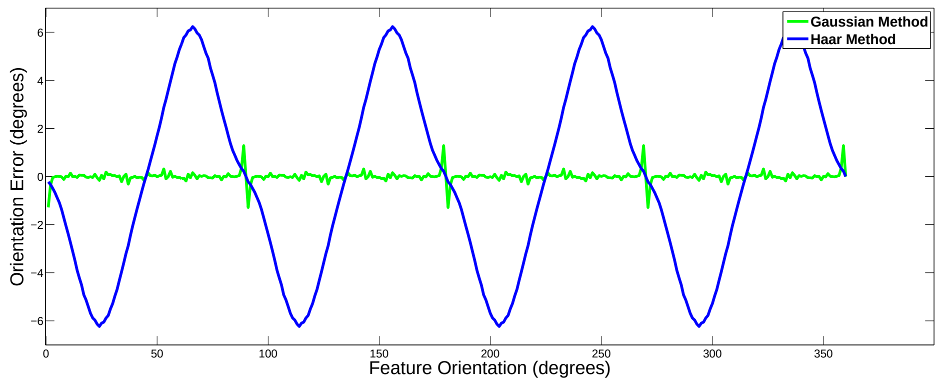
As our aim is to compare Gaussian to Haar-based filters, the aforementioned procedure is repeated substituting the Gaussian basis functions by their Haar counterparts. The comparison of both approaches is made by computing the absolute difference, in degrees, between the orientation of the artificial feature and the estimated orientation –see Fig. 3.5–. For the case of Gaussian filters, the orientation error is negligible to quantization. In contrast, Haar-based filters introduce an orientation error that varies periodically with respect to the input angle. This is because Haar-based filters are not polar separable. This fact is an essential requirement to obtain an error free steerable filter [22].

The experiment reported maximum orientation errors of eight degrees for edges and six degrees for stripe features. The maximum errors for Haar-based filters occur when features are oriented to $\frac{\pi}{8}(1+2n)$, being $n = 0, 1, 2, ..16$. However, for features with horizontal, vertical and diagonal orientations the error is negligible. This result shows that Haar-based filters are not suitable for an accurate orientation estimation of edge and stripe features. In spite of this, we show next how they can be used for fast feature evaluation to any given orientation with low orientation error.

Feature Extraction. We are interested in determining relevant image features in images such as strong oriented edges or stripes. In this test, we apply the filter for an



(a) Edge Feature.

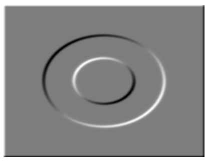


(b) Stripe Feature.

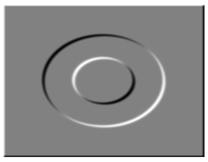
Figure 3.5: Feature orientation errors using Gaussian and Haar-based filters. (a) Orientation errors for oriented edge features. (b) Orientation errors for oriented stripe features. Note that the Gaussian-based approach retrieves correctly the feature orientation, while the approach based on Haar-based features is more efficient at the expense of a small periodic orientation estimation error.

orientation θ and consider its strongest responses as important image features. This is illustrated in the Fig. 3.6 where the input image I is convolved with a steered filter. We see the strong similarity between the responses of Gaussian-based filters and Haar-based filters. Fig. 3.6(f,k,p) shows the difference between both filter responses. These differences are relatively small and do not jeopardize the extraction of meaningful features. This comparison allows us the use of Haar-based filters instead of Gaussian

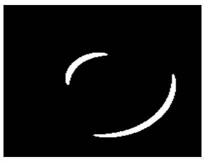
(a) I



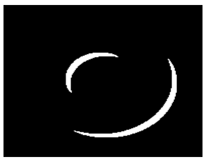
(b) $I * G'_{\frac{\pi}{4}}$



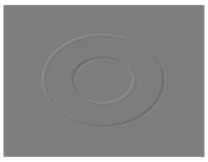
(c) $I * H'_{\frac{\pi}{4}}$



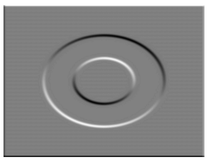
(d) $I * G'_{\frac{\pi}{4}} > 0.8$



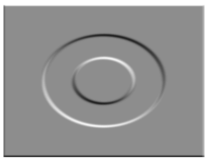
(e) $I * H'_{\frac{\pi}{4}} > 0.8$



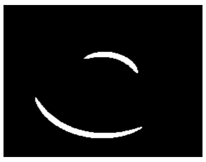
(f) $I * G'_{\frac{\pi}{4}} - I * H'_{\frac{\pi}{4}}$



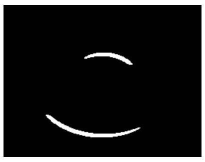
(g) $I * G'_{\frac{11\pi}{18}}$



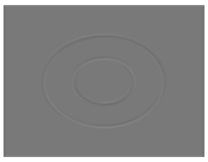
(h) $I * H'_{\frac{11\pi}{18}}$



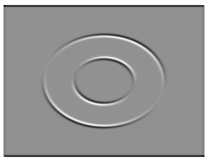
(i) $I * G'_{\frac{11\pi}{18}} > 0.8$



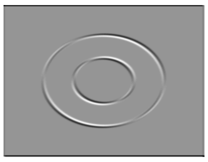
(j) $I * H'_{\frac{11\pi}{18}} > 0.8$



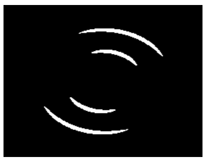
(k) $I * G'_{\frac{11\pi}{18}} - I * H'_{\frac{11\pi}{18}}$



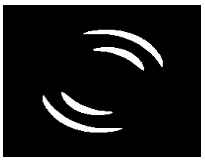
(l) $I * G''_{\frac{11\pi}{18}}$



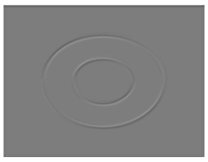
(m) $I * H''_{\frac{11\pi}{18}}$



(n) $I * G''_{\frac{11\pi}{18}} > 0.8$



(o) $I * H''_{\frac{11\pi}{18}} > 0.8$



(p) $I * G''_{\frac{11\pi}{18}} - I * H''_{\frac{11\pi}{18}}$

Figure 3.6: Oriented feature detection. Oriented features are extracted over an input image I using Gaussian derivatives and their Haar-based filter approximations. (a) Input image. (b-f) Filters computed at $\frac{\pi}{4}$. (g-p) Filters computed at $\frac{11\pi}{18}$.

ones for fast feature as well as for object description. This is motivated because Haar-based features, contrary to Gaussian filter response, can be calculated very fast at any location, orientation and scale in constant time.

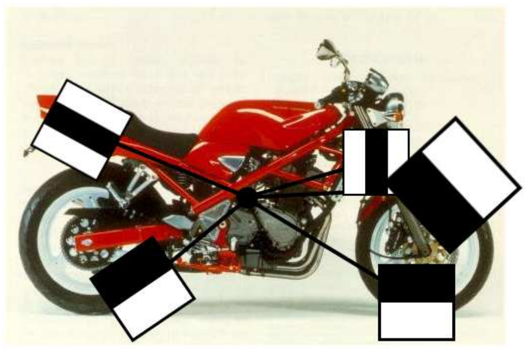
H	Object (strong) classifier
h	Weak classifier
α	Weak classifier weight
f	Haar-like feature
θ	Feature orientation
s	Feature scale
ρ	Parity value
δ	Feature threshold
ϕ	Estimated object orientation
ϕ_0	Reference orientation
φ	Steering orientation
I	Image
x	Image sample
T	Number of weak classifiers
β	Classifier threshold
y_n	Class label of image sample n
D	Distribution of sample weights
λ	Gradient map

Table 3.1: Notation for the computation of the object classifier.

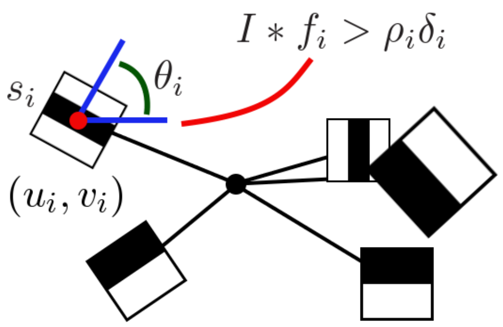
3.3 The Object Classifier

The classifier used for detecting specific objects in images is described in this section. The computation of this classifier makes use of both, the discrete AdaBoost algorithm –see Sec. 2.2– and the efficient oriented features described in the previous section. The aim is to seek out the most relevant and discriminative features for classifying an object class against another class formed by background images. The selected features via the boosting algorithm represent the most informative features for describing the object, and the resulting classifier is a combination of weak classifiers. Given the nature of filters used, informative classifiers tend to clutter around object contours. Notation for this section is shown in Table 3.1.

Pool of Features. The first step for building an object classifier with AdaBoost is to create a large pool of features over the whole image. This is done by computing edge and stripe features for different spatial image locations, sizes and orientations. Each combination –feature type, location, size and orientation– is a local feature for the pool and a potential weak classifier. The size of the feature pool depends on the number orientations, sizes and locations that the user chooses, but in general,



(a)



(b)

Figure 3.7: The object classifier. (a) The object is represented by a set of local features –oriented Haar-like features–. (b) Each local feature f_i is defined by its spatial location (u_i, v_i), orientation θ_i, size s_i and derivative order.

this value is commonly over thousands. This is another motivation to use a boosting algorithm, it performs feature selection at the same time that it builds the object classifier. Therefore, the most discriminative features are selected at each boosting iteration. This reduces the large number of possible features inside the pool to a small set that corresponds to the number of weak classifiers.

Computation. Discrete AdaBoost –DAB– in each iteration calls a weak learner in order to compute a weak classifier. This classifier is built by selecting the feature that best discriminates the positive samples from negative ones for the current distribution of weights. Positive samples refers to images containing object instances, whereas negative samples consists of images containing background. The weak classifier is formed by one Haar-like feature that is defined by its location (u_i, v_i) with respect to the object center, its orientation θ_i, its scale s_i and the order of derivative used, see Fig. 3.7. This weak classifier h, at iteration t, gives a binary decision,

$$h_t(x) = \begin{cases} 1 & : \quad x * f_t > \rho_t\delta_t \\ 0 & : \quad \text{otherwise} \end{cases}, \tag{3.10}$$

where x is a training image sample, f is the Haar-like filter being tested, $*$ indicates the convolution operation, $\rho = \{+1, -1\}$ is a parity indicating the direction of the inequality sign, and δ is the filter threshold. These last two parameters are determined by the weak learner as those values for which the test feature is most discriminative.

Algorithm 3 Object classifier computation.

```
1: Given a number of weak classifiers $T$, a feature consisting of $K$ Haar-like filters, and
   $N$ image samples $(x_1, y_1)...(x_n, y_n)...(x_N, y_N)$, where $y_n \in \{+1, -1\}$ is the label for
   positive and negative classes, respectively.
2: Initialize the sample weights $D_1(x_n) = \frac{1}{N}$, whit $n = 1, 2, .., N$
3: for $t = 1$ to $T$ do
4:    for $k = 1$ to $K$ do
5:       Using the current distribution of weights $D_t$, compute the weak classifier $h_k$
         with parameters $\rho_k$ and $\delta_k$.
6:       Compute its classification error $\epsilon_k$.
         $\epsilon_k = \sum_{n=1}^{N} D_t(x_n)|h_k(x_n) - y_n|$
7:    end for
8:    Select the weak classifier $h_t$ whose error $\epsilon_t$ minimizes $\epsilon_k$.
9:    Compute the weight $\alpha_t$ for the selected weak classifier.
      $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$
10:   Update the sample weights.
      $D_{t+1}(x_n) = \frac{D_t(x_n) \exp[-y_n h_t(x_n)]}{\sum_{n=1}^{N} D_t(x_n) \exp[-y_n h_t(x_n)]}$
11: end for
12: Using a validation set of images, calculate the classifier threshold $\beta$ that best dis-
    criminates positive from negative image samples.
13: Final strong classifier.
    $H(x) = sign \left( \sum_{t=1}^{T} \alpha_t h_t(x) - \beta \right)$
```

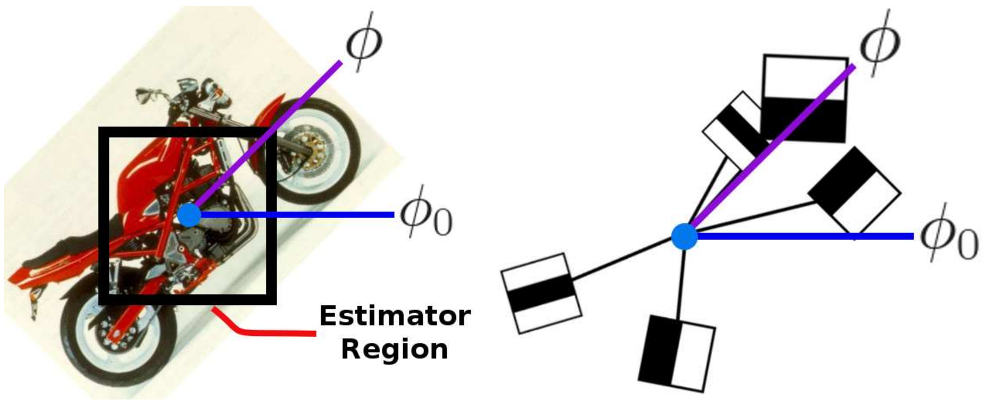
Finally, the strong classifier H collects each one of the computed weak classifiers h to assemble the final object hypothesis,

$$H(x) = \begin{cases} 1 & : \sum_{t=1}^{T} \alpha_t h_t(x) > \beta \\ 0 & : \text{otherwise} \end{cases} \tag{3.11}$$

where α is the weight assigned to each weak classifier and that is calculated according to its classification error over the training samples. The parameter β corresponds to the object classifier threshold which is obtained using a set of validation images. Pseudocode for the classifier computation is shown in Alg. 3.

3.4 The Orientation Estimator

As commented before, the detection of objects in images is done as a binary classification problem where the object classifier is evaluated at multiple locations and scales with the objective of achieving invariance to location and scale changes of objects. To



(a) Orientation estimator. (b) Rotation of the object classifier.

Figure 3.8: Rotation of the object classifier. (a) The estimator predicts the object orientation ϕ using the gradients inside the support region. The canonical or reference object orientation ϕ_0 is established during the learning phase. (b) Given the orientation estimations, the object classifier is rotated and tested.

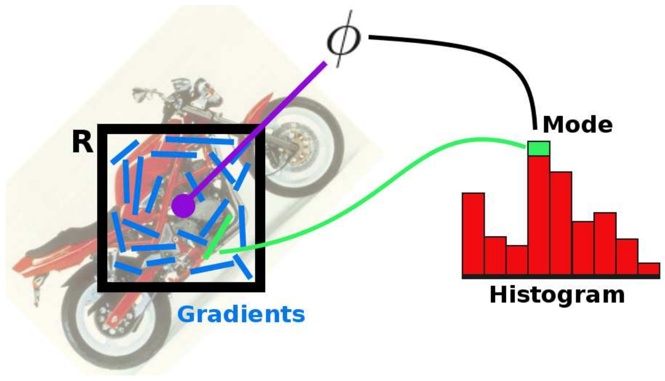
deal with in-plane orientation changes, object matching should be evaluated for multiple orientations as well, increasing the already high computational cost of evaluating the classifier for numerous hypotheses. Instead of trying each possible orientation, a more efficient way is to use an orientation estimator to indicate the most likely object orientation and then to rotate the object classifier according to this orientation. Rotating the classifier means to steer each local Haar-like feature and to change its spatial location in relation to the object center.

To compute the angle at which the classifier must be rotated, we compute the difference between the estimated orientation and the canonical or reference angle,

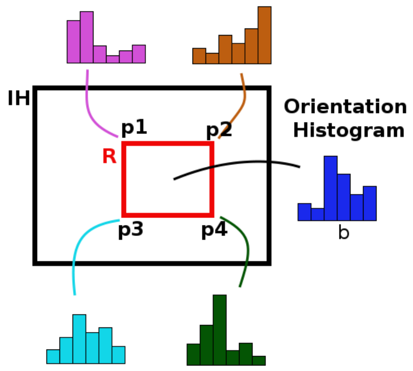
$$\varphi = \phi - \phi_0, \tag{3.12}$$

with ϕ the estimated orientation and ϕ_0 the reference one, this is the classifier orientation computed during the learning step. To illustrate this, Fig. 3.8 shows an example where the object orientation is computed using the orientation estimator. Subsequently, the object classifier is rotated according to this reference orientation.

To compute the object orientation ϕ, a histogram of gradient orientations within a support image region R is built. This histogram was introduced in Sec. 2.7 as a feature descriptor, but here, it is utilized to estimate the object orientation from image



(a) Object orientation



(b) Orientation histogram

Figure 3.9: Object estimator computation. (a) The orientation estimator is built in the support region R using the oriented gradients. Each gradient casts a vote for one orientation bin b which is weighted by its magnitude. From the resulting histogram, the orientation of the object ϕ is chosen as the mode of the histogram. (b) Any local orientation histogram can be calculated efficiently by means of the integral histogram. For this example, the number of orientation bins is set to six.

gradients. Basically, this histogram is a distribution of gradient orientations that allows us to parameterize the object orientation by its mode, since it is the most frequent orientation value and because the gradient orientation changes covariantly with in-plane rotations. Refer to Fig. 3.9(a) to see a graphical example.

For the computation of this histogram, firstly image gradients are obtained via Prewitt operators over the input image I,

$$\lambda_y(x,y) = I(x,y) * P_y, \quad \lambda_x(x,y) = I(x,y) * P_x, \tag{3.13}$$

where λ_y and λ_x are the gradient image maps, P_y and P_x are the Prewitt operators for axes y and x, respectively, and $*$ denotes convolution. Using these gradient maps, the gradient orientation ψ is calculated within the support image region with

$$\psi(x,y) = \arctan \frac{\lambda_y(x,y)}{\lambda_x(x,y)}, \quad \forall (x,y) \in R. \tag{3.14}$$

Subsequently, this orientation ψ is discretized into m orientation bins with the aim of constructing an m-dimensional gradient orientation histogram for the region R. To this end, each gradient casts a vote for one orientation bin that is weighted by its magnitude $F(x, y)$. This can formulated as:

$$F(x,y) = \|[\lambda_y(x,y), \lambda_x(x,y)]\|_2, \tag{3.15}$$

$$Hist_R(b) = \sum_{\psi(x,y)\to b} F(x,y) \quad \forall (x,y) \in R, \tag{3.16}$$

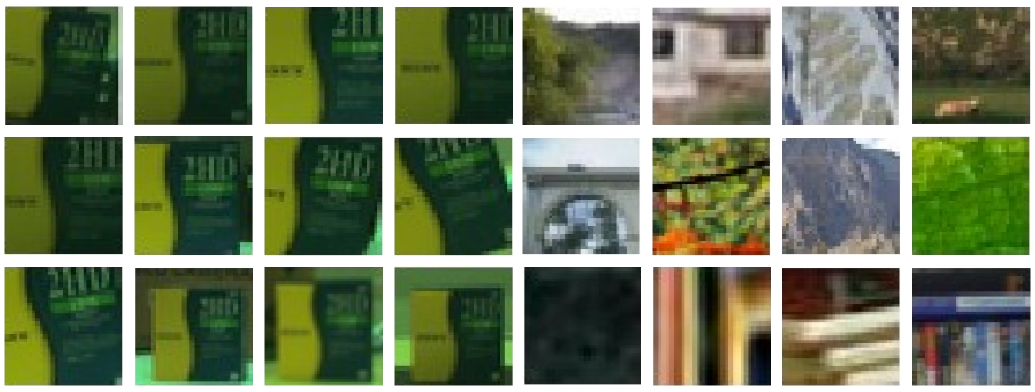
being $Hist_R(b)$ the value at bin b in the orientation histogram for image region R. This histogram computation is depicted in Fig. 3.9(a) and the object orientation is chosen as the mode of this distribution.

This idea is similar to the methodology used in the SIFT descriptor [49] to compute the orientation of interest points. In that case, each gradient casts a vote that is weighted by a Gaussian function centered at the support region in order to give more importance to gradients near to the center of keypoint. In our case, we remove such process because all gradients are equally important for the estimation of the object orientation. As objects or images may be highly structured, the orientation histogram may be easily multimodal. For that case, we follow the same convention as SIFT features: for any peak in the histogram greater than 80% the size of the mode, a new orientation is considered.

Determining the orientation over the image using a sliding window is a costly process that implies computing local histograms for every support region. To sort out this drawback, we add an integral histogram [68]. This histogram representation is basically an extension to the integral image that allows to compute any histogram in constant time independent of its size and location. Its main cost lies in computing the integral histogram that depends on the number of bins in the histogram. The integral histogram is explained in Sec. 2.8. In this case, the feature used is the gradient orientation. Fig. 3.9(b) shows an example for the computation of a local histogram using the integral histogram.

3.5 Experiments

The proposed object detection method is validated in this section over a collection of images with a specific object under rotations in the plane. The evaluation is carried out in terms of detection –recognizing and localizing the object– and its robustness to mild occlusions.



(a) (b)

Figure 3.10: Training image samples. (a) Positive samples used to train the boosted classifier. Shift, scale and orientation changes are added to images to increase the robustness of the object detector. (b) Negative images collected from images without the target object.

To train the object classifier –a CD box in this case– a set of training images is generated. This set has 5250 negative images and 1100 positive images. Negative images were collected as random patches in background images, whereas positive samples are synthetic images that are generated after adding affine image distortions, translations, rotations and scaling. Object translations reach 5 pixels in all directions. Scaling of the object images goes up to 20% of the original image size, and rotated images reach ±10 degrees. Fig. 3.10 shows some sample images.

The object classifier is constructed using 1000 weak classifiers, each one with an associated oriented feature. The orientation estimator is computed by considering 36 gradient orientations. This gives an orientation distribution with 36 elements that is calculated efficiently with the integral histogram.

Fig. 3.11 shows some frames of the live sequence in which the trained object is recognized. At some point, the object is detected at multiple neighboring locations, fact indicated by the repetitive superimposed squares on the top-left frame given that a non-maxima suppression process is not utilized. Also partial occlusions are taken into account in the image sequence.

To evaluate local orientation accuracy, the proposed detector has been tested using two approaches for determining the local orientation. The first one is based on Gaussian

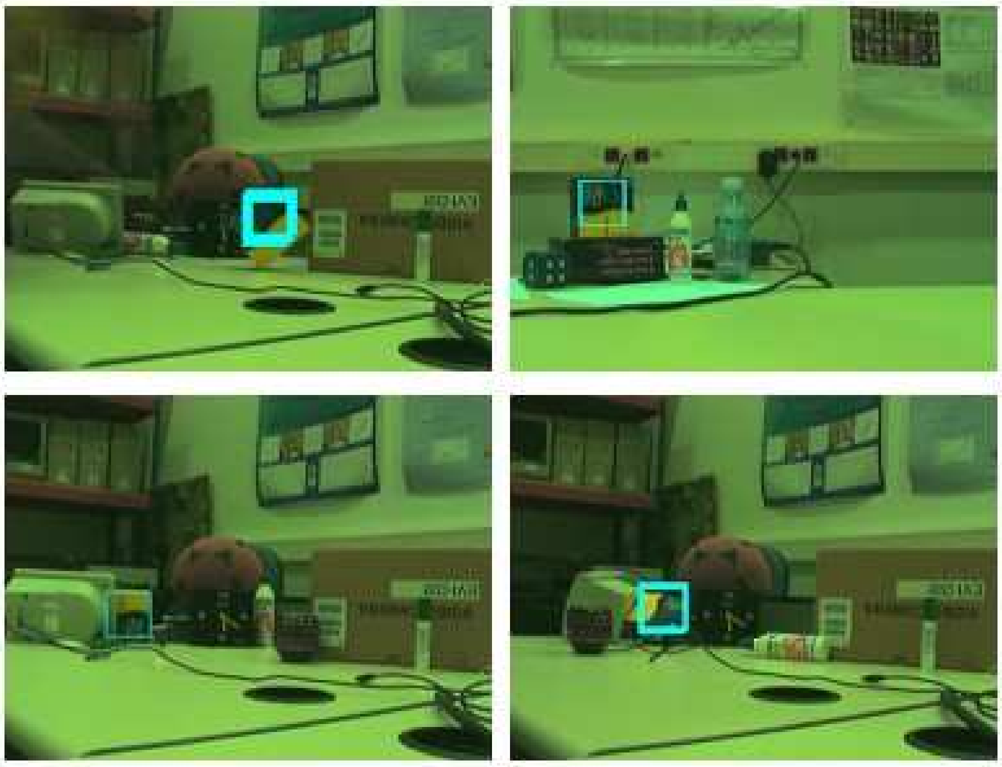


Figure 3.11: Detection of the specific object. Some object detections are shown in spite of orientation, scale and location changes. Cyan rectangles indicate true positive detections.

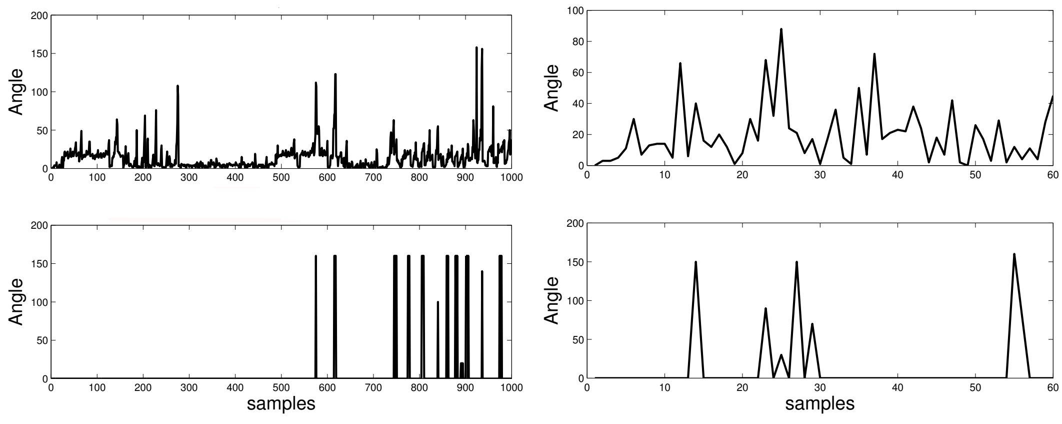
derivatives. More specifically, it computes the ratio of first order Gaussian derivatives G'_x and G'_y to compute the object orientation according to:

$$\phi = arctan \frac{I * G'_y}{I * G'_x}, \tag{3.17}$$

where I is the local image region.

The second approach is the proposed estimator. It consists on using the mode of the gradient-based orientation histogram as the local orientation ϕ. In this case, all gradients inside the region contribute to the construction of the distribution of gradient orientations, and hence, it is more robust to partial occlusions and object variations.

To illustrate the performance of both approaches, two experiments have been performed. The first one is concerned with the robustness when translation and scale changes are applied. The second one measures the robustness of both methods to partial occlusions. The orientation estimation error is computed as the absolute error



(a) (b)

Figure 3.12: Local orientation evaluation. (a) Object orientation error in terms of translation and scale variations. (b) The orientation error is measured under mild occlusions. The plots at the top row correspond to derivative-based approach, whereas at the bottom row to the approach based on the gradient-based histogram.

between the object orientation in the synthetic sample and the estimated orientation given by the tested approach.

For the first experiment 1000 sample images containing the object are used. In the first 600 images the object has translation changes in all directions –5 pixels–. The remaining 400 images contain the object with scale variations. The error curves of this experiment, for the addressed estimation approaches, are shown in Fig. 3.12(a). According to the plots, computing the local orientation using the histogram deteriorates more with scale changes than computing the gradient over the entire image region. Conversely, its performance over translation changes is much better.

For the second experiment, 60 images containing the object under mild occlusions are used for the evaluation, see Fig. 3.12(b). The plots indicate that the proposed approach is more robust to occlusions given that the mode is a nonlinear filter focusing on the most repetitive orientation in the object. Therefore, this technique is much more reliable in the presence of small occlusions in comparison to the derivative-based method.

3.6 Summary

In this chapter, an efficient method for the detection of specific objects was proposed. This method is capable of dealing with object variations such as changes of location, scale or in-plane orientations. With the objective of having a rapid evaluation of the object classifier the method makes extensive use of integral images and Haar-like features. Unlike other works using these techniques, we can deal with in-plane rotations in a very simple and efficient way by introducing fast oriented features and an orientation estimator.

Our proposed method outperforms classical works that use Haar-like features by incorporating features that can be oriented to any given orientation with low computational cost. Classical methods are only able to capture horizontal and vertical contours, or some complex features at the expense of an increased computational effort. Our features, by contrast, can be computed in constant time thanks to the integral image and steerable filters. We approximate Gaussian derivatives with Haar-based filters to reduce the cost of their computation. In our experiments, this approximation has shown to be reliable when feature extraction is carried out.

From other perspective, this type of features allows to enrich the feature pool used to train the object classifier. The result is that more informative and discriminative features with which to describe objects are obtained. From this large feature pool, AdaBoost is used for feature selection and to construct the object classifier. It yields a robust and discriminative classifier that is used for recognizing objects in images. This classifier consists of a weighted combination of weak classifiers where each one of them is based on a Haar-like feature.

As the object may have orientation changes in the image plane, an orientation estimator is also proposed. It reduces the computational cost of having to test the learned boosted classifier for multiple orientations. This gradient-based estimator is constructed by a simple procedure with the help of an integral histogram. Once the orientation estimations have been generated, the boosted classifier is rotated according to these estimations and tested to determine if the current image location contains an object instance.

Chapter 4

Boosted Random Ferns for the Efficient Detection of Objects

In this chapter, Boosted Random Ferns –BRFs– are proposed for the computation of a very discriminative, efficient and robust classifier that can be used for the problem of recognizing and localizing object categories in complex scenes containing cluttered background, intra-class variability, lighting changes and occlusions. Basically, BRFs are a classifier that is constructed via a boosting algorithm –i.e, AdaBoost– with the aim of combining, in an iterative and supervised process, the most discriminative Random Ferns –RFs– into a final hypothesis. Each RF consists on a set of simple comparisons that are evaluated over image cues. They capture the appearance of image structures by means of the co-occurrence of their feature outputs –comparisons–. Since RFs are made of binary comparisons, they can be computed very fast and thus constitute an efficient and effective detection method. The results reported in this chapter have been presented in [92, 94].

4.1 Introduction

The problem of detecting object categories in images is known to be very challenging and needs to address several issues such as large intra-class object variations, changes in object pose, cluttered background or lighting changes. Many recent methods have shown a remarkable success when they use machine learning techniques such as boosting [4, 21, 39, 58, 77, 95, 104] or support vector machines [13, 16, 38, 60, 65]. Besides, some of these methods use a vocabulary of visual words [61] in order to obtain a compact representation of local appearance with which to describe object components and to perform their matching. This vocabulary is constructed from the computation of feature descriptors and is usually followed by a clustering algorithm, namely k-means [50] or agglomerative clustering [43]. The most common descriptor for that task is the SIFT descriptor proposed by Lowe [49]. It captures local image texture by means of a histogram of oriented gradients –HOGs– around interest points. Orientation, scale and affine invariance of this descriptor is achieved by a series of image processing steps that are carried out in the neighborhood of interest points [48, 49, 56].

Randomized Trees –RTs– [45], on the other hand, have shown that keypoint matching can be carried out as a generic classification technique where initial processing steps for insensitivity of keypoints to image distortions are avoided. This fact gives a fast and efficient method for keypoint classification that relies on the co-occurrence of simple intensity-based comparisons. Recently, a novel and non-hierarchical structure named Random Ferns –RFs– was proposed by Ozuysal *et al.* [64] that replaces the Randomized Trees. RFs are simpler and much faster than RTs but keep the same classification performance as them [9, 64]. Both Random Ferns and Randomized Trees have demonstrated similar classification results with approaches based on the SIFT descriptor but with less computational demand.

RTs and RFs have proven to perform a very fast classification thanks to the nature of their features, simple intensity-based comparisons over images. Moreover, they have reported successful results because the co-occurrence of their feature outputs gives a powerful description with which to characterize keypoints. A related descriptor with those methods is the local binary pattern –LBP– presented by Ojala *et al.* in [32, 62]. It is a gray-scale invariant operator that measures image texture using the co-occurrence of pixel comparisons arranged in a local neighborhood. Because of its computational simplicity, it can be used in real-world applications for texture analysis and object recognition. Similarly, Mita *et al.* [58] proposed to use the co-occurrence of feature sets as weak classifiers in a boosting-based approach. In that work, they showed that feature co-occurrence improves the detection rates in contrast to using only single Haar-like features. The proposed method was validated for the detection of faces and hands despite that those categories have strong intensity patterns that ease their classification, like eyes tend to being darker than cheeks.

Although the binary features used in RFs and RTs are usually computed over image intensities, either in gray-scale levels or in color channels [45, 64, 78], recent works have proposed to compute these features over other cues like HOGs with the objective of capturing spatial variations and to gain robustness to lighting changes [75, 92]. However, this is at the expense of computing the HOG over the whole image in a previous step. To reduce the computational burden that this step generates, integral images are used for the efficient computation of HOGs [68, 98].

The proposed method relies in building a discriminative classifier by means of the boosted combination of RFs which are computed in the HOG space. The aim of using a boosting algorithm is to extract a set of the most discriminative and prominent RFs to assemble them into the object classifier. This is contrary to the original work of RFs [64] where these features were chosen randomly. In addition, and in contrast to other works that use a vocabulary of appearances [17, 38, 55, 65], the proposed method relies on a simpler procedure based on sets of binary features that are densely computed over local HOGs. The resulting classifier is therefore a very discriminative and fast classifier that is capable of detecting objects in spite of high intra-class variations, different illumination conditions, partial occlusions and cluttered background.

The rest of this chapter is organized as follows. Sec. 4.2 shows the formulation and computation of BRFs by means of local features over the HOG space and a boosting algorithm. Differences with the original RFs are also discussed. Sec. 4.3 is concerned with the experimental validation of BRFs over some benchmark datasets where the proposed method achieves competitive results but being more efficient and straightforward. Finally, the chapter is summarized in Sec. 4.4.

4.2 Boosted Random Ferns

4.2.1 HOG-based Features

The features used for constructing the RFs and the object classifier are described in this section. Basically, they are local features whose output is a Boolean value that

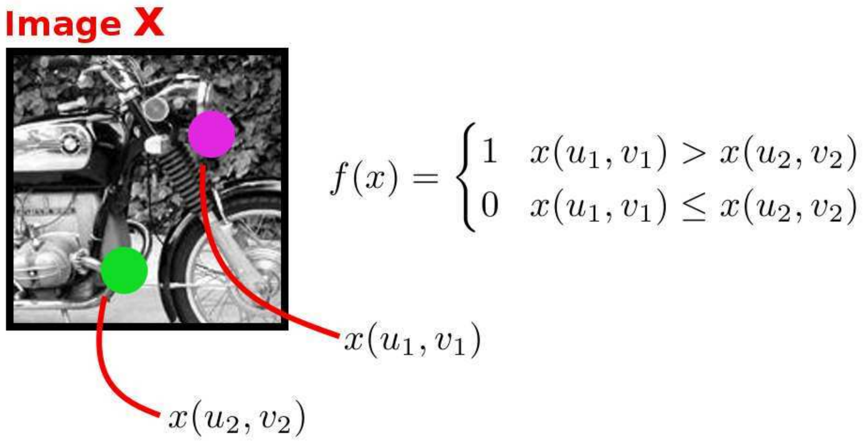


Image X
$f(x) = \begin{cases} 1 & x(u_1,v_1) > x(u_2,v_2) \\ 0 & x(u_1,v_1) \leq x(u_2,v_2) \end{cases}$
$x(u_1,v_1)$
$x(u_2,v_2)$

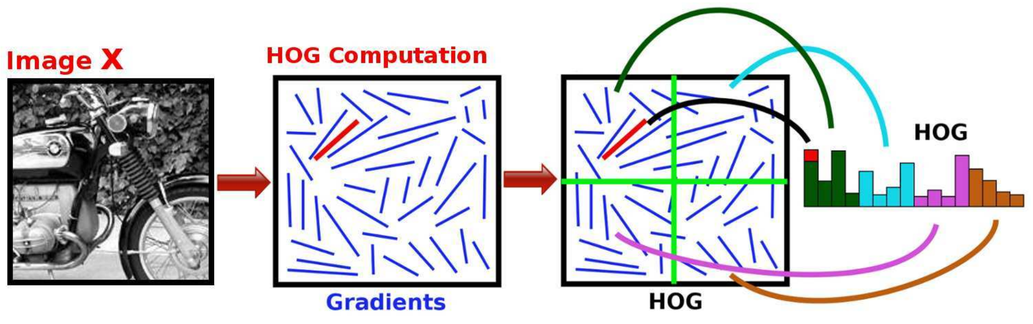
Figure 4.1: Intensity-based feature. This sort of features is computed from binary comparisons between two different pixel intensities whose locations (u, v) are chosen at random.

f	Binary feature
x	Image sample
(u, v)	Image pixel coordinates
$x(u, v)$	Image pixel intensity at coordinates (u, v)
HOG_x	Histogram of oriented gradients computed in image x

Table 4.1: Notation for image features.

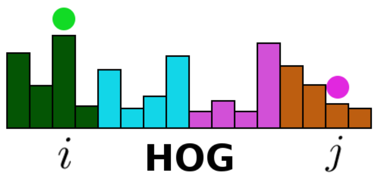
depends on the signed comparison between two image features. Classic methods commonly calculate this sort of features over the image intensity space [45, 64, 78]. This is exemplified in Fig. 4.1 where the feature output is defined by the signed comparison between two pixel intensities –colored dots– whose locations have been chosen randomly. Notation for this section is shown in Table 4.1 and Table 4.2.

We propose instead to compute these binary features over the HOG space with the aim of being robust to lighting changes and small image distortions, but keeping their simplicity and fast computation. This idea was used by Schroff *et al.* in [75]. In that work, features are computed for several distinct image cues, including the HOG space, and then combined into a final object hypothesis. However, the robustness to lighting changes and image variations is at the expense of computing a HOG over whole image. This fact increases the computational cost and, hence, some integral images are required for keeping efficiency [68, 98].



(a)

$$f(x) = \begin{cases} 1 & HOG_x(i) > HOG_x(j) \\ 0 & HOG_x(i) \leq HOG_x(j) \end{cases}$$



(b)

Figure 4.2: HOG-based feature. (a) The proposed features are computed over local HOGs. The computation of HOG is carried out by calculating gradients, where each one casts a vote for a spatial and orientation bins. Votes are weighted according to gradient magnitude. The resulting descriptor is a concatenation of local and adjacent distributions of oriented gradients. (b) Once the HOG is computed, features are calculated from binary comparisons between two different bins –colored dots– of HOG.

As commented before, the feature yields a binary output that allows to map an image sample x to a boolean space in the form,

$$f : x \rightarrow \{0, 1\}\,, \tag{4.1}$$

by the comparison between two image feature values. For the present case, this comparison test is carried out over the HOG space, that is, the signed difference between two HOG bin values. Therefore, the HOG-based features can be defined as follows:

$$f(x) = \begin{cases} 1 & HOG_x(i) > HOG_x(j) \\ 0 & HOG_x(i) \leq HOG_x(j) \end{cases}\,, \tag{4.2}$$

where i and j are the histogram bin locations, see Fig. 4.2. In the figure, a HOG is computed using 2×2 spatial grid and four gradient orientations. This yields a local descriptor –histogram– with 16 elements.

In summary, the proposed features are simple binary comparisons evaluated over HOGs that measure the layout of oriented gradients in a local image region. Although one single binary feature is not discriminative enough, we shall see in next sections that the co-occurrence of several of these simple features gives a powerful and suitable tool for describing object components.

4.2.2 Formulation

In contrast to the original formulation of the RFs for keypoint matching, the Ferns expression is written in terms of likelihood ratios between the target object class C and a background class B. This allows to seek for the feature combinations that maximize this ratio by means of the boosting algorithm, and to extract the most prominent features to discriminate the object category from the non-object category –background–.

The goal is to model the logarithmic ratio between the posterior probabilities of the object and background classes given a set of N_f binary features (f) with the purpose of building a two-class classifier $H(x) = \{+1, -1\}$, defined by

$$H(x) = sign\left(\log \frac{P(C|f_1(x), f_2(x), ..f_{N_f}(x))}{P(B|f_1(x), f_2(x), ..f_{N_f}(x))} - \beta\right), \tag{4.3}$$

where x is a sample image and β is the classifier threshold. If $H(x) = +1$, the image x is assigned to the object class, otherwise, it is assigned to the background one.

Similarly to the formulation of RFs –Sec. 2.1–, this classifier can be expressed using the Bayes rule, where by removing the evidence factor $P(f_1(x), f_2(x), ..f_{N_f}(x))$, common for both classes, and assuming uniform prior class probabilities $P(C) = P(B)$, the logarithmic ratio of probabilities can be written as:

$$\log \frac{P(C|f_1(x), f_2(x), .., f_{N_f}(x))}{P(B|f_1(x), f_2(x), .., f_{N_f}(x))} = \log \frac{P(f_1(x), f_2(x), .., f_{N_f}(x)|C)}{P(f_1(x), f_2(x), .., f_{N_f}(x)|B)}. \tag{4.4}$$

By computing Ferns over the set of binary features, $\mathcal{F}_r = \{f_1^r, f_2^r, ..f_M^r\}$, the object classifier becomes a combination of logarithmic Fern probability ratios,

$$H(x) = sign\left(\sum_{r=1}^{R} \log \frac{P(\mathcal{F}_r(x)|C)}{P(\mathcal{F}_r(x)|B)} - \beta\right), \tag{4.5}$$

where R is the number of Ferns. These Ferns are constructed randomly and each one has M binary features, with $M = N_f/R$.

$\mathcal{F}$	Random Fern
f	Fern (binary) feature
x	Image sample
X	Sample space
$H(x)$	Object classifier
T	Number of weak classifiers
N_f	Number of binary features
R	Number of Random Ferns
M	Number of features (f) per Fern
z	Fern observation (output) value
Z	Dimension of Fern output space
Q	Bhattacharyya coefficient
g_t	Location of Fern t
C	Object (positive) class
B	Background (negative) class
β	Classifier threshold
N	Number of training samples
y_n	Class label of image sample n
D	Distribution of sample weights
ϵ	Smoothing factor

Table 4.2: Notation for Boosted Random Ferns.

Each Fern captures the local appearance of objects because it encodes the co-occurrence of their M features which are tested in a local HOG. The response or output of each Fern, called Fern observation from now, is represented by the combination of their Boolean feature outputs. For instance, the observation z_r of a Fern $\mathcal{F}_r$ made of $M = 3$ features with binary outputs $0, 1, 0$, would be $(010)_2 = 2$. In other words, each Fern maps 2D image samples to an $Z = 2^M$-dimensional space,

$$\mathcal{F} : x \to z, \quad x \in X, \quad z \in \{1, 2, .., Z\}. \tag{4.6}$$

Since the probability of each Fern $\mathcal{F}_r$ is given by its observation z_r conditioned to each class, the object classifier can be written as:

$$H(x) = sign\left(\sum_{r=1}^{R} \log \frac{P(\mathcal{F}_r(x) = z|C, g_r)}{P(\mathcal{F}_r(x) = z|B, g_r)} - \beta\right), \tag{4.7}$$

where g_r, $g \in \mathbb{R}^2$, is the image spatial location where the Fern $\mathcal{F}_r$ is evaluated, measured from the image center. Table 4.2 shows the notation used for the formulation and computation of BRFs.

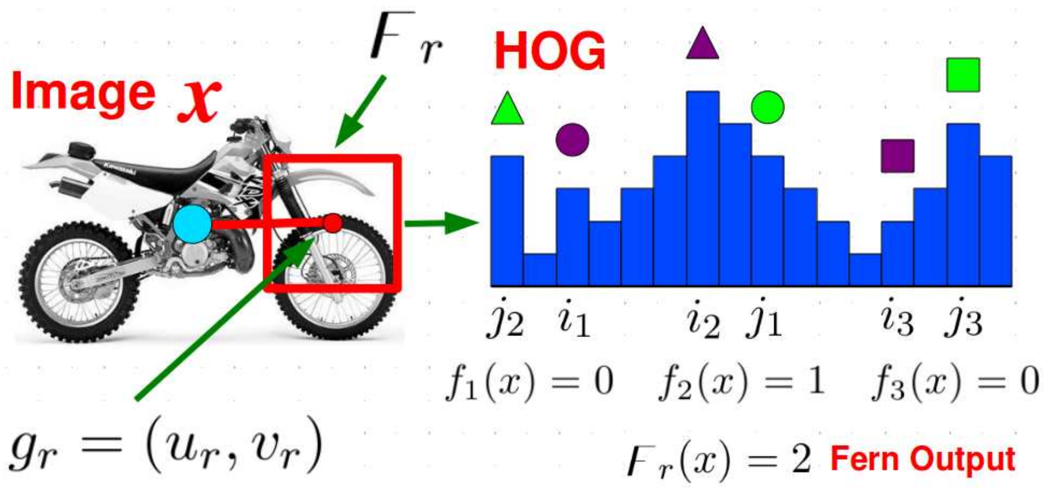


Image x
F_r
HOG
j_2 i_1 i_2 j_1 i_3 j_3
$f_1(x) = 0$ $f_2(x) = 1$ $f_3(x) = 0$
$g_r = (u_r, v_r)$
$F_r(x) = 2$ Fern Output

Figure 4.3: Random Fern computation. Binary features forming the Fern F_r are computed over the local HOG –red square–. The distance between this region and the object center is given by $g_r = (u_r, v_r)$. The RF captures the appearance of object components by means of the co-occurrence of gradient-based features.

Fig. 4.3 shows an example where one Random Fern F_r is computed over a local HOG. This HOG is constructed from a local image region –red square– where HOG-based features –colored and paired symbols– are then computed. The changes on the histogram are encoded by the Fern observation z_r. This observation value represents a specific signature of the histogram.

4.2.3 Computation of BRFs

We want to build the object classifier $H(x)$, yielding the most discriminative sets of Ferns F and locations g that maximize Eq. 4.7. This is achieved by means of the Real AdaBoost algorithm –Sec. 2.3– that iteratively assembles weak classifiers and adapts their weighting values [73].

To build the classifier, Real AdaBoost –RAB– uses an input set of training samples $(x_1, y_1)...(x_n, y_n)...(x_N, y_N)$, where x_n corresponds to one sample belonging to sample space X, and y_n is its class label, $Y = \{+1, -1\}$, that indicates the object and background classes, respectively. The algorithm also uses a distribution of weights over training samples D that is updated after each boosting iteration. This distribution

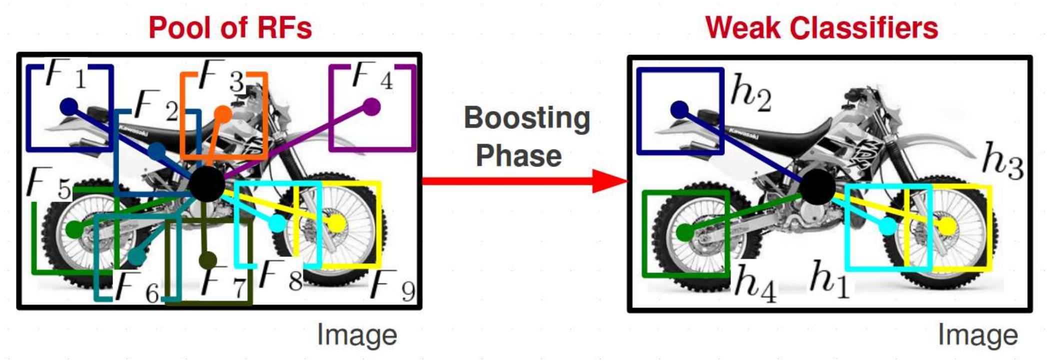


Figure 4.4: Boosted Random Ferns. From a large pool of RFs, computed for multiple image locations, the most discriminative ones for classifying the target object category are chosen via a boosting phase.

measures the difficulty of training samples during boosting. Initially, all weights are set equally, but on each round, the weights of misclassified samples are increased so that the algorithm is forced to focus on such hard samples in the training set.

In order to choose the most discriminative object features, a large pool of RFs is constructed in a previous step. For every image location, multiple Ferns are computed at random and each one represents a potential weak classifier. After the boosting phase, the algorithm selects the best weak classifiers that build up the strong classifier. This can be illustrated in Fig. 4.4 where a few RFs are shown.

Hence, the object classifier consisting of T weak classifiers can be expressed as:

$$H(x) = sign\left(\sum_{t=1}^{T} h_t(x) - \beta\right), \tag{4.8}$$

where each weak classifier h_t is defined by the logarithmic probability ratio of Fern $\mathcal{F}_t$ conditioned to both classes,

$$h_t(x) = \frac{1}{2}\log\left(\frac{P(\mathcal{F}_t(x)|C, g_t) + \epsilon}{P(\mathcal{F}_t(x)|B, g_t) + \epsilon}\right). \tag{4.9}$$

The parameter ϵ is a smoothing factor that avoids infinite values and whose value is very small. RAB seeks to maximize this ratio during the learning step by evaluating different RFs and keeping the most discriminative ones. This is done at each boosting iteration t by calling a weak learner to compute the most discriminative weak classifier according to the distribution of sample weights D_t. The weak classifier, defined by a

Algorithm 4 Boosted Random Ferns

```
1:  Given a number of weak classifiers $T$ and a dataset consisting of $N$ image samples
    labeled $(x_1, y_1)...(x_n, y_n)...(x_N, y_N)$, where $y_n \in \{+1, -1\}$ is the label for object
    and background classes, respectively.
2:  Construct a pool of $R$ Random Ferns densely computed over the whole image.
3:  Initialize the sample weights $D_1(x_i) = 1/N$ where $i = 1, 2, ...N$.
4:  for $t = 1$ to $T$ do
5:    for $r = 1$ to $R$ do
6:      Using the current distribution $D_t$ and the current Fern $\mathcal{F}_r$ with location $g_r$,
        compute the weak classifier $h_r$ as follows:
        $h_r(x) = \frac{1}{2} \log \left( \frac{P(\mathcal{F}_r(x)|C, g_r)+\epsilon}{P(\mathcal{F}_r(x)|B, g_r)+\epsilon} \right)$
7:      Compute its Bhattacharyya coefficient by
        $Q_r = 2 \sum_{z=1}^{Z} \sqrt{P(\mathcal{F}_t = z|C, g_r) P(\mathcal{F}_t = z|B, g_r)}$
8:    end for
9:    Select the weak classifier $h_r(x)$ that best discriminates the object samples from
      background ones using the Bhattacharyya coefficient.
10:   Update the sample weights.
      $D_{t+1}(x_n) = \frac{D_t(x_n) \exp[-y_n h_t(x_n)]}{\sum_{n=1}^{N} D_t(x_n) \exp[-y_n h_t(x_n)]} \quad n = 1, 2, .., N$
11:   Aggregate the computed weak classifier $h_t$ to the strong classifier $H$.
      $H(x) \leftarrow h_t$
12: end for
13: Final strong classifier.
    $H(x) = sign \left( \sum_{t=1}^{T} h_t(x) - \beta \right)$
```

Random Fern $\mathcal{F}_t$ and its location g_t, that maximizes the classification power in terms of the Bhattacharyya coefficient Q is selected. This coefficient is calculated as:

$$Q = 2 \sum_{z=1}^{Z} \sqrt{P(\mathcal{F}_t = z|C, g_t) P(\mathcal{F}_t = z|B, g_t)}, \tag{4.10}$$

where the Ferns distributions for object and background classes are computed by

$$P(\mathcal{F}_t = z|C, g_t) = \sum_{n:\mathcal{F}_t(x_n)=z \wedge y_n=+1} D_t(x_n), \tag{4.11}$$

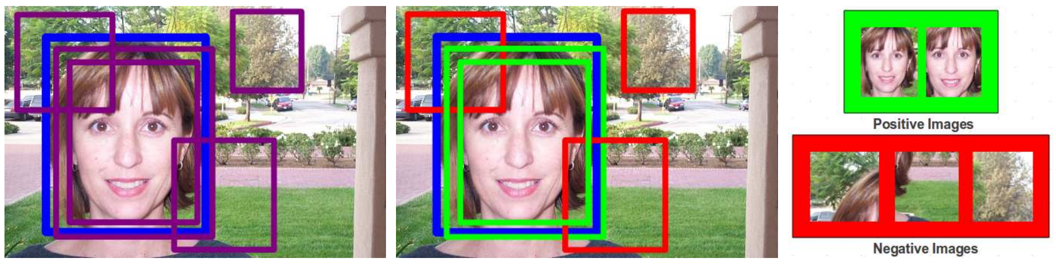
$$P(\mathcal{F}_t = z|B, g_t) = \sum_{n:\mathcal{F}_t(x_n)=z \wedge y_n=-1} D_t(x_n), \tag{4.12}$$

with indexes $z = 1, 2, .., Z$ and $n = 1, 2, ...N$. After each boosting iteration the distribution of sample weights D is updated and normalized and the procedure is repeated recursively until a stopping point is achieved –i.e maximum number of weak classifiers–. Alg. 4 shows the pseudocode used to compute this classifier.

4.2.4 Comparison with Random Ferns

Boosted Random Ferns and Random Ferns are quite similar since the former ones are basically RFs that are trained using a supervised learning algorithm –boosting algorithm–. The main differences between both approaches are:

- BRFs are computed using a boosting algorithm –i.e, Real AdaBoost– in order to extract the most discriminative RFs for classifying the object category. In contrast, RFs use a semi-naïve classifier where features are extracted completely at random without a supervision step.
- Each training sample has a weight corresponding to its classification difficulty. This weight is updated after each boosting round according to the classification given by the current weak classifier h_t. Therefore, each BRF is trained to focus on difficult samples that have been classified incorrectly by previous weak classifiers. Contrary, original RFs are trained independently and without sample weights. This yields a more straightforward learning procedure but results in a less robust classifier.
- BRFs are proposed for a two-class separability problem, whereas the RFs are conceived for a multi-class problem. However, the proposed method can be extended to cope with multiple classes by considering a multi-class boosting algorithm –i.e, JointBoosting [87]– to train discriminative features for every object class.
- To compute object-specific RFs, the proposed BRFs use a logarithmic ratio between the Fern probabilities conditioned to both classes, object and background. In this way, the boosting phase attempts to maximize this ratio by selecting and combining the most discriminative weak classifiers against background. Contrary, classic RFs store the probabilities of each Fern conditioned on each class and do not use a background class. In consequence, the computed RFs are less discriminative when compared with the features we propose.
- Classic RFs are the product of multiplying Fern probabilities whereas BRFs are an average of Fern probability ratios. This is an important disadvantage for RFs, because they require that posterior probabilities be trusted. Otherwise, if just one Fern probability is zero the classification might be incorrect.



(a) (b) (c)

Figure 4.5: The bootstrapping phase. (a) Once the classifier is computed, it is tested over training images. (b) According to the ground truth –blue rectangle– and the bootstrapping criterion, Eq. 4.13, current detections –magenta rectangles– are classified as either positive or negative samples. (c) Positive samples correspond to object instances –green rectangles–, whereas negative ones are background regions –blue rectangles–.

4.2.5 Bootstrapping the BRFs

Boosted Random Ferns have shown to be a good alternative for detecting object categories in an efficient and discriminative way [92]. Although this method has shown impressive results in spite of its simplicity, its performance is conditioned to the quantity and quality of its training data. This is because boosting algorithms require a large number of training samples and because each Fern has an observation distribution whose size depends on the number of features forming the Fern. The larger this distribution is, the more training samples are needed. This dependence is shown in the experimental validation of BRFs, refer to Sec. 4.3. This problem becomes critical in cases where datasets have a small number of images for training –below 50–. To overcome this limitation, some methods introduce image transformations over the training data in order to enlarge the set of positive images, and to collect a large number of random patches over background images as negative samples.

In contrast, we show in this section that adding an iterative bootstrapping phase during the learning of the object classifier increases its detection rates because additional positive and negative samples are collected –bootstrapped– for retraining the boosted classifier. After each bootstrapping iteration, the learning algorithm is concentrated on computing more discriminative and robust features since the bootstrapped samples extend the training data with more difficult images, see Fig. 4.5.

The procedure is carried out as follows: given the training data, consisting of sets of positive and negative samples, the BRFs are first computed using those initial samples. Once the classifier has been computed, it is tested over the same training images with the aim of extracting new positive and negative samples. As a result, a more challenging training dataset with which to train the following classifiers is obtained. The extraction of new training samples is shown graphically in Fig. 4.5.

The criterion for selecting the samples is based on the overlapping rate r between detections –bounding boxes– and the image ground truth given by the dataset. If that rate is over a defined threshold, 0.3 by default, those detections are considered as new positive samples. Otherwise, they are considered as false positives and are aggregated to set of negative images. The overlapping rate can be written as:

$$r = \frac{|B_{gt} \bigcap B_d|}{|B_{gt} \bigcup B_d|} \tag{4.13}$$

where B_{gt} indicates the bounding box for the ground truth, and B_d is the bounding box for the current detection. This procedure automatically yields a set of new positive samples containing the object under scale changes and shifts since the classifier is tested at every image location and multiple scales. The robustness of the classifier is then increased thanks to object samples with small distortions are considered in the following learning phase. On the other hand, the extracted negative samples correspond to background regions which have been misclassified by the current classifier. Those images force the boosting algorithm to seek out more discriminative Ferns towards object classification in the following iterations.

4.3 Experiments

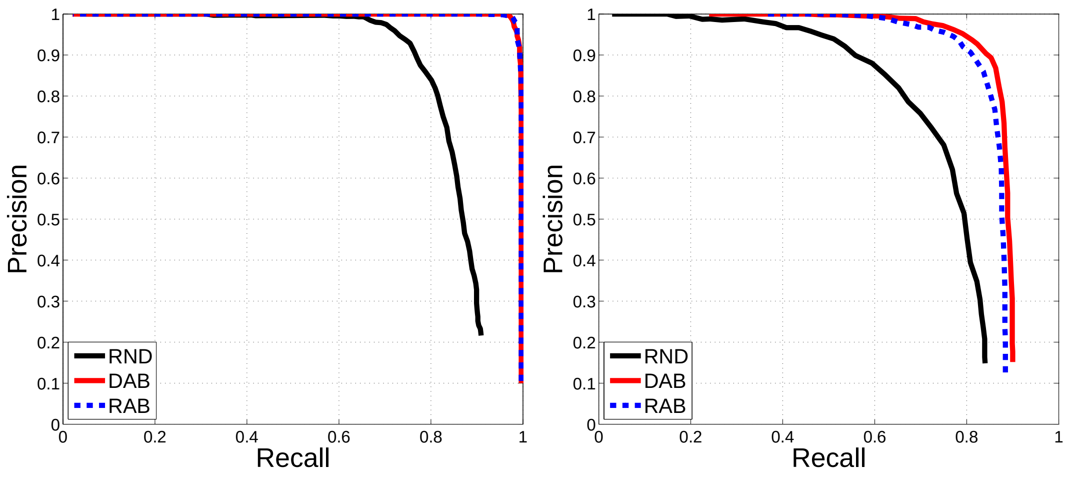
BRFs were evaluated in diverse datasets with the objective of validating and comparing their performance with some successful and recent works.

The datasets used are public and consist of a collection of images of an object category that has been acquired using bounding boxes around specific objects. For this chapter, the addressed datasets are the well-known UIUC car dataset [1], the TUD motorbike dataset [25], and the INRIA horse dataset [18]. In spite of the “simplicity” of the chosen datasets, in comparison with other recent datasets, they allow the comparison with a wide number of detection methods. In this chapter, we are not interested in 3D object recognition or multi-class recognition.

UIUC car dataset [1]. This dataset contains car-side views under difficult imaging conditions such as illumination changes, cluttered backgrounds and mild occlusions. This dataset has two sets of images for testing. The first one has 170 images containing 200 car instances with similar scale to that of the training samples -40 $\times$ 100 pixels-. The second one has 108 images consisting of 139 cars at different scales, varying from 36×89 to 85×212 pixels. This dataset has also a set of training images that is formed by 550 positive images and 500 negative ones. Positive images are images containing different cars facing to the left and right. Contrary, negative images are background images containing diverse things except cars. In this work, the second set of test images has been chosen because it implies more challenging object recognition task with car images at several scales.

TUD motorbike dataset [25]. This dataset consists of 115 images containing 125 motorbike instances under occlusions, different scales and difficult backgrounds. This dataset is included in the PASCAL object recognition database collection [14]. Furthermore, this dataset shows high intra-class variability due to quite different motorbikes models. For training, the Caltech motorbike dataset is used [17]. It has 826 object images and a large number of background samples.

INRIA horse dataset [18]. This dataset has 170 images containing one or more horses. In addition, this dataset has 170 images with background that are used as negative samples. The dataset is challenging because horses appear at several scales, poses, and have out-of-plane rotations. Furthermore, this dataset has some visible horses in images that are not labeled, and thus, they affect negatively the recognition rates. For training, the first 50 positive and 50 negative images are used. The remaining images are utilized for testing the computed object classifier.



(a) Car dataset (b) Motorbike dataset

Figure 4.6: Learning phase. Average detection performances are shown for the addressed datasets: cars (a) and motorbikes (b). Classifiers learned via a boosting algorithms yield better detection rates than those ones learned using random features.

4.3.1 Learning Phase

Three different learning schemes are evaluated. The first two are based on a supervised learning strategy, boosting algorithms, with the aim of selecting the most relevant Random Ferns for classification, whereas the last scheme consists on choosing the Random Ferns at random –RND–, as proposed originally in [64]. The selected boosting algorithms are the Real AdaBoost –RAB– [73] and the Discrete AdaBoost –DAB– [24], see Fig. 4.6. These algorithms have been used in the past by many detection approaches and have demonstrated successful detection rates using a collection of weak classifiers which their computation can be carried out rapidly [34, 39, 58, 98, 99, 100].

To take into account the feature randomness, each object classifier has been learned and tested eight times. From the resulting curves, the average curve is computed. The classification performance is shown in Figs. 4.6 and 4.7. In all cases adding a boosting algorithm to compute the specific object classifier outperforms the classic feature selection mechanism of the original RFs. Both boosting algorithms show comparable results, and from now, the RAB is the algorithm selected for the remainder experiments, and only average values are plotted in the detection curves.

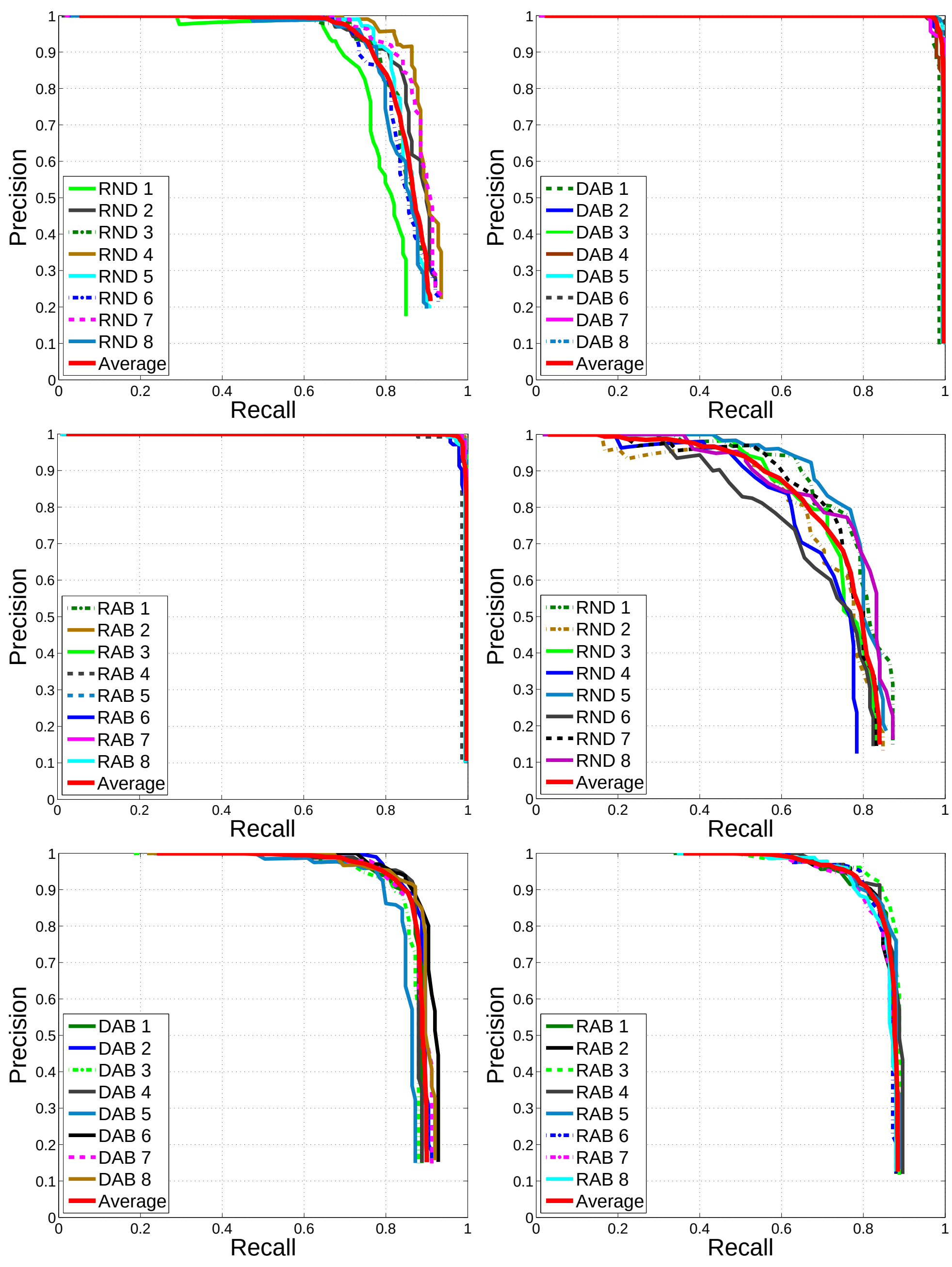
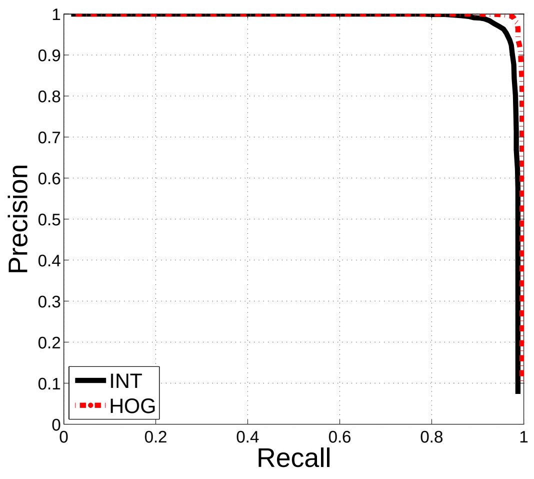
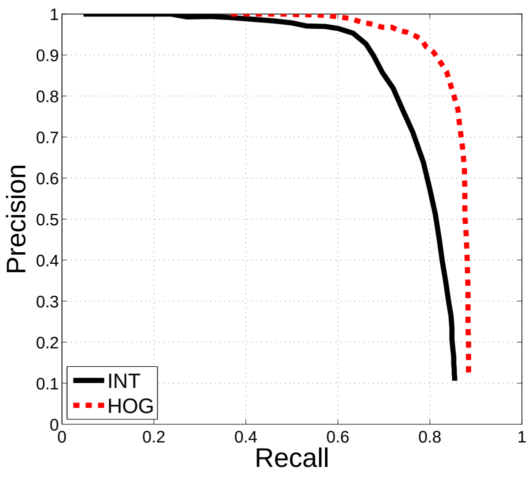


Figure 4.7: Comparison of three learning phases for the selection of RFs. Adding a boosting phase, the BRFs achieve remarkable detection rates over the UIUC car dataset –top row– and the TUD motorbike dataset -bottom row-. By contrast, Random Ferns without a boosting phase -RND- yield considerably lower detection rates. Each object classifier has been learned and tested eight times.



(a) Car dataset



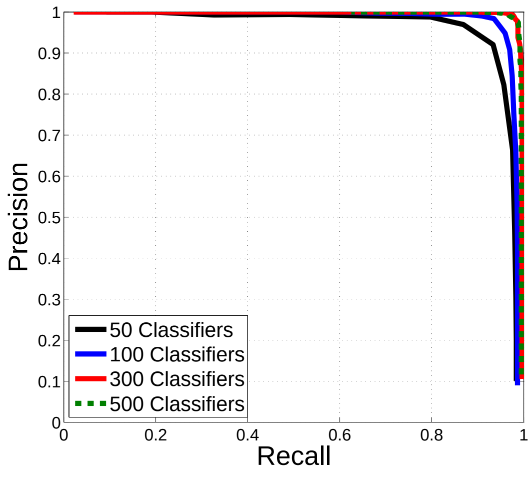
(b) Motorbike dataset

Figure 4.8: Feature space. BRFs are trained using two different feature spaces: HOG and image intensity –INT–. The detection curves for the car and motorbike datasets are shown. In both cases, HOG-based features outperform to features computed over image pixel intensities.

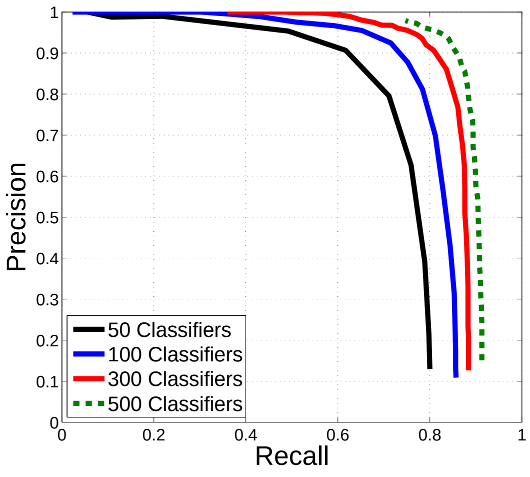
4.3.2 Feature Space

The influence of the feature space over the detection performance of BRFs is measured in this experiment. Toward this end, RFs forming the object classifiers are computed over either local HOGs or image intensities. Here, we wish to measure and to compare HOG-based features against traditional methods which calculate RFs over the image intensity space –INT–. The resulting detection curves are visualized in Fig. 4.8.

Working on the HOG space increases the detection rates because HOGs are more robust to lighting changes and intra-class variability than intensity-based features. Note however that for the car dataset, the classifier using intensity-based features achieves very good results. This is because the images in this dataset have strong intensity patterns that ease object recognition. Some examples are car shadows that appear in most of images and pavement color which is almost constant in all images. BRFs end up learning these patterns and using them for car classification. We can say in consequence that the context is contributing positively to the detection of cars.



(a) Car dataset

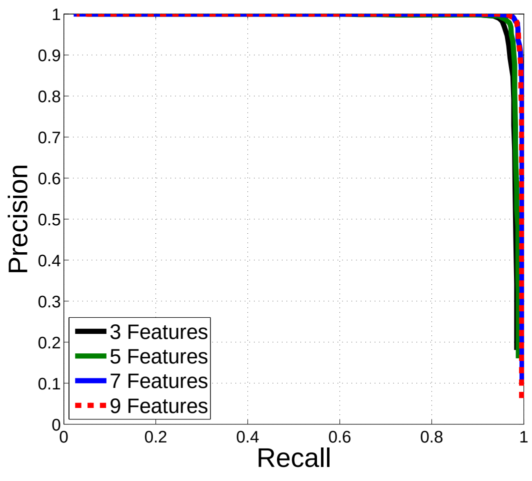


(b) Motorbike dataset

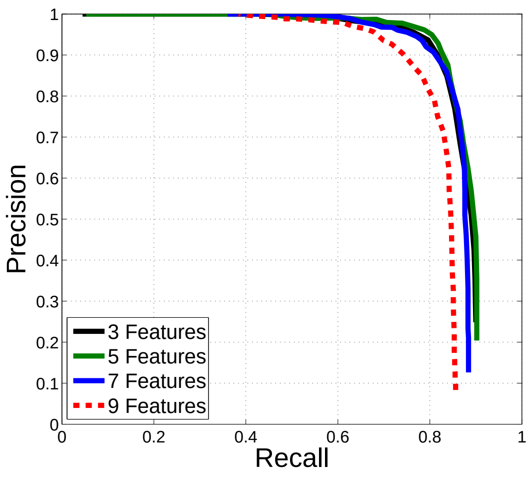
Figure 4.9: Weak classifiers. Detection rates according to the number of weak classifiers in the boosted classifier. (a) Results for the UIUC car dataset. (b) Results for the TUD motorbike dataset. The number of weak classifiers affects positively the detection performance of BRFs.

4.3.3 Number of Weak Classifiers

It is well known that detection approaches based on boosting algorithms require of a large number of weak classifiers to achieve good classification rates because they rely on weak hypotheses. The combination of more weak hypotheses leads to a more robust and stronger classifier. In this experiment, BRFs are evaluated according to the number of weak classifiers allowed to build the boosted classifier $H(x)$. The detection performance for varying number of weak classifiers used are given in Fig. 4.9. Note that as the number of classifiers is increased, BRFs improve the detection performance because more RFs are contributing in the final classification hypothesis. Nevertheless, after 500 weak classifiers are used, the rate change becomes irrelevant. In that case, it is said that the detector has achieved its maximum score value despite still having more weak hypotheses. Although increasing the number of weak classifiers improves detection rates, it implies more feature computation and therefore a larger computational cost. Its choice is a trade-off between the computational cost of the detection phase and its performance.



(a) Car dataset



(b) Motorbike dataset

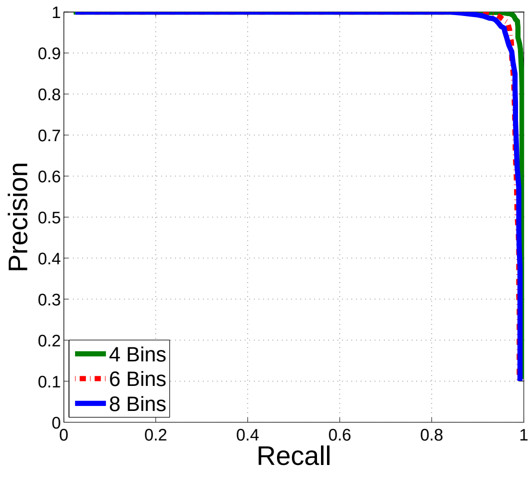
Figure 4.10: Feature co-occurrence. BRFs are learned using Ferns with different number of HOG-based features. Feature co-occurrence improves the detection rates since more features implies a better description of HOGs.

4.3.4 Number of Features

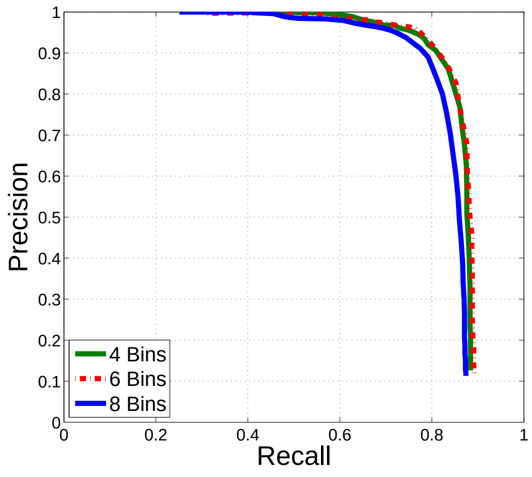
The number of HOG-based features per Fern is also evaluated in order to measure the importance of feature co-occurrence over the performance of BRFs. For this experiment, each object classifier is learned using the same number of weak classifiers –300– but RFs are constructed using a different number of binary features per Fern. The results for the addressed datasets are depicted at Fig. 4.10. It shows that feature co-occurrence improves the detection performance until a saturation point where there are many binary features for a local HOG, and hence, the detection performance of BRFs deteriorates. Contrary, few features over the histogram may not encode the object appearance properly thus giving a poor performance as well.

4.3.5 HOG

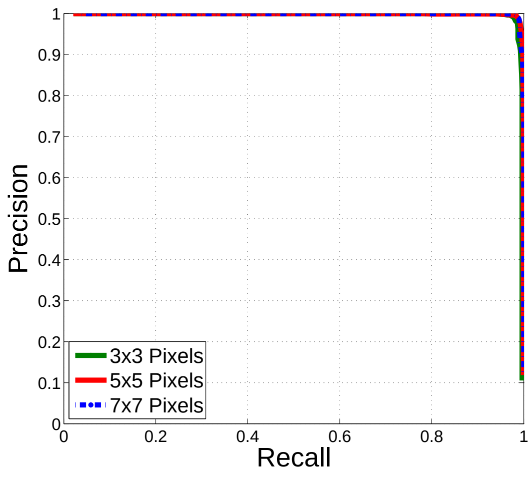
In this experiment, the parameters concerning to the construction of HOGs are evaluated. They include the number of gradient orientations and the size of the HOG cell. Gradient orientations refer to the number of bins in which gradients are discretized, while HOG cell size is the image area where gradient orientation distributions are com-



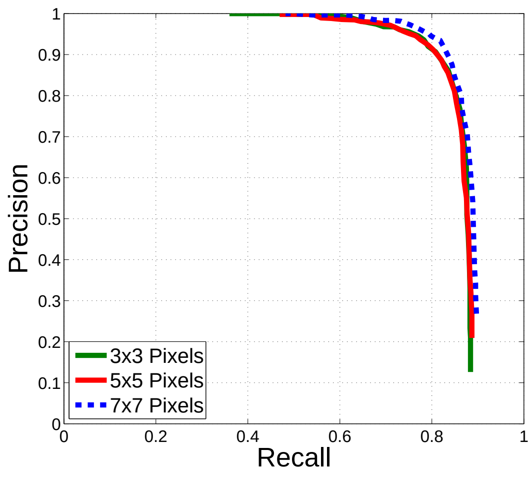
(a) Car dataset



(b) Motorbike dataset



(c) Car dataset

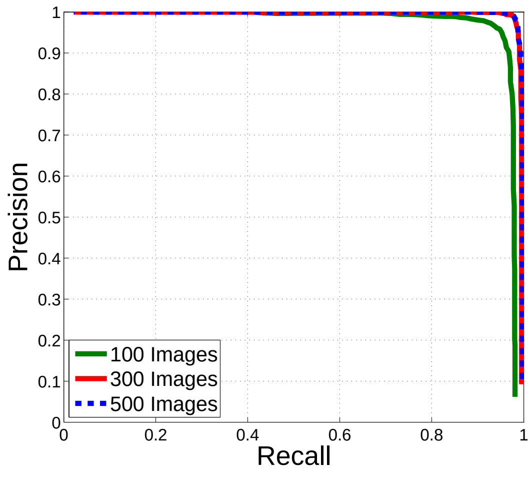


(d) Motorbike dataset

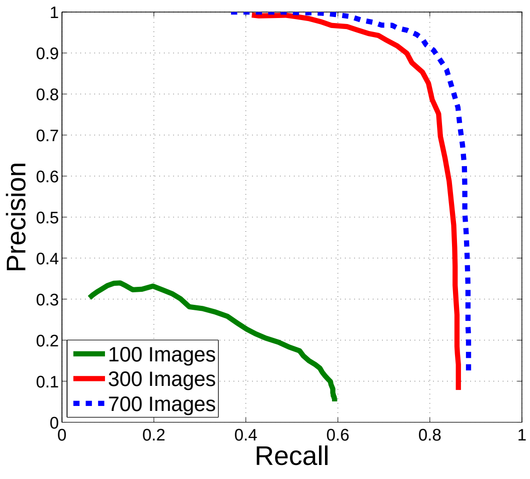
Figure 4.11: The detection performance of BRFs in terms of the parameters concerning to the construction of HOG. (a,b) Detection curves corresponding to the number of gradient orientations. (c,d) Detection curves for different cell sizes.

puted. The cell size is measured in pixels. The spatial concatenation of orientation distributions leads to the resulting HOG, see Fig. 4.2(a).

Fig. 4.11(a-b) shows the performance of BRFs using different number of gradient orientations θ. We can see that the best detection rates are achieved using only four orientation bins. From now, this value is used for the rest of experiments. The detection curves for different cell sizes are reported in Fig. 4.11(c-d). The results are quite similar



(a) Car dataset



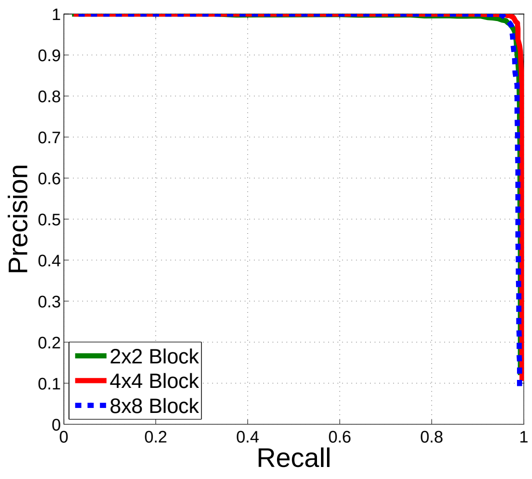
(b) Motorbike dataset

Figure 4.12: Training data. BRFs are trained using different image set sizes. Note that the detection performance of BRFs depends on the quantity of training data. More images leads to better detection rates.

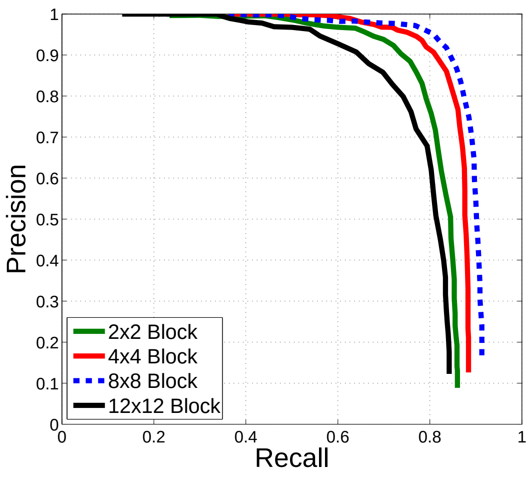
for all cases. This indicates that the performance of BRFs does not rely essentially on the choice of this parameter.

4.3.6 Training Data

Here, BRFs are computed using several sets of training data. That is, sets with different amount of images, both positives and negatives. The aim is to observe the influence of training data size over the detection results. To this end, the BRFs are learned using 300 weak classifiers, seven HOG-based features per Fern and several training image sets. The detection curves for the different cases are visualized in Fig. 4.12. According to the plots, there is a strong dependence between the amount of images in the training data and the performance of BRFs. The more images the dataset has, better detection rates report. This is because the boosting approaches require a large set of images to compute robust classifiers since they are based on a combination of weak hypotheses. Besides, more negative images helps to compute a more discriminative and stronger classifiers against background.



(a) Car dataset



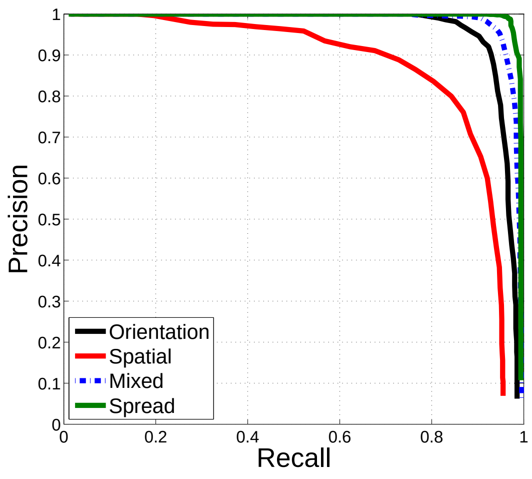
(b) Motorbike dataset

Figure 4.13: Fern size. BRFs are learned and tested using different Fern sizes. The best detection rates are achieved for a compromise between local features and description of relevant object components.

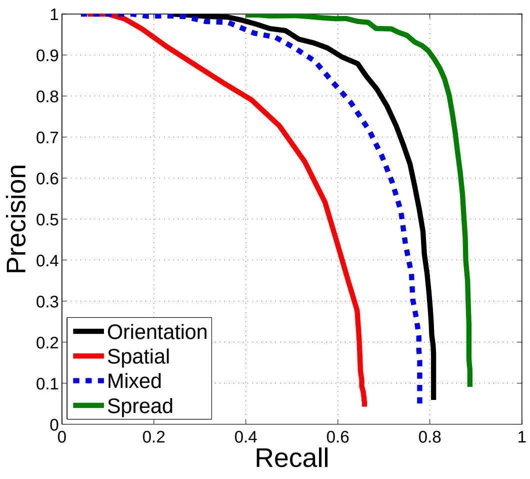
4.3.7 Fern Size

The Fern size is the size of local HOGs where Random Ferns are computed. This parameter has a strong influence over the classification performance of BRFs, see Fig. 4.13, because if the local HOG is large, the feature co-occurrence might not capture local appearance properly, and there would be few binary comparisons over the histogram of oriented gradients. On the other hand, if the HOG size is small it will not capture relevant object components. This behavior can be seen in the detection plots for the car and motorbike datasets shown in Fig. 4.13. The best detection results for both datasets are reported for a Fern size of 4×4 cells and 8×8 cells, respectively. It is important to emphasize that the car and motorbike classifiers were trained using an image size of 12×30 and 18×26 HOG cells.

In summary, the choice of Fern size should be local, relative to image size, with the aim of being robust to partial occlusions, but not too small so as to capture important object components –i.e car wheels–.



(a) Car dataset



(b) Motorbike dataset

Figure 4.14: Fern configuration. Detection performance is measured according to four different Fern configurations. From all configurations, spread features achieve the best detection rates.

4.3.8 Fern Configuration

To show how the layout of RFs inside a local HOG affects the detection performance, four different types of Fern configurations have been evaluated. The first three have in common that feature comparisons are carried out between adjacent HOG cells in spatial and orientation directions, and in a combination of both. These Fern configurations resemble the Haar-based features but computed in the HOG domain [103]. Finally, a spread configuration is proposed in which features are distributed randomly over the whole local HOG. The detection plots for the car and motorbike datasets shown in Fig. 4.14 indicate that spread features outperform all other configurations because they capture important features through the histogram without imposing restrictions. In addition, we see that orientation-based features outperform, in both datasets, spatial-based features. The reason is that comparisons between different orientation channels offer rich and powerful information to describe locally object appearance.

4.3.9 Computational Cost

The computational cost of testing the BRFs over an input image depends basically on the number of weak classifiers and the number of binary features per Fern. Moreover, as the detection phase is performed by a sliding window where the BRFs are tested for every image location and for several scales, the size of input image and the number of image scales also affect the computational cost. The idea of having an image representation consisting of multiple scaled images –pyramid image– is because the classifier has been learned with a fixed size, the trained object size. Hence, BRFs must be tested for several scales in order to cope with changes in object size.

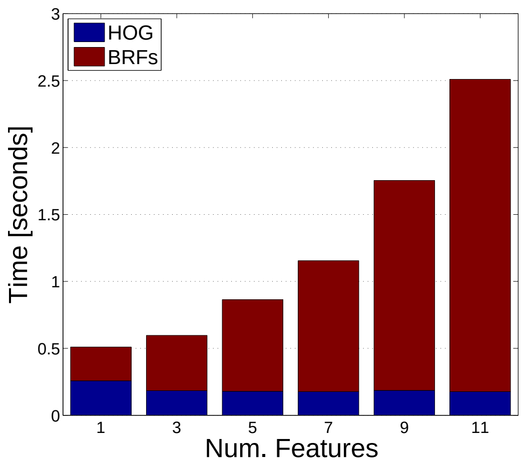
Then, we can say that the cost of evaluating the BRFs over an input image I of size $N_u \times N_v$ is of the order

$$O(N_u\, N_v\, S\, T\, M), \tag{4.14}$$

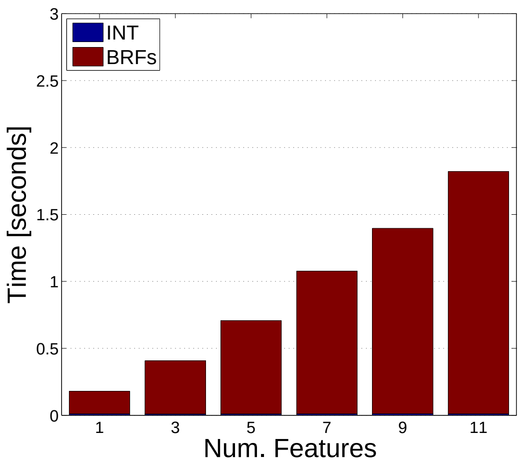
where S is the number of image scales, and T and M are the number of weak classifiers and HOG-based features per Fern, respectively.

In order to measure the efficiency of BRFs according to the above mentioned parameters, three different experiments were carried out. The first one consists on learning and evaluating the BRFs over the UIUC car dataset using several numbers of HOG-based features per Fern. The second experiment proceeds in the same way but computing the RFs over the image intensity space. The aim of this test is to measure the increment in time of using HOG-based features instead of simple intensity-based features. Finally, the last experiment concerns the evaluation of BRFs using varying numbers of weak classifiers.

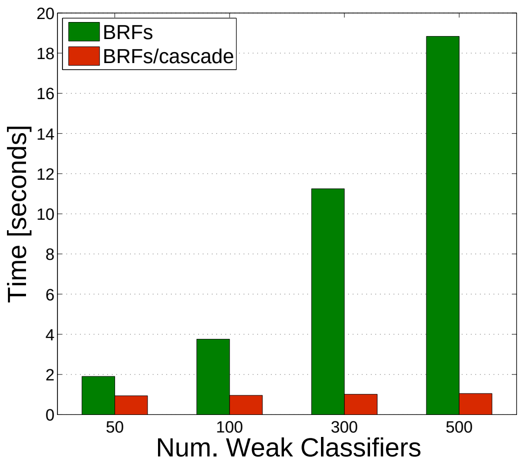
To speed up the detection phase a cascade strategy is added. In the past, many methods have used cascade strategies to increase the efficiency of the detection algorithms. The idea is to test a large number of features in an iterative procedure. The complete feature set is tested by a series of subset evaluations –cascades–, where each one has more complex and computationally expensive features. In contrast to more complex strategies [38, 98, 104], we opt for a naïve cascade because of its easy implementation and to focus on evaluating the BRFs and not the detection schemes. This simple cascade allows to reduce drastically feature computation time by controlling the



(a)



(b)



(c)

Figure 4.15: BRFs efficiency. (a) Detection times of BRFs using different number of HOG features per Fern. (b) Detection times of BRFs using intensity features. (c) The BRFs are evaluated according to the number of weak classifiers.

classifier output in its iterative evaluation. This can be formulated by,

$$H(x) = sign\left(\sum_{t=1}^{T} h_t(x) > \rho\right), \tag{4.15}$$

where ρ is the cascade threshold. If the BRFs output at iteration t is over this threshold, the testing goes on, otherwise, it is stopped and the current image region is discarded. This simple methodology allows to concentrate on promising image regions.

The detection times, given in seconds, are shown in Fig. 4.15. Results of the first experiment are shown in Fig. 4.15(a). Increasing the number of features has a quadratic impact on overall computation time, even when the computation of HOGs is constant. The same situation is observed in Fig. 4.15(b). The cost of image computation is constant and independent of BRFs. Working on the intensity space offers a reduction in detection time since the computation of HOGs is removed, but this at the expense of lower detection rates. This fact was evidenced in Sec. 4.3.2.

The number of weak classifiers also affects the efficiency of BRFs. Two testing approaches are considered. One uses the BRFs in combination with a naïve cascade, and another without this strategy. The number of HOG-based features for this experiment has been set to seven. Fig. 4.15(c) shows that the naïve strategy reduces considerably detection time because less weak classifiers are tested per pose –location and scale–. In contrast, using BRFs without this simple cascade requires to evaluate the entire and large set of weak classifiers for every image location and scale, with a time complexity linear with respect to the number of classifiers used.

The choice of the number of classifiers and the type of features to use is a trade-off between detection performance and computational cost. To conclude, the previous experiments were computed using the average detection times over 100 test images –UIUC car dataset–. The algorithms are implemented in Matlab with some functions in C++. However, the method is not completely optimized.

4.3.10 Comparison with the State of The Art

In this section, BRFs are placed in context with other successful detection methods comparing their detection rates over the addressed datasets. Table 4.3 summarizes these average detection rates. BRFs achieve remarkable results in comparison to the state-of-the-art methods with the advantage of being computationally more efficient, because they do not require the computation of complex features or the combination of multiple cues [44].

Noteworthy, for the UIUC car dataset, the proposed method achieved a detection rate of 98.2% EER –Equal Error Rate–, with a margin of 1%, for the multi-scale test. We can also see that computing BRFs via a boosting algorithm –RAB– they outperform Classic Random Ferns where features are chosen randomly –RND–. Moreover, computing RFs over local HOGs reports better rates than computing features over the image intensity domain –INT–. For the single scale test, the best detection rate

Method	UIUC Multi-scale	UIUC Single scale	TUD motorbikes
Agarwal et al. [1]	39.6%	76.5%	-
Fergus et al. [17]	-	88.5%	-
Fritz et al.[25]	87.8%	88.6%	-
Mutch et al. [60]	90.6%	99.9%	-
Shotton et al. [77]	-	92.8%	-
Mikolajczyk et al. [55]	94.7%	-	89.0%
Leibe et al. [43]	95.0%	97.5%	87.0%
Lampert et al. [38]	98.6%	98.5%	-
Leibe et al. [44]	-	-	92.8%
Gall et al. [26]	98.6%	98.5%	-
Maji et al. [51]	-	97.5%	-
BRFs (RND/HOG)	**81.2%**(±5.0)	**71.2%**(±9.4)	**72.3%**(±5.7)
BRFs (RAB/INT)	**95.9%**(±2.5)	**92.7%**(±1.7)	**74.8%**(±2.0)
BRFs (RAB/HOG)	**98.2%**(±1.0)	**96.2%**(±1.8)	**86.9%**(±2.6)

Table 4.3: Category detection rates for the addressed datasets.

achieved by the proposed classifier is 96.2% EER. Unlike the method presented by Shotton *et al.* in [77], BRFs do not need to test each image twice, and the detector is able to simultaneously detect cars facing to the left or right.

With regards to the TUD motorbike dataset, BRFs give a detection rate of 89.6% EER with a margin of 2.6%. This rate is competitive with other successful works but at a reduced computational cost, both in learning and detection. This dataset is very challenging given its very large intra-class variability, as well as the multiple occlusions present.

4.3.11 Detection Results

Exemplar detection results over the addressed datasets are shown in Fig. 4.16. BRFs are capable of detecting object categories in spite of difficult imaging conditions such as lighting changes, inter-class variability or mild occlusions.

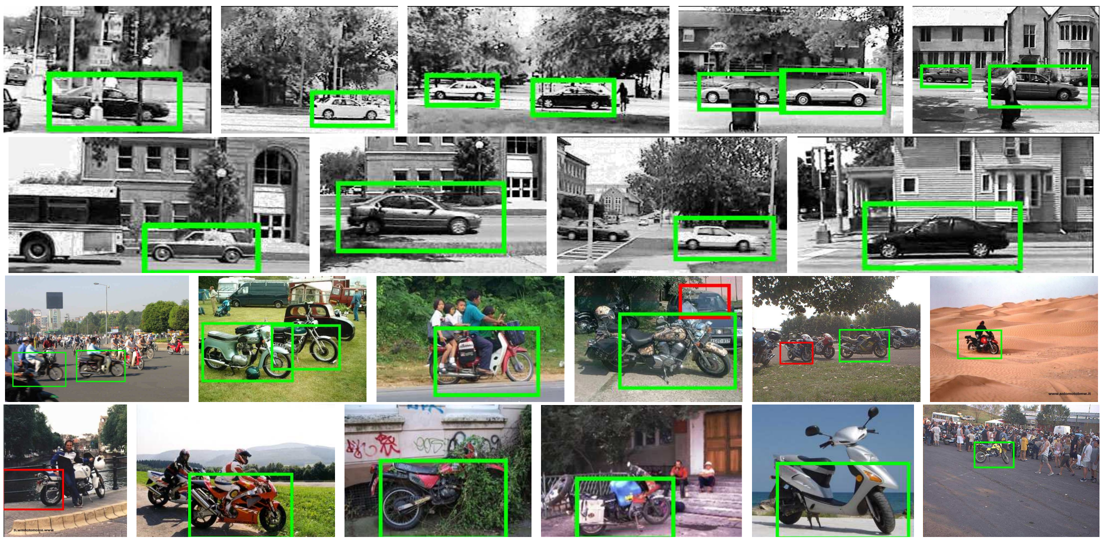


Figure 4.16: Detection results for the UIUC car dataset [1] and the TUD motorbike dataset [17]. The proposed method, based on BRFs, shows remarkable detection results in spite of difficult image conditions. Green rectangles indicate true positives –correct detections–, whereas red ones indicate false positives.

4.3.12 Bootstrapping BRFs

BRFs are evaluated on the INRIA horse dataset [18]. This dataset presents a certain degree of difficulty because of the reduced training set size. More specifically, only 50 positive images and 50 negative images are available. This dataset has a collection of images containing horses at multiple scales, deformations and out-of-plane rotations.

Given the small number of images for training in this dataset and since BRFs depend on the quality and quantity of the training data, shown experimentally in Sec. 4.3.6, we propose to learn the classifier by means of an iterative bootstrapping process. The idea is to extract more images, from the set, both positive and negative, in order to enlarge the training data with more difficult samples. In this way, the BRFs are retrained after every iteration to yield a more robust and discriminative object classifier [94]. The procedure for bootstrapping was explained in Sec. 4.2.5.

Basically, bootstrapping is carried out by testing the current BRFs over the training data. The resulting detections are then validated according to ground truth. Those detections close to ground truth are assigned as new positive samples. Otherwise, they

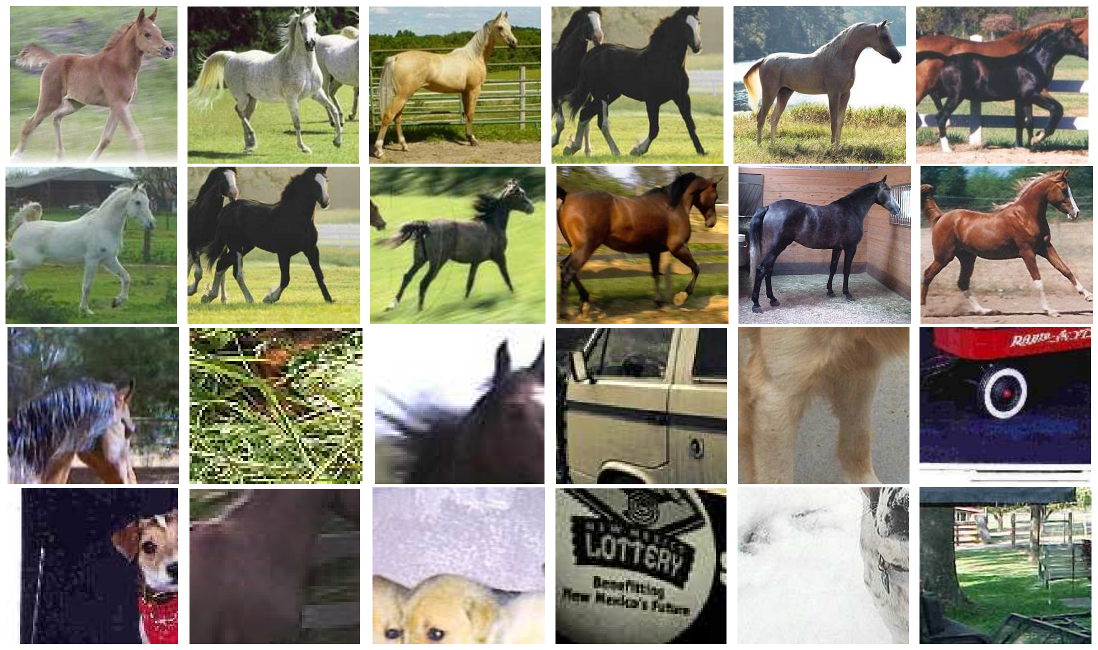
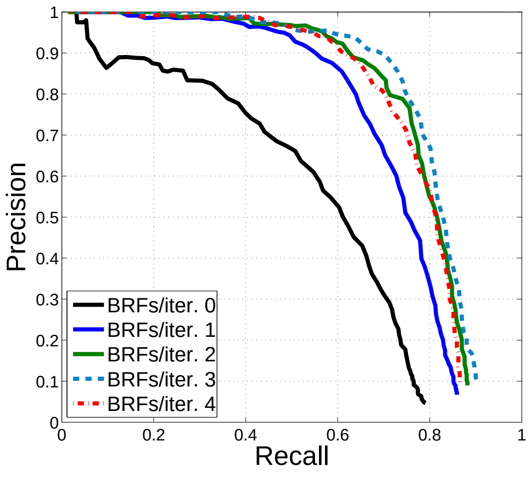


Figure 4.17: Bootstrapped images. Positive image samples correspond to the target object under scale and shift distortions –first and second rows–. By contrast, negative samples refer to difficult background regions or small portions of the object –third and fourth rows–.

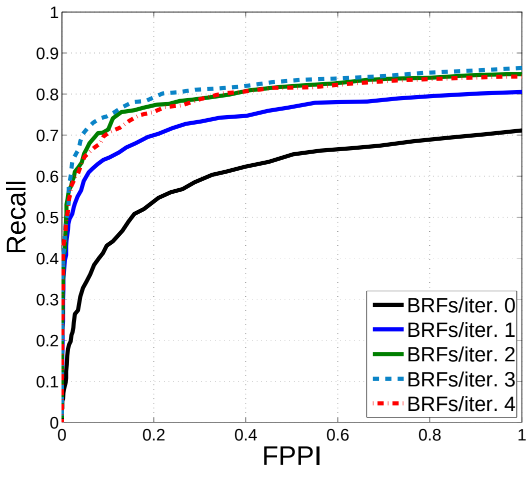
are assigned to the set of negative images. New positive samples correspond to object instances under diverse scales and shifts. They help BRFs to be more robust to object variations inside images. Contrary, bootstrapped negative images represent difficult images that are false positives emitted by the current classifier. These images force the algorithm to select more discriminative features against the background. Some bootstrapped samples after one iteration are shown in Fig. 4.17.

Bootstrapping produces significantly better recognition rates on this dataset. As shown in Fig. 4.18, our BRF classifiers are evaluated for several bootstrapping iterations, with the conclusion that it takes three iterations to optimize the results of bootstrapping. Note how for more iterations, overfitting starts to kick in and the performance of the classifier slightly degrades. Moreover, the method competes favorably with the state of the art as shown in Table 4.4.

With regard to the state of the art, our method outperforms some related works with a detection rate of 86.0% ±2.8 at 1 FPPI. This detection result and the num-



(a)



(b)

Figure 4.18: The detection performance of Boosted Random Ferns on the horses dataset improves substantially as the number of bootstrapping iterations increases. (a) Precision-recall curve. (b) Receiver-operator curve.

Method	**Num. Samples** pos. / neg.	**Num. Boots. Samples** pos. / neg.	**PR EER**	**ROC 1.0 FPPI**
Ferrari et al. [19]	50/50	–	–	73.7%
Ferrari et al. [18]	50/50	–	–	80.8%
Riemenschneider et al. [70]	50/0	–	–	83.7%
Maji et al. [51]	50/50	–	–	86.0%
Yarlagadda et al. [101]	50/0	–	–	87.3%
Toshev et al. [88]	50/50	–	–	92.4%
Monroy et al. [59]	50/50	–	–	94.5%
BRFs/iter. 0	50/50	–	$56.7\% \pm 4.6$	$71.2\% \pm 4.2$
BRFs/iter. 1	50/50	295/576	$68.4\% \pm 3.5$	$80.0\% \pm 4.6$
BRFs/iter. 2	50/50	577/1044	$75.0\% \pm 3.2$	$85.1\% \pm 2.9$
BRFs/iter. 3	50/50	861/1350	$77.0\% \pm 3.0$	$86.0\% \pm 2.8$
BRFs/iter. 4	50/50	1120/1590	$75.6\% \pm 1.3$	$84.4\% \pm 1.6$

Table 4.4: Quantitative detection results and comparison with the state of the art. For every bootstrapping iteration the training data is extended with new positive and negative samples. Indeed, they are the same original samples –50 samples– under image distortions: scaling, location shifts and affine transformations.

ber of training samples used at each bootstrap iteration are shown in Table 4.4. The proposed method is only superseded by Yarlagadda *et al.* [101], Toshev *et al.* [88] and Monroy *et al.* [59]. However, these works require larger computational effort. In Yarlagadda's work, for example, a novel Hough-based voting scheme is proposed for object

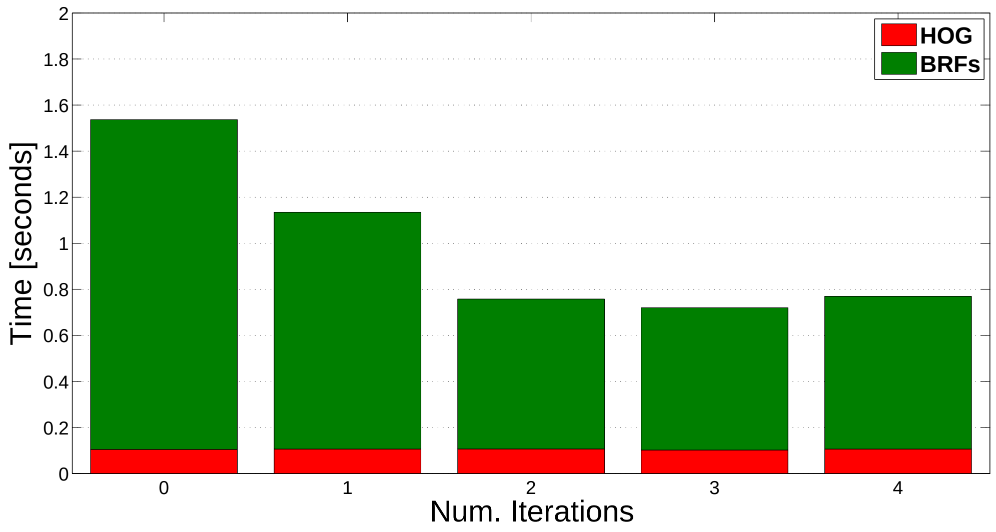
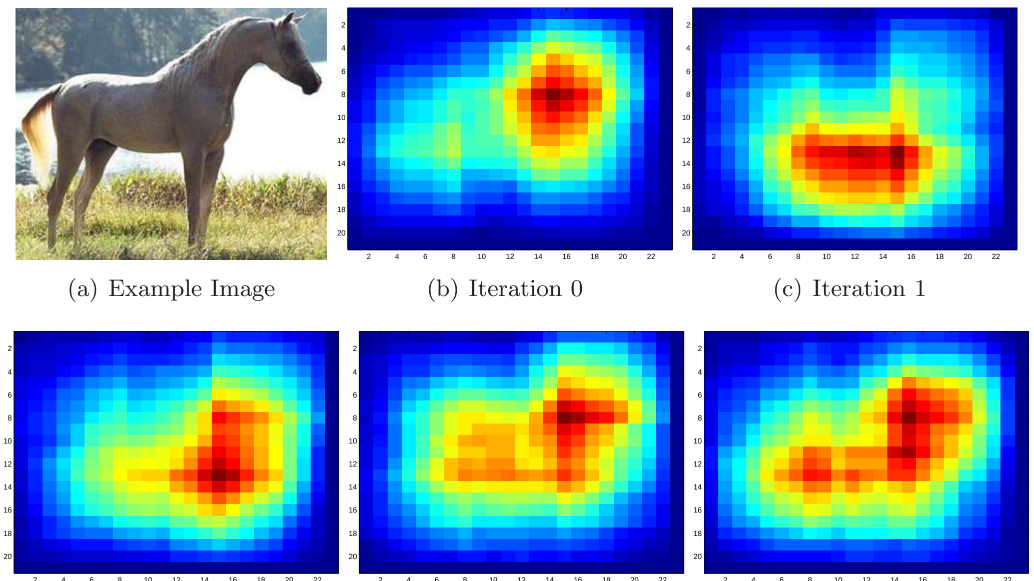


Figure 4.19: Detection times for BRFs according to the number of bootstrapping iterations. The bootstrapped classifier is not only more discriminative but also more efficient.

detection. This method requires an optimization procedure to group dependent parts and a verification stage to refine the voting hypotheses. In Toshev's work, a boundary structure segmentation model is proposed. Particularly, this method makes use of an initial segmentation based on superpixels and an optimization problem that is solved using semidefinite programming relaxation. In Monroy's work, the method integrates curvature information with HOG-based descriptors to produce better recognition and accuracy results. However, this method also requires additional processing steps, such as the computation of sophisticated edges, the extraction of segments based on connected elements and the computation of the distance accumulation. Besides, this work also performs bootstrapping. More precisely, three iterations are carried out to bootstrap more negative samples.

The proposed BRFs are very fast to compute, and take about 1 second to detect horses in one image. On the contrary, Yarlagadda's method takes about a couple of minutes per image on this dataset and Riemenschneider *et al.* takes about 5 seconds per image. This fact reflects the computational benefit of our method. This efficiency is achieved by using simple but discriminative Random Ferns in combination with a boosting phase [92]. Fig. 4.19 shows the detection times of our proposed method in terms of the bootstrap iterations. Specifically, the times for computing the HOG from



(a) Example Image (b) Iteration 0 (c) Iteration 1

(d) Iteration 2 (e) Iteration 3 (f) Iteration 4

Figure 4.20: Spatial feature distribution. (a) Object category. (b-f) Spatial layout of Random Ferns for diverse bootstrapping iterations. Note, RFs are mainly concentrated on the neck, head and legs of horses.

images, and testing the BRFs are shown. We also see that the bootstrapped classifier is more efficient because it is more selective and can discards background image regions more quickly.

Fig. 4.20 illustrates the benefits of the bootstrapping process for this horse database. The color coded images represent the distribution of the location of the local HOG Fern-based weak classifiers chosen by our method. Reddish tones indicate higher concentrations of classifiers and bluish colors correspond to lower concentrations. Note hat at early stages of bootstrapping, features concentrate on the single most discriminative part of the object around the neck. However, as the number of bootstrapping iterations increases, discriminative features move towards other parts of the horse such as the belly and the head.

Finally, some detections results for this dataset are shown in Fig. 4.21. In spite of challenging poses of the horses, the BRFs are capable of detecting the object category remarkably well.

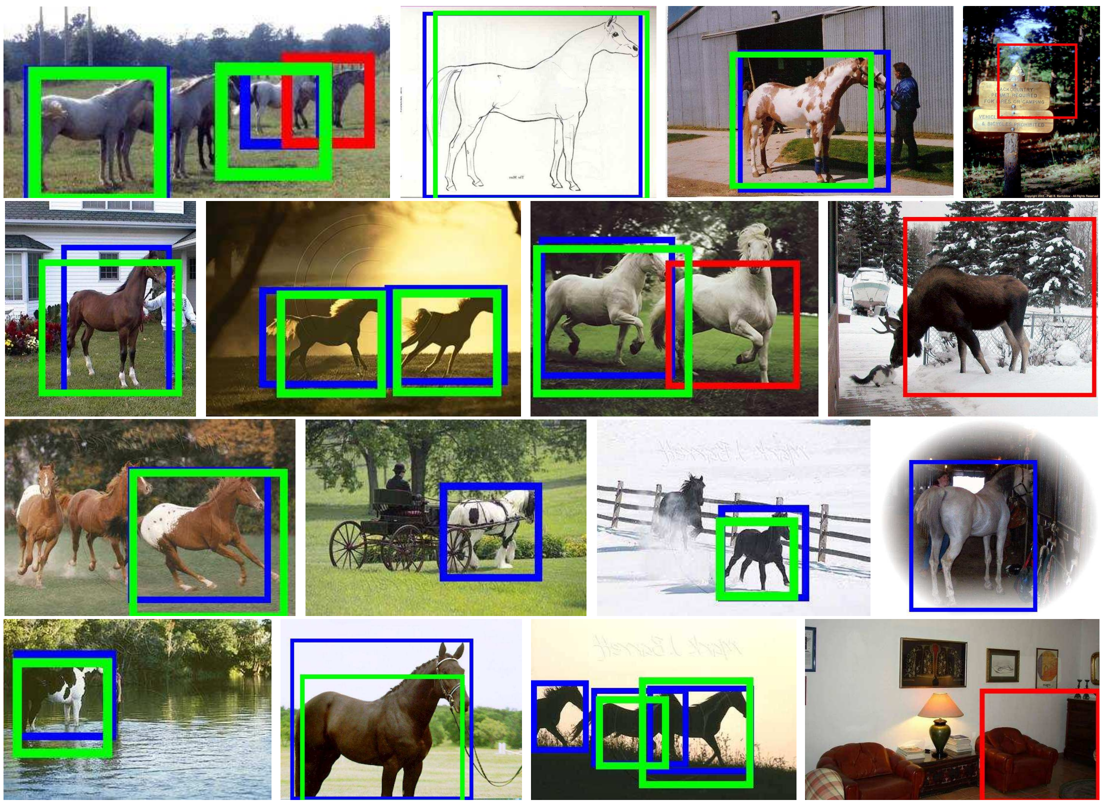


Figure 4.21: Detection results over the INRIA horse dataset. BRFs are able to detect the horse category in spite of difficult image and object variations. Blue rectangles indicate the ground truth given by the dataset, whereas green ones correspond to correct detections –true positives–. False positive are indicated by red rectangles.

4.4 Summary

The detection of specific objects categories is addressed in an efficient way by means of the computation of fast features –Random Ferns– in combination with a boosting algorithm with aim of extracting the most discriminative features for the object category. The resulting classifier, Boosted Random Ferns, is a robust and discriminative classifier that is capable of recognizing and localizing objects under difficult imaging conditions. BRFs have been validated in standard datasets where competitive results with the state of the art are achieved, with the advantage of being computationally more efficient and straightforward than other works.

Nevertheless, the performance of BRFs is conditioned to the quality and quantity of the training data. For cases where the datasets have a reduced set of training images, a bootstrapping procedure for collecting more difficult images is presented. This method allows to improve the detection rates of BRFs by retraining the classifier iteratively. The resulting BRFs are more robust to object variations and discriminative against background.

Chapter 5

Efficient Rotation-Invariant Object Detection Using Boosted Random Ferns

In this chapter, a novel and efficient method for detecting object categories that may appear in images under planar rotations is proposed. This detector is based on (*i*) the combination of Random Ferns via AdaBoost, Chapter 4, and (*ii*) a two-step detection strategy that prevents the classifier from being tested for multiple orientations. More specifically, we compute an orientation estimator and an object classifier using BRFs. The first step evaluates efficiently an estimator at each image location to determinate potential object hypotheses. The second step tests the object classifier only on the hypotheses given by the estimator. The resulting detection approach is very efficient and achieves promising detection rates over a dataset specifically created for the problem of object detection subject to in-plane rotations. This work was presented at [92].

5.1 Introduction

BRFs are used in this chapter to detect objects of a specific category that may appear in images subject to planar rotations. This problem has been traditionally addressed from a multi-class perspective using classifiers specifically trained at different orientations [34]. These methods however, suffer from two limitations. First, the computational cost for both the training and test stages increases with the number of classifiers, and second, the use of multiple classifiers increases the number of false positives.

A simpler strategy to deal with in-plane rotations would be to rotate the image or steer the object classifier for multiple orientations. However, this approach would have a high computational cost, because it would require to test the classifier several times over the image, one for each discretized orientation. In addition, it would produce a large number of false positives because the classifier would have to be evaluated significantly more times.

In [55] objects at different orientations are detected by means of an implicit shape model –ISM– using rotation-invariant features. Yet, the method is computationally expensive, as it requires to compute SIFT descriptors over edges, and apply a PCA analysis followed by a voting strategy. Other approaches address the problem as a multi-class one using different boosting versions [34]. However, since they decompose the problem into several classes they require more features and a higher computational effort. Torralba *et al.* [87] presented an efficient multi-class boosting algorithm where the overall number of features are reduced given that features may be shared among classes. The approach also requires an expensive learning step. Our method is closely related with Torralba's work in which we consider each object orientation a different class, and the estimator would be a top-level multi-class detection stage where Random Ferns are shared for all classes. The classifier would be a bottom-level stage containing selective RFs for the orientation-specific class.

Rotation-invariant detection is also addressed by Rowley *et al.* [71] for rotation-invariant face detection, and by Ozuysal *et al.* [65] for detecting cars under general 3D poses requiring several and relatively complex steps. In our approach, the computation of the estimator and the classifier is based on BRFs. The estimator is trained for all orientations with the aim of retrieving the object orientation using the joint probability between Fern observations and trained object orientations. The classifier is trained for a canonical orientation in order to construct a very discriminative classifier. This two-step approach speeds up the detection phase reducing significantly the search space.

An overview of the proposed detection method is given in Sec. 5.2. We then describe its components: the object estimator –Sec. 5.3– and the object classifier –Sec. 5.4–. Experimental evaluation of the two-step detection method is illustrated in Sec. 5.5. The chapter is concluded with a summary showing the achievements and results of the method –Sec. 5.6–.

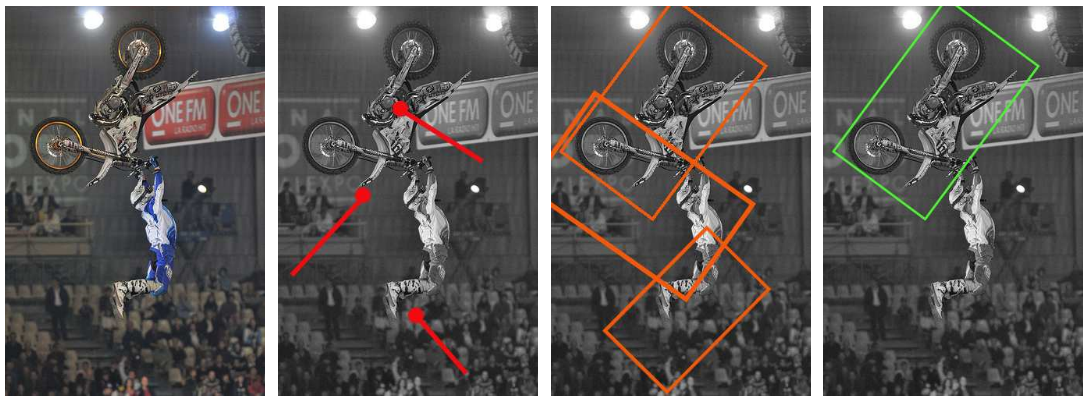


Figure 5.1: The proposed rotation-invariant object detection approach. First, the estimator yields an initial set of potential object poses –location, scale and orientation– over the input image (b). Then, each hypothesis is validated by steering and testing the object classifier (c). The hypotheses that remain after non-maxima suppression are considered object instances (d).

5.2 Two-Step Detection

Decoupling orientation estimation from object classification allows to reduce the computational time for the learning and testing phases, maintaining the detection results high. The estimation step is used as a pre-filter that generates object hypotheses. Given these hypotheses an orientation-specific classifier is properly steered and verified according to the estimated orientation, see Fig. 5.1. In this way, the proposed method avoids having a set of orientation-specific classifiers, or the computation of just one but steering it for multiple orientations in the detection phase. Both approaches are computationally expensive, both for testing and learning.

In contrast, the proposed approach only needs to test the estimator over the image to determine potential object candidates with an associated pose –location, scale and orientation–. This estimator can be performed efficiently thanks to the use of Random Ferns. Moreover, in a second step, the object classifier is very discriminative since it is trained using image samples at a canonical orientation. However, to avoid false positives, the object classifier is only tested on the hypotheses given by the estimator.

5.3 The Object Estimator

The object pose estimator $E(x)$ is computed in the same way as BRFs –Sec. 4.2–, that is, by a boosted combination of weak classifiers where each one is based on a RF that is computed at specific image location. The most discriminative sets of Ferns $\mathcal{F}$ and locations g are chosen via a RAB algorithm –Sec. 2.3– to construct, iteratively, the object estimator. The estimator can, therefore, be expressed as a sum of T weak classifiers,

$$E(x) = sign\left(\sum_{t=1}^{T} h_t(x) - \beta_e\right), \tag{5.1}$$

where h_t is a weak classifier and β_e is the estimator threshold whose default value is zero. In practice, when computing the pose estimator each weak classifier incorporates an additional orientation parameter Θ that is a label assigned to every training sample indicating the object orientation that has been applied by rotating training data to W in-plane rotations. Thereby, a weak classifier is calculated by the co-occurrence of Fern observation and image orientation,

$$h_t(x) = \frac{1}{2}\log\left(\frac{P(\mathcal{F}_t(x), \Theta(x)|C, g_t) + \epsilon}{P(\mathcal{F}_t(x), \Theta(x)|B, g_t) + \epsilon}\right), \tag{5.2}$$

where $\Theta(x) \in \{1, 2, .., W\}$, g_t indicates the location where $\mathcal{F}_t$ is computed, and ϵ is a smoothing factor. Refer to Table 5.1 for the nomenclature of this section.

The estimator seeks to maximize this co-occurrence during the learning step by evaluating different Random Ferns and keeping the most discriminative ones. This is done at each iteration t of the boosting step by calling a weak learner to compute and select the most discriminative classifier according to a weight distribution over the training samples D_t,

$$P(\mathcal{F}_t = z, \Theta = w|C, g_t) = \sum_{n:\mathcal{F}_t(x_n)=z \wedge \Theta(x_n)=w \wedge y_n=+1} D_t(x_n), \tag{5.3}$$

$$P(\mathcal{F}_t = z, \Theta = w|B, g_t) = \sum_{n:\mathcal{F}_t(x_n)=z \wedge \Theta(x_n)=w \wedge y_n=-1} D_t(x_n), \tag{5.4}$$

with $w = 1, 2, .., W,\ z = 1, 2, .., Z,\ n = 1, 2, .., N$. The number of training samples is denoted by N.

$\digamma$	Random Fern
$E(x)$	Estimator
$H(x)$	Object classifier
β_e	Estimator threshold
β_c	Object classifier threshold
h	Weak classifier
T	Number of weak classifiers
x	Image sample
z	Fern observation (output) value
Z	Dimension of Fern output space
Θ	Sample orientation label
Q	Bhattacharyya coefficient
g_t	Location of Fern t
N	Number of training samples
y_n	Class label of image sample n
D	Distribution of sample weights

Table 5.1: Notation for computing the object estimator and classifier.

At each iteration, the weak classifier that maximizes the classification power in terms of the following Bhattacharyya coefficient is selected,

$$Q = 2\sum_{w=1}^{W}\sum_{z=1}^{Z}\sqrt{P(\digamma_t = z, \Theta = w|C, g_t)P(\digamma_t = z, \Theta = w|B, g_t)}. \tag{5.5}$$

The weak classifiers built using this methodology are focused on Random Ferns that are both discriminative for their observations and for their orientation distributions. Thus, if one weak classifier tends to favor some orientations, subsequent classifiers are forced to classify those samples labeled as misclassified orientations. Details of this methodology for computing the estimator are given in the pseudocode of Alg. 5.

In order to estimate the object orientation at runtime using the previously computed estimator, this is evaluated according to the following conditioned expression:

$$E(x) = \frac{T}{2}\sum_{t=1}^{T}\log\frac{P(\digamma_t(x)|C, g_t)}{P(\digamma_t(x)|B, g_t)} + \frac{T}{2}\sum_{t=1}^{T}\log\frac{P(\Theta(x)|C, g_t, \digamma_t(x))}{P(\Theta(x)|B, g_t, \digamma_t(x))}. \tag{5.6}$$

The left-hand side of this equation is the root classifier Φ that corresponds to the ratio of observation probability of the T selected RFs. Note that it does not consider the orientation parameter Θ, and hence, this classifier responds to object instances under multiple in-plane rotations. By setting a threshold $\Phi > \beta_e$, we can choose a

Algorithm 5 Object Estimator

```
 1: Given a number of weak classifiers $T$ and a dataset consisting of $N$ image samples
    labeled as $(x_1, y_1, \Theta_1)..(x_n, y_n, \Theta_n)..(x_N, y_N, \Theta_N)$, where $y_n \in \{+1, -1\}$ is the label
    for the object and background classes, respectively; and $\Theta_n \in \{1, 2, .., W\}$ is the
    orientation label.
 2: Construct a pool of $R$ Random Ferns densely computed over the whole image.
 3: Initialize sample weights $D_1(x_i) = \frac{1}{N}$, where $i = 1, 2, .., N$.
 4: for $t = 1$ to $T$ do
 5:   for $r = 1$ to $R$ do
 6:     For the current distribution $D_t$ and Fern $F_r$, with location $g_r$, compute the
        weak classifier $h_r(x)$,
        $h_r(x) = \frac{1}{2} \log \left( \frac{P(F_r(x), \Theta(x)|C, g_r) + \epsilon}{P(F_r(x), \Theta(x)|B, g_r) + \epsilon} \right)$
 7:     Calculate the Bhattacharyya coefficient $QS$.
 8:   end for
 9:   Select the weak classifier $h_r$ that minimizes $Q$.
10:   Update the sample weights.
      $D_{t+1}(x_n) = \frac{D_t(x_n) \exp[-y_n h_t(x_n)]}{\sum_{n=1}^{N} D_t(x_n) \exp[-y_n h_t(x_n)]} \quad n = 1, 2, .., N.$
11:   Aggregate the computed weak classifier $h_t$ to the estimator $E(x)$.
      $E(x) \leftarrow h_t$
12: end for
13: Final strong classifier.
    $E(x) = sign \left( \sum_{t=1}^{T} h_t(x) - \beta_e \right)$
```

large number of potential hypotheses at runtime. The right-hand side of the Eq. 5.6 is the orientation estimation term which is made by the combination of local orientation estimations given the observations of BRFs. According to this distribution, the object orientation for the image sample x is calculated as:

$$\phi = \arg\max_{w} \sum_{t=1}^{T} \log \frac{P(\Theta(x) = w|C, g_t, F_t(x))}{P(\Theta(x) = w|B, g_t, F_t(x))}. \tag{5.7}$$

The method is exemplified in Fig. 5.2. The upper row shows the initial object hypotheses for varying values of the root detector threshold β_e. The object pose – location, scale and orientation– is represented by means of red lines. The images in the second row show the object detection results after testing the steered object classifier over those initial hypotheses. The parameter β_e controls the number of false positives of the estimator and, consequently, the computational cost of the algorithm. Therefore, the choice of this parameter is a trade-off between false positives and computational burden.

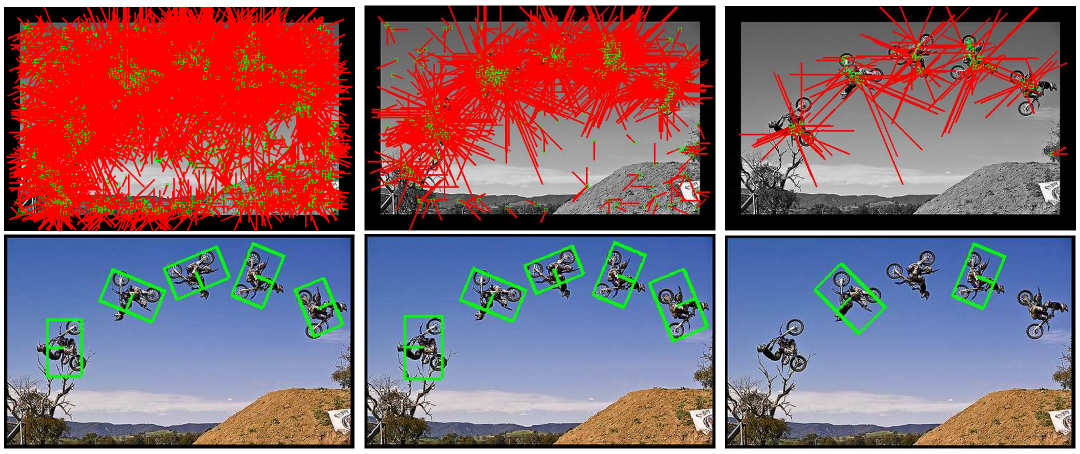


Figure 5.2: Object estimation and classification. First row: object hypotheses given by the object estimator $E(x)$. Red lines indicate location, size and orientation of potential objects. Second row: classification results after testing the object classifier over initial hypotheses. In each case, the classifier is steered and evaluated according to estimated poses. Each column corresponds to a different value of the parameter $\beta_e = \{0, 2, 4\}$.

5.4 The Object Classifier

This section provides the computation of the object classifier $H(x)$, via Boosted Random Ferns, and the procedure for steering this object classifier according to hypotheses given by the estimator $E(x)$.

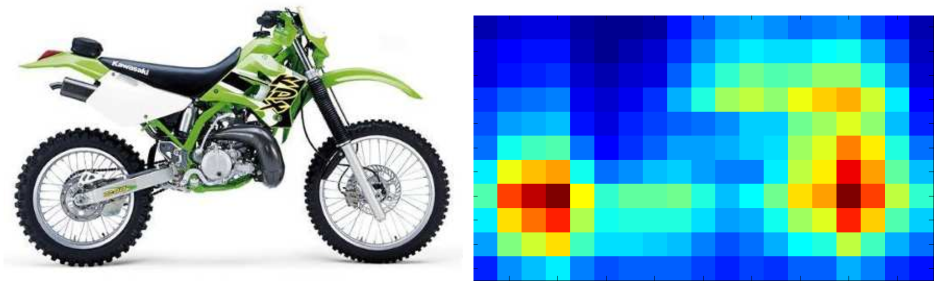
5.4.1 Computation of the Object Classifier

Unlike the estimator $E(x)$, that is computed using images for multiple object orientations, the object classifier –BRFs– is trained using image samples at a canonical orientation –i.e non-rotated images–. This yields a classifier that is very discriminative but not rotationally invariant. This is similar to the detection approach addressed in the previous chapter: BRFs for the detection of view-specific objects. To achieve rotation invariance, the classifier is steered at every orientation prediction given by the estimator. This prevents from having to train a set of orientation-specific classifiers.

Hence, the object classifier $H(x)$ is given by

$$H(x) = sign\left(\sum_{t=1}^{T} h_t(x) - \beta_c\right), \tag{5.8}$$

where β_c is the classifier threshold and h_t is a weak classifier.



(a) (b)

Figure 5.3: Spatial feature layout. (a) Motorbike category. (b) Spatial locations of computed Random Ferns for the given object category. Locations correspond mainly to prominent category parts –i.e wheels and handlebars–. Reddish tones indicate higher concentrations of features and bluish colors correspond to lower concentrations.

The procedure to compute BRFs is carried out as explained in the previous chapter. That is, by computing weak classifiers using AdaBoost, and selecting, according to Bhattacharyya coefficient, the most discriminative Ferns. Following this method, each weak classifier h_t is expressed as:

$$h_t(x) = \frac{1}{2} \log \left(\frac{P(\mathcal{F}_t|C, g_t) + \epsilon}{P(\mathcal{F}_t|B, g_t) + \epsilon} \right), \tag{5.9}$$

where ϵ is a smoothing parameter whose value is very small. At iteration t, the probabilities $P(\mathcal{F}_t|C, g_t)$ and $P(\mathcal{F}_t|B, g_t)$ are computed under the distribution of sample weights D_t by

$$P(\mathcal{F}_t = z|C, g_t) = \sum_{n:\mathcal{F}_t(x_n)=z \wedge y(x_n)=+1} D_t(x_n), \tag{5.10}$$

$$P(\mathcal{F}_t = z|B, g_t) = \sum_{n:\mathcal{F}_t(x_n)=z \wedge y(x_n)=-1} D_t(x_n), \tag{5.11}$$

where $z = 1, 2, .., Z$, $n = 1, 2, .., N$ and N is the number of training samples.

The weak classifier is chosen to minimize the following Bhattacharyya coefficient,

$$Q = 2 \sum_{z=1}^{Z} \sqrt{P(\mathcal{F}_t = z|C, g_t) P(\mathcal{F}_t = z|B, g_t)}. \tag{5.12}$$

The Random Fern that minimizes this coefficient is selected and added to construct the object classifier. The combination of all weak classifiers gives the strong and robust classifier classifier $H(x)$. Fig. 5.3 illustrates that the boosting step extracts discriminative Random Ferns for a given class –i.e. motorbikes–. For that case, features are concentrated mainly in semantic object parts such as wheels and handlebars.

5.4.2 Steering the Classifier

In order to be invariant to object rotations in the image plane, the object classifier must be steered and tested. To this end, for each object hypothesis made by the estimator $E(x)$, the classifier $H(x)$ is steered by simply rotating the coordinates of each HOG-based feature. This is done by,

$$\Omega^* = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \Omega + \begin{bmatrix} 0 \\ 0 \\ p \end{bmatrix}, \tag{5.13}$$

where Ω refers to feature location expressed by their spatial and orientation bin coordinates $[u, v, \theta]'$ in the HOG, ϕ is the rotation angle given by the estimator, and p is the angular translation increment defined by $\frac{\phi W}{\pi K}$. The parameter K is the number of gradient orientations used in the construction of HOG.

5.5 Experiments

The decoupled detection approach consisting of BRFs is validated for the recognition and localization of an object category under in-plane rotations. The evaluation is carried out in terms of the detection performance and the computational efficiency of the proposed two-step detection strategy. The estimator accuracy and its efficiency are also described in this section.

The parameters concerning to the computation of the detectors have been fixed for all experiments. The object classifier is trained using 300 weak classifiers –Random Ferns–, each one with 7 binary features. The estimator is trained using 100 RFs, each one with 7 binary features. With regards to the computation of HOGs, 8 gradient orientation channels have been chosen. Fern sizes are 6×6 and 4×4 HOG cells for the estimator and the classifier, respectively. To train the estimator, 16 object orientations are considered, resulting in accuracy of 22.5 degrees.

A new dataset is introduced to evaluate the decoupled detection approach against object rotations in the image plane. This dataset was created due to non-existence of datasets for the problem of detecting and categorizing objects under in-plane rotations. Most of datasets are focus on general 3D rotations.

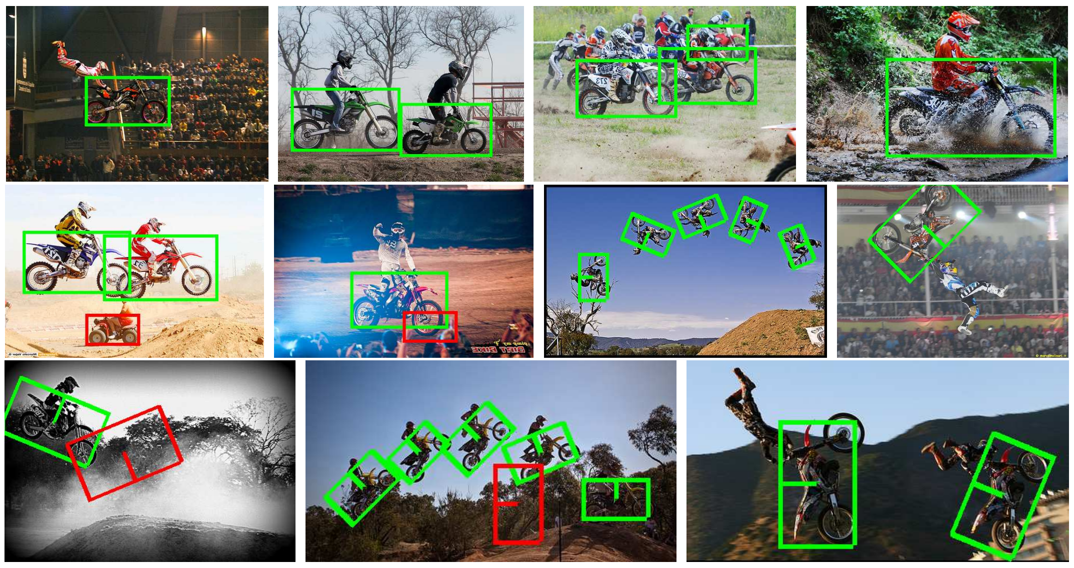
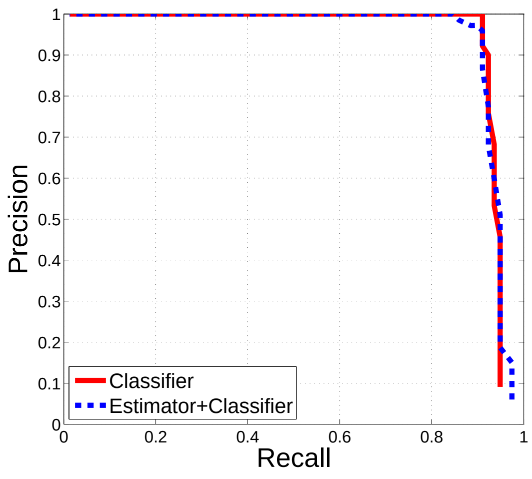
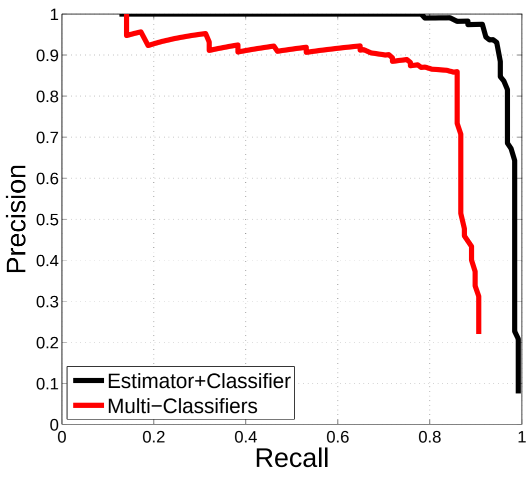


Figure 5.4: Detection results for the IRI freestyle motocross dataset. Green rectangles indicate true positives, while red rectangles indicate false positives. BRFs in combination with a two-step detection strategy lead to a robust and efficient method that is able to detect objects in spite of partial occlusions, planar rotations and scale changes.

IRI freestyle motocross dataset. This dataset was built in order to explicitly evaluate the proposed algorithm to rotations in the plane. The images were extracted from the Internet and correspond to motorbikes with challenging conditions such as extreme illumination, multiple scales or partial occlusions. Moreover, some instances show some degree of out-of-plane rotations, see Fig. 5.4. There are two sets of images for testing. The first set has 69 images with 78 motorbikes without in-plane rotations, whereas the second one has 100 images with 128 motorbikes instances with multiple rotations in the plane. The learning was done using images from the Caltech motorbike dataset [17]. However, the positive training set was created selecting the most closely related images to the target motorbike model. Following, affine transformations were added into images in order to extend the training set with new samples considering image distortions. This process was performed until 700 images were created.



(a)



(b)

Figure 5.5: Two-step detection strategy. (a) Detection curves for two different detection strategies. The evaluation is carried out, first, over samples without orientation change. Adding an estimator does not affect the final object detection since false hypotheses, given by the estimator, are rejected by the classification step. (b) Detection performance for the object category under in-plane rotations. The proposed approach reports better detection rates than testing the specific classifier at multiple orientations.

5.5.1 Two-step Detection

To validate the estimation-classification method, two types of experiments are performed. In the first experiment, two detection approaches are considered; one that only uses the object classifier and another where the classifier is tested in combination with the estimator. For this experiment, the first set of test images of the addressed dataset is used. This set refers to images containing motorbikes without in-plane rotations but with difficult imaging conditions such as occlusions or extreme illumination. Results for both detection strategies are shown in Fig. 5.5(a). According to the performance curves, both methods report similar detection rates –91.03% EER–. This experiment shows that adding the estimator to detect objects without rotations does not undermine the performance of the detector, because even when many object candidates are given by the estimator, the classification step is still able to reject false hypotheses.

For the second experiment, two detection approaches were considered again. The first one uses the estimator-classifier combination and the second one tests the classifier

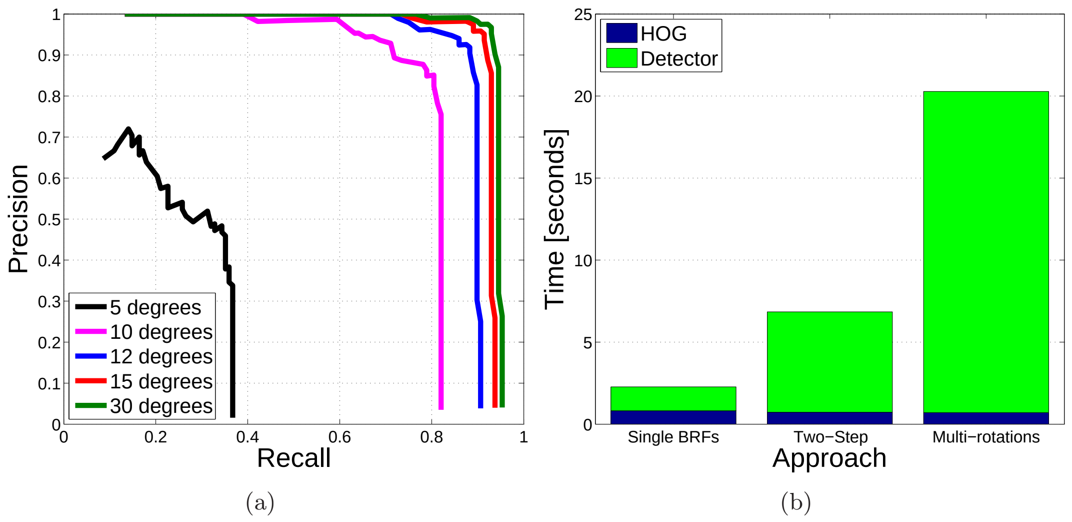


Figure 5.6: Estimator accuracy and efficiency. (a) Detection performance is measured in terms of estimator accuracy. Larger accuracies indicate high detection rates but with larger error in the retrieved object orientation. (b) Detection efficiency for the different detection strategies. Single BRFs achieve the lowest detection time, but are not invariant to object rotations. The proposed two-step detection approach achieves a reasonable detection time, whereas the method based on testing the BRFs at multiple rotations is considerably slow.

at multiple orientations. This experiment is carried out over the second set of test images that corresponds to images having motorbikes with orientation change in the plane. The detection rate for combining estimator and classifier is 93.75% EER while for the detection under multiple classifier rotations is 85.94%, see Fig. 5.5(b). This large difference shows that the combination of estimator and classifier yields better results than the classifier tested at multiple orientations since the latter has to be evaluated N times, being N the number of discretized orientations. Hypotheses verification at multiple orientations increases the number of false positives and computational cost.

5.5.2 Estimator Accuracy

Results from another batch of experiments designed to measure the orientation accuracy of the estimator are shown in Fig. 5.6(a). In these experiments, a true positive detection is only considered when the difference between the estimated orientation and the ground truth orientation is below a given accuracy value. The figure plots different detection curves for different accuracy values. The proposed method provides good detection results, above 91% EER, for an error margin of 15 degrees. The accuracy of the orientation estimation could be improved whether more object orientations are considered in the learning phase. However, this is at the expense of increasing the computational cost during the training phase.

5.5.3 Computational Efficiency

The computational cost of testing the proposed detection approach depends on the time spent on both steps, the orientation estimator and the object classifier. As both steps are computed using BRFs, their costs depend directly on the amount of weak classifiers and HOG-based features used. The cost of evaluating the estimator over an image of size $N_u \times N_v$ can be expressed as follows:

$$O(N_u \, N_v \, S \, T_e \, M_e), \tag{5.14}$$

where S is the number of scaled images –pyramidal representation–, and T_e and M_e are the number of weak classifiers and binary features for the estimator, respectively. Similarly, the cost of the object classifier has an upper-bound complexity of

$$O(\eta \, T_c \, M_c). \tag{5.15}$$

where T_c and M_c are the number of weak classifiers and Fern features for the classifier, and η denotes the image locations where the classifier is tested. Remember that the decoupled approach allows to initially evaluate the estimator and just consider the classifier when the estimator score exceeds a certain parameter β_e. Hence, the parameter η depends on the value of β_e, whose choice is a compromise between efficiency and detection performance. A small value means more classifier evaluations, while a large one gives less feature computation but more false negatives, that is, missing object instances. This was shown in Fig. 5.2. In the worst case, the detector cost will equal to the cost of applying both the estimator and the classifier sequentially. In fact, this cost is lower than testing multiple independent classifiers at different orientations, for which the cost would be:

$$O(N_u \, N_v \, S \, T \, M \, N_\theta), \tag{5.16}$$

with N_θ is the number of discretized orientations.

Fig. 5.6(b) shows the detection times for the different strategies utilized. This plot was computed for the average detection times over the test 1 of the motorbike dataset –69 images–. The strategies are: the classic BRFs described in the previous chapter, which is not rotationally invariant; the proposed decoupled detection approach, invariant to object orientation changes; and an approach that evaluates exhaustively the BRFs for several and discrete orientations.

We see that the most efficient is the single BRFs. The proposed method is an order of magnitude faster than evaluating exhaustively BRFs at multiple rotations. This fact demonstrates the computational benefit of the proposed work. The estimator alleviates the cost of evaluating this classifier many times over images and reduces the number of false positives.

5.5.4 Detection Results

Exemplar detection results for the IRI freestyle motocross dataset are shown in Fig. 5.4. Note how the two-step approach and BRFs are able to recognize and localize objects within difficult scenes. We can also see that object orientations are estimated correctly.

5.6 Summary

The detection of object categories under in-plane rotations has been addressed in an efficient way in this chapter. We propose the use of BRFs together a decoupled detection approach. This allows having fast feature computation and avoiding to test the object classifier at multiple orientations. The two-step detection approach, first evaluates the estimator in images to provide possible object candidates. Then, given these candidates, indicated by location, scale and orientation, the discriminative object classifier validates each hypothesis.

The method has been evaluated in a dataset specifically created for the presented problem. It has a set of images containing side-view motorbikes under rotations in the plane and challenging image conditions such as lighting changes or occlusions.

Chapter 6

Shared Random Ferns for Efficient Detection of Multiple Categories

Boosted Random Ferns –BRFs– are proposed in this chapter to detect multiple object categories, exploiting the fact that different categories may share common features but with different geometric distribution. This yields an efficient detector which, in contrast to existing approaches, considerably reduces the computation cost at runtime where the feature computation step is traditionally the most expensive process. More specifically, at the learning stage, shared features are computed by applying the same Random Ferns –RFs– over the histograms of oriented gradients –HOGs– on the training images. Following, a boosting step is applied to build discriminative weak classifiers, and learn the specific geometric distribution of the RFs for each class. At runtime, only a few Random Ferns have to be densely computed over each input image, and their geometric distribution allows performing the detection. The results from this chapter are the bulk of [93] and [91].

6.1 Introduction

In the last years the problem of detecting many different kinds of objects in cluttered scenes has received attention in the computer vision community. Nowadays, there are some works for this problem that have shown successful detection rates in spite of difficult scene and object conditions [13, 16, 30, 60, 88, 92]. Nevertheless, the computational efficiency of these methods becomes critical when several category classifiers are evaluated in images. The problem is that most methods scan exhaustively the image using a sliding window where the object classifier is tested for every image location and several scales. This strategy is computationally expensive and even more when multiple categories are considered. Some works overcome this problem using branch-and-bounds schemes that reduce significantly the amount of classifier evaluations and speed up object detection [38, 42]. However, these methods are not specially conceived for the multi-class object detection task.

The simplest approach to cope with the detection of multiple objects is to learn each category-specific classifier independently from the rest. While this procedure allows focusing on the most discriminative features for each category, it has a high computational cost at runtime, because the total number of features increases with the number of object categories. This is the case of BRFs which were proposed in Chapter 4. They constitute very efficient and robust classifiers, but each one has its own set of features –RFs–. This yields a detection strategy that involves the computation of a large number of features in images.

Recent approaches have attempted to reduce this computational burden inherent in the multi-class object detection problem by splitting the detection process in two steps: initially the object class is estimated by either using joint class classifiers [34, 87, 100] or a rough class estimator [15, 65, 92], and subsequently the object is accurately detected through category-specific classifiers. Nevertheless, in both situations the initial estimation is only reliable when object categories may be represented by predefined-regions with the same size or aspect ratio. To overcome this limitation of object size variability, other works have introduced hough-based methods, where object constituents cast probabilistic votes for the object center [6, 26, 51, 55, 69]. This avoids using detectors with rough and fixed sizes, but requires the computation of object-part codebooks or more sophisticated descriptors.

In order to address this situation, we proceed as the former methods mentioned above, independently learning a robust classifier for each category. However, to make their computation as efficient as possible, we propose to build a random pool of features

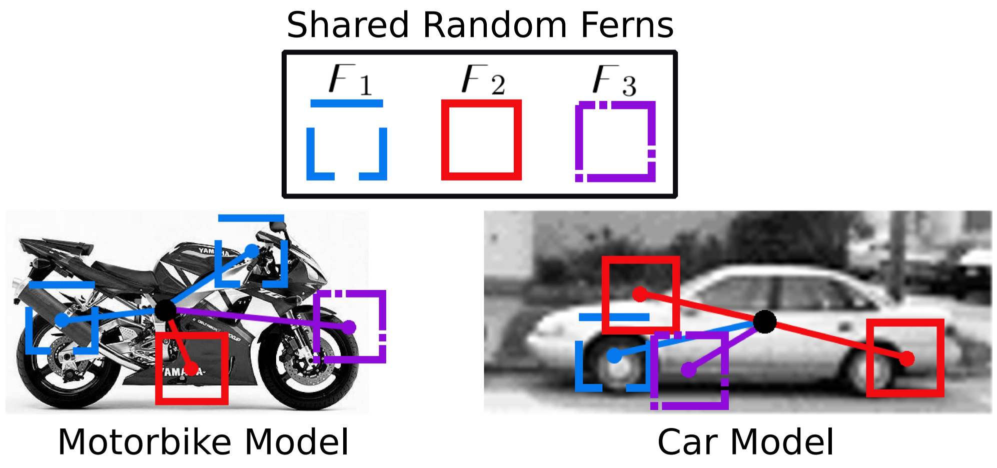
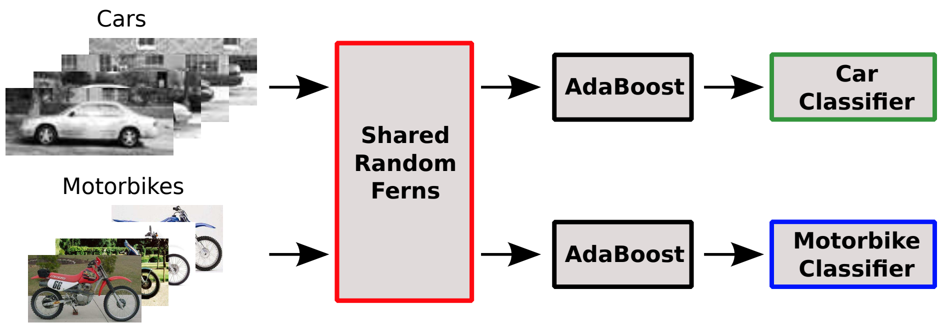


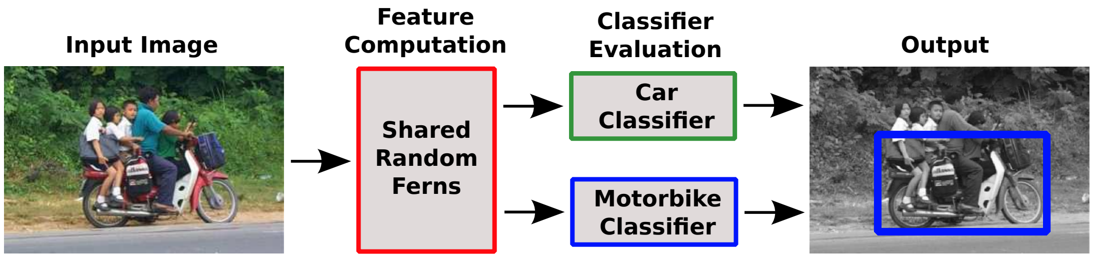
Figure 6.1: Shared Random Ferns. Each colored square F_i represents to a specific RF. The set of RFs are used for the construction of each category-specific classifiers.

that is shared by all the categories. For instance, in the example shown in Fig. 6.1, three features are shared by the “car” and “motorbike” classes. Then, at runtime, only these three features have to be evaluated over the input image, and the decision to classify such an image to belong to one class or the other, depends on the response of category classifiers built from the common features. Note that the process to calculate the features only needs to be done once, and is independent from the number of categories.

More specifically, the shared pool of features is built by applying the very same Random Ferns [64] on the HOGs of a set of training images from multiple classes. Given these features, we then use a boosting step to learn discriminative object classifiers. The result of the boosting step is a specific combination of Random Ferns for each category, that although sharing the same Ferns, a geometric distribution that is particular for each class. As will be shown in the Experiments section, our detector yields similar recognition and detection results as some state-of-the-art works when applied to each of the individual classes. With regards to BRFs –Chapter 4–, the proposed method increases even more their efficiency by sharing features among classes and by reusing features in the construction of a large number of weak classifiers. Moreover, a new and efficient methodology for testing BRFs is provided. It splits the feature computation from the evaluation of classifiers.



(a)



(b)

Figure 6.2: The proposed approach. (a) Training phase. Each object classifier is trained by separated but sharing the same features –Shared Random Ferns–. For the given example, car and motorbike detectors are computed via AdaBoost. (b) Testing phase. Object detection is carried out in two consecutive and efficient steps: feature computation, that is common for all categories, and classifier evaluation.

6.2 Overview of the Method

The proposed work copes with the efficient detection of multiple object categories by learning each category-specific classifier independently but sharing the same features –Shared Random Ferns–. This is exemplified in Fig. 6.2(a). In a first step, the set of features is computed at random over HOGs. This set represents the common features that are subsequently used by the boosting algorithm with the aim of building each object classifier. The algorithm constructs in each iteration, and for each category, a weak classifier to assemble it into the final object classifier. This weak classifier is computed by testing all RFs, inside the feature pool, over multiple object locations. The pair, Fern and location, that best distinguishes positive samples from negative ones is chosen as a weak classifier. This classifier computation is described through Sec. 6.4.

The result of this procedure is a set of very discriminative and robust object classifiers that shares the same Random Ferns. This fact leads to an efficient testing of diverse detectors given that feature computation is done once. Basically, the detection phase is performed via two consecutive steps, see Fig. 6.2(b). The first step is the most expensive process but it is common for all object detectors. It is feature computation that involves the convolution of the set of RFs over the input images. The second step consists on evaluating every object classifier given the Ferns outputs.

The remainder of this chapter is organized as follows. The set of random and shared features is described in Sec. 6.3, whereas the computation of specific classifiers using the shared features is explained in Sec. 6.4. The computational efficiency and the validation of the proposed method are given in Sec. 6.5 and Sec. 6.6, respectively. In Sec. 6.7, the chapter is concluded with a summary and conclusions.

6.3 Shared Random Ferns

The computation of shared features is performed by computing a reduced set of Random Ferns over HOGs. This set ϑ encodes different image appearances by means of the co-occurrence of several features outputs. Ferns are computed at random and using a fixed size –i.e 4×4 cells inside the HOG space– in order to guarantee generalization and a common size for all object categories. In this way, RFs are not specific for an object class but generic features for multiple categories. This fact motivates the learning of several classifiers using a compact and shared representation of features that alleviates the cost of testing multiple classifiers over images.

This feature set consisting of R Random Ferns can be written as:

$$\vartheta = \{\mathcal{F}_1, \mathcal{F}_2, .., \mathcal{F}_R\}, \tag{6.1}$$

where each Fern $\mathcal{F}_i$ is defined by a collection of M binary features, which are computed over the HOG. Similarly, a Random Fern can be expressed by

$$\mathcal{F}_i = \{f_1, f_2, .., f_M\}, \tag{6.2}$$

where f_i denotes a random binary feature. Refer to Table 6.1 for the nomenclature.

The impact of the amount of RFs over the detector performance is evaluated in Sec. 6.6 in terms of detection rates over a public dataset. The number of RFs determines not only detector performance but also its detection efficiency given that more common features involves more feature computation.

$\digamma$	Random Fern
f	Fern (binary) feature
x	Image sample
H^j	Boosted classifier for object class j
h^j	Weak classifier for class j
T	Number of weak classifiers
R	Number of Random Ferns
M	Number of features (f) per Fern
z	Fern observation (output) value
Z	Dimension of Fern output space
Q	Bhattacharyya coefficient
g_t	Location of Fern t
C_j	Object (positive) class j
B	Background (negative) class
β_j	Classifier threshold for classifier j
N	Number of training samples
D	Distribution of sample weights

Table 6.1: Notation for BRFs using common features.

6.4 Category-specific Classifier

The category-specific classifier is computed using BRFs. Basically, each category classifier is trained independently using its own set of training data, but sharing the same features –Random Ferns–. This shared set of features is initially constructed by computing a reduced number of Random Ferns. This set is used recursively for building discriminative weak classifiers in the boosting phase. Hence, the classifier is constructed as a combination of weak classifiers where each one is based on one RF, selected from the feature pool ϑ, with an associated spatial image location.

More formally, the object category classifier $H^j(x)$, for the category j, is computed to yield the sets of Ferns $\digamma$ and locations g that best discriminate the object category C^j from the background B. This is done by means of the Real AdaBoost algorithm described in Sec. 2.3. The idea behind the boosted learning step is to extract the most discriminative and reliable features and their spatial distributions for classifying the target class in an iterative and supervised process where each training sample has a weight corresponding to its classification difficulty. The algorithm updates these weights on each round. The weights of those samples incorrectly classified are increased in order to put more effort in their classification in the following iterations.

Algorithm 6 Object Classifier j.

```
1: Given a number of weak classifiers $T$, a shared feature pool $\vartheta$ consisting of $R$ Random Ferns, and $N$ image samples $(x_1, y_1)...(x_n, y_n)...(x_N, y_N)$, where $y_n \in \{+1, -1\}$ is the label for object $C_j$ and background classes $B$, respectively:
2: Initialize the sample weights $D_1(x_i) = \frac{1}{N}$, where $i = 1, 2, .., N$.
3: for $t = 1$ to $T$ do
4:   for $r = 1$ to $R$ do
5:     for $g \in \mathbb{R}^2$ do
6:       For the current distribution of weights $D_t$, Fern $\digamma_r$ and image location $g$, compute the weak classifier $h^j_{r,g}$.
         $h^j_{r,g}(x) = \frac{1}{2} \log \left( \frac{P(\digamma_r|C_j,g)+\epsilon}{P(\digamma_r|B,g)+\epsilon} \right)$
7:       Compute the Bhattacharyya coefficient $Q$
8:     end for
9:   end for
10:  Select the weak classifier $h^j$ that minimizes the Bhattacharyya coefficient.
11:  Update the samples weights.
     $D_{t+1}(x_n) = \frac{D_t(x_n) \exp[-y_n h_t(x_n)]}{\sum_{n=1}^{N} D_t(x_n) \exp[-y_n h_t(x_n)]} \quad n = 1, 2, .., N$
12:  Aggregate the computed weak classifier $h^j_t$ to the strong classifier $H^j \leftarrow h^j_t$.
13: end for
14: Final strong classifier.
    $H^j(x) = sign \left( \sum_{t=1}^{T} h^j_t(x) - \beta_j \right)$
```

Then, a category-specific classifier, consisting of T weak classifiers, is given by

$$H^j(x) = sign \left(\sum_{t=1}^{T} h^j_t(x) - \beta_j \right), \tag{6.3}$$

where β_j is the classifier threshold and h^j_t is a weak classifier defined by

$$h^j_t(x) = \frac{1}{2} \log \left(\frac{P(\digamma_t(x)|C_j, g_t) + \epsilon}{P(\digamma_t(x)|B, g_t) + \epsilon} \right), \tag{6.4}$$

being ϵ a smoothing factor. At iteration t, the probabilities $P(\digamma_t|C_j, g_t)$ and $P(\digamma_t|B, g_t)$ are computed using the distribution of sample weights D,

$$P(\digamma_t = z|C_j, g_t) = \sum_{n:\digamma_t(x_n)=z \wedge y_n=+1} D_t(x_n) \ , \quad n = 1, 2, .., N \tag{6.5}$$

$$P(\digamma_t = z|B, g_t) = \sum_{n:\digamma_t(x_n)=z \wedge y_n=-1} D_t(x_n) \ , \quad n = 1, 2, .., N \tag{6.6}$$

where N is the number of training samples, and $z = 1, 2, .., Z$ indicates the Fern output or observation. The value of Z is determined by 2^M.

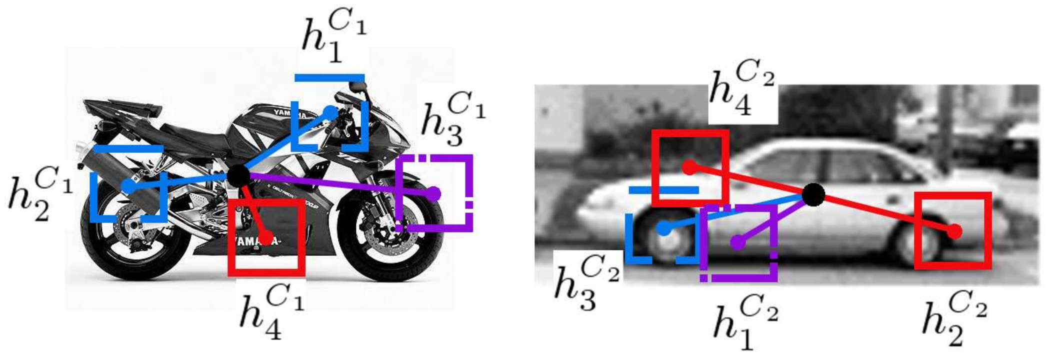


Figure 6.3: Category-specific classifiers. Each weak classifier $h_i^{C_j}$ consists of a specific Fern $\mathcal{F}_i$, chosen from the pool ϑ, and a distance g_i relative to object center. Each resulting classifier has a specific geometric and appearance distribution that makes each one very discriminative.

For each round t, the boosting algorithm seeks the most discriminative weak classifiers by evaluating each RF at every image location. The selection of the weak classifier, corresponding to one Fern $\mathcal{F}_i$ at specific location g_i, is carried out by the classification power of this weak classifier over the training samples. This is measured by means of the Bhattachryya coefficient between object and background distributions. Therefore, the classifier h_t^j that minimizes the following criterion is selected,

$$Q = 2 \sum_{z=1}^{Z} \sqrt{P(\mathcal{F}_t = z | C_j, g_t) P(\mathcal{F}_t = z | B, g_t)} \,. \tag{6.7}$$

Fig. 6.3 depicts some weak classifiers –colored squares– retrieved for the car and motorbike categories. Pseudocode to compute the object classifiers is shown in Alg. 6.

6.5 Efficiency of Sharing Features

The efficiency of the proposed method lies in the simplicity and efficiency of sharing Random Ferns in multiple classes. Sharing features allows to decouple the detection phase into two consecutive steps. While the first one performs feature computation, the second step evaluates every object classifier. Feature computation involves to test each RF, inside the feature set, over the input image. The cost of this step can be formulated as:

$$O(N_u \, N_v \, R \, M), \tag{6.8}$$

where $N_u \times N_v$ is the image size, R is the number of RFs, and M is the number of features per Fern. As we can observe, the computational cost of this step does not depend neither on the number of categories nor the amount of weak classifiers forming the object classifier. Furthermore, this process is carried out just once. This is the main advantage of the proposed method, the reduction of computational cost when multiple object categories are considered.

On the other hand, the cost of evaluating each object classifier can be expressed by

$$O(N_u\, N_v\, T\, K_c), \tag{6.9}$$

being T and K_c the number of weak classifiers and object categories, respectively. Note also that this step does not involve feature computation. It only evaluates the weak classifiers given the responses of Random Ferns –the first step–. The object classification is then computed from the combination of weak classifier outputs. This proposed detection procedure reduces the expensive cost of evaluating multiple and feature-independent classifiers, whose cost would be

$$O(N_u\, N_v\, R\, M\, T\, K_c). \tag{6.10}$$

This cost is significantly larger than our method, and augments drastically with the number of classes.

6.6 Experiments

In this section, we validate several aspects of the proposed method on public datasets, and compare its performance to some state-of-the-art works. The datasets we consider are the well-known UIUC car dataset [1], the TUD motorbike dataset [25] and the Caltech face dataset [17].

UIUC car dataset [1]. This dataset contains car-sides under difficult imaging conditions such as illumination changes, cluttered backgrounds and mild occlusions. This dataset has two sets of images for testing. The first one has 170 images containing 200 car instances. The second one has 108 images including 139 cars at different scales. This dataset has also a set of training images that is formed by 550 positive images and 500 negative ones. The second test set has been chosen for the experiments as it represents a more challenging object recognition problem since images contain cars at several scales.

TUD motorbike dataset [25]. This dataset consists of 115 images containing 125 motorbike instances under occlusions, several scales and difficult backgrounds. Furthermore, this dataset shows high intra-class variability because the quite different motorbike models. For training, the Caltech motorbike dataset is used [17]. It has 826 object images and a large number of background samples.

Caltech face dataset [17]. The dataset has 450 images containing frontal faces of a specific group of people with different lighting, face expressions and backgrounds. For training, the first 250 images were chosen. The rest are used for testing.

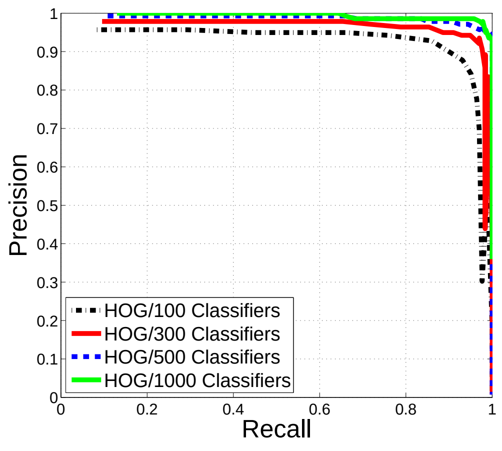
6.6.1 HOG-based Ferns

The proposed method has been tested using Ferns on the HOG and intensity domains, see Fig. 6.4(a,b). This experiment is performed over the multi-scale car dataset. The aim of this experiment is to show detector performance with respect to the feature space. Besides, the classifiers are trained using different weak classifiers sizes –100, 300, 500, and 1000–, but the same number of Random Ferns. That is, the same pool size. For this experiment, we consider 10 shared Random Ferns and 7 features per Fern. The plots show that using HOG-based features instead of intensity-based features results in better detection rates. In addition, we realize that increasing the number of weak classifiers the detection performance is also increased. Yet, since the number of features remains the same, the cost of the algorithm does not increase significantly.

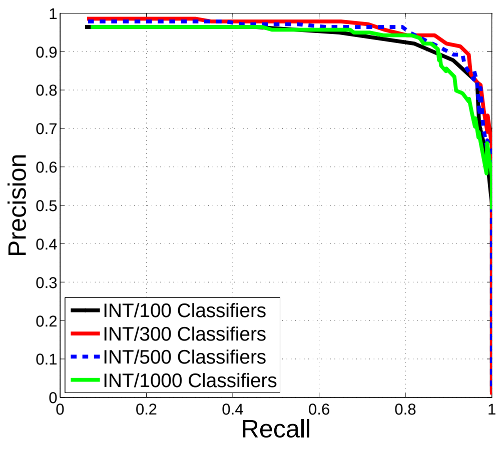
6.6.2 Feature Set Size

Here, classifier performance is evaluated according to the size of the feature pool. The choice of this parameter has an impact over classification performance. More Random Ferns implies increased ability to capture different image appearances but also more feature computation given that the set of Ferns must be evaluated inside images exhaustively.

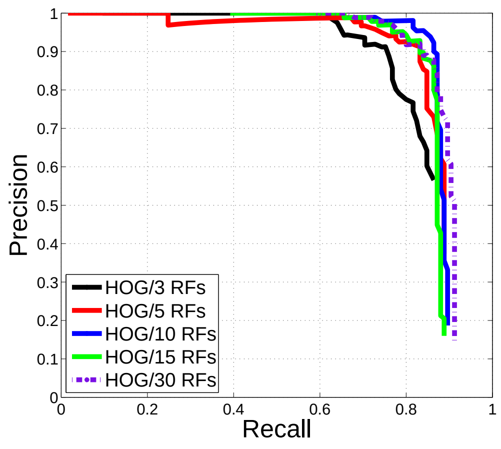
The present evaluation is carried out using the TUD motorbike dataset. For training, we have used 400 motorbike images from the Caltech motorbike dataset [17].



(a)



(b)



(c)

Figure 6.4: Detection performance of our method with different training parameters. (a) HOG-based features with varying sets of weak classifiers. (b) Features based on pixel intensities –INT–. (c) Performance in terms of the feature set size.

Fig. 6.4(c) shows the detection curves for varying number of Random Ferns. It also shows that with only 10 shared Random Ferns the classifier achieves remarkable results comparable to some state-of-the-art methods specifically tailored to single object detection, refer to Table 6.2. In this experiment, the classifier has been trained using the same number of weak classifiers and features per Fern, 300 and 7, respectively.

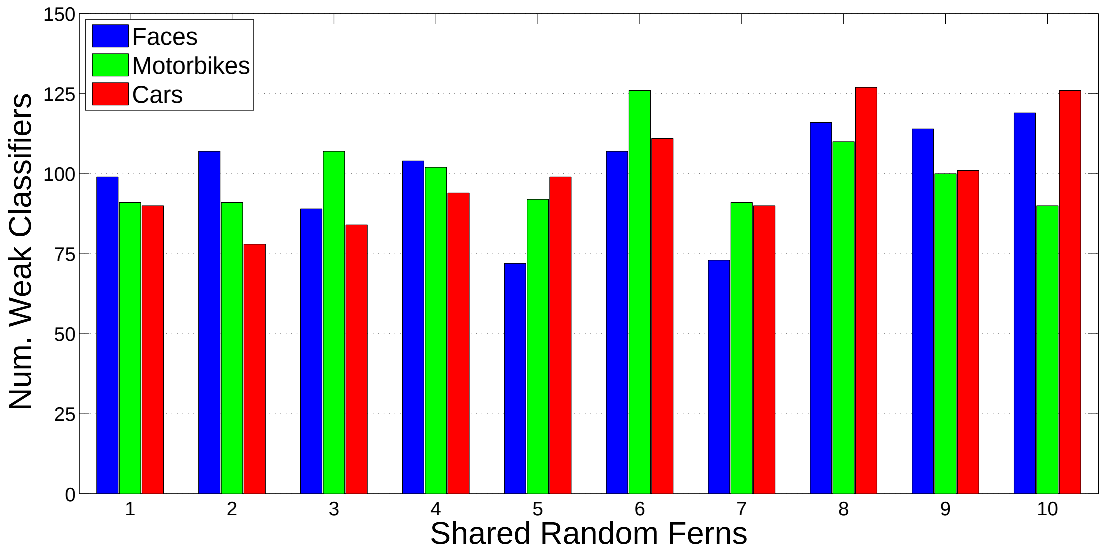


Figure 6.5: Sharing features. Distributions of shared Random Ferns for the construction of three different category-specific classifiers. Classifiers were trained using a fixed set of 10 Random Ferns and 1000 weak classifiers. Note that RFs are repeatably used for constructing the 1000 weak classifiers and in similar proportions for all object categories.

6.6.3 Sharing Features

To illustrate how features are shared by different object categories in order to compute their corresponding classifiers, Fig. 6.5 shows the distribution of Random Ferns for the given category-specific classifiers. Each classifier has been learned using the same parameters, 7 HOG-based features and 1000 weak classifiers. We see that RFs are repeatably used for constructing the 1000 weak classifiers and in similar proportions for all categories. This fact shows that RFs are generic and that can be utilized for the description and classification of multiple categories. Other important fact to stand out is that RFs are reused to compute different weak classifiers and thus to preserve the same computational cost. Consequently, feature computation has constant cost, independent of the number of object categories and weak classifiers.

As commented before, although the object-specific classifiers share the same set of features, they are highly discriminative since a particular geometric distribution of features is extracted for each class. This is done by AdaBoost that selects, for each class, robust and relevant features at specific image locations. This is exemplified in

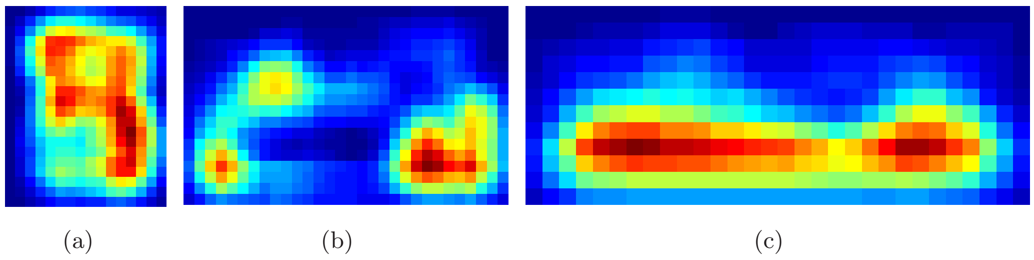


Figure 6.6: Spatial feature distribution. Spatial layout of Random Ferns for different object categories: faces (a), motorbikes (b) and cars (c). Reddish tones indicate higher concentrations of classifiers and bluish colors correspond to lower concentrations

Method	**UIUC Cars**	**Caltech Faces**	**TUD Motorbikes**
Agarwal et al. [1]	39.6%	-	-
Fergus et al. [17]	-	96.4%	-
Fritz et al. [25]	87.8%	-	81.0%
Mutch et al. [60]	90.6%	-	-
Shotton et al. [77]	-	94.0%	-
Mikolajczyk et al. [55]	94.7%	-	89.0%
Leibe et al. [43]	95.0%	-	87.0%
Lampert et al. [38]	98.6%	-	-
Leibe et al. [44]	-	-	92.8%
Gall et al. [26]	98.6%	-	-
Our Method	**97.8%**	**99.1%**	**86.7%**

Table 6.2: Object detection performance. Our best detection rates together with other rates provided by some state-of-the-art works. The evaluation indicates Equal Error Rates –EER–. The proposed method yields competitive results.

Fig. 6.6 where three feature distribution maps are shown, each one corresponding to a specific object category. Note, for instance, that for the car category features are mainly concentrated on wheels.

6.6.4 Detection Results

In order to establish a comparison with some recent and successful works, Table 6.2 summarizes our best detection rates and some rates reported by other works. The proposed method achieves competitive results with the benefit of its computational efficiency. In spite of the simplicity of the shared features used, the method yields high detection rates.

6.7 Summary

We have presented an algorithm for multiple object detection, that makes use of a common pool of features, computed using Random Ferns over the HOG domain. We have shown that sharing common features yields an efficient method for the recognition and localization of multiple object categories, with detection rates similar to current approaches that compute specific features for each category.

The proposed method computes very discriminative and robust classifiers by learning each category independently. Nevertheless, and in oder to reduce the inherent cost of testing multiple classifiers, we present a new methodology where Random Ferns are first computed and then used to build up specific classifiers. This leads to feature computation that is evaluated only once. The classifier reuses RFs for the construction of its weak classifiers. This reduces even more the detector cost.

With regards to BRFs, this chapter provides a new method for learning and testing the object classifier via BRFs. The new approach we present is much more efficient because a small and shared set of features is utilized to compute a large number of weak classifiers across classes. This is contrary to our previous BRFs –Chapter 4–, where each object classifier has its own feature set, and each weak classifier is based on unique and independent Random Fern.

Chapter 7

Efficient 3D Object Detection using Multiple Pose-Specific Classifiers

In this chapter, we combine the two-step detection strategy using BRFs from Chapter 4 with feature sharing from Chapter 6 to come up with an efficient method for object localization and 3D pose estimation. In the first step, a pose estimator is evaluated in the input images in order to estimate potential object locations and poses. These candidates are then validated, in the second step, by the corresponding pose-specific classifier. The result is a detection approach that avoids the inherent and expensive cost of testing the complete set of specific classifiers over the entire image. Further speedup is achieved by feature sharing. Features are computed only once and are then used for evaluating the pose estimator and all specific classifiers. The proposed method has been validated on two public datasets for the problem of detecting of cars under several views. The results show that the proposed approach yields high detection rates while keeping efficiency. The results from this chapter have been presented in [91].

7.1 Introduction

The problem of efficiently testing multiple specific classifiers has recently gained popularity for tackling the problem of detecting multiple object categories or specific objects seen from different viewpoints. In these problems, each object class, or object view, is commonly considered as a different topic represented by a distinct classifier. As a

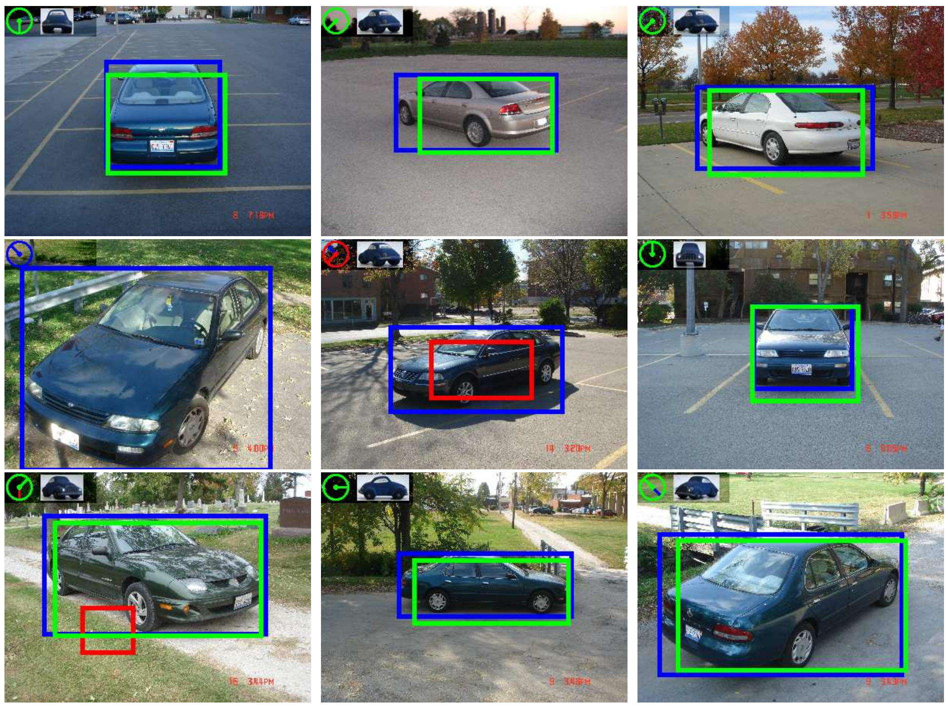


Figure 7.1: Efficient localization and pose estimation in the Savarese car database [72]. Our approach allows localizing cars and estimating their pose despite large inter-class variations and in about 1 second. Correct detections are depicted by green rectangles, whereas false positives are indicated by red ones. The ground truth is shown by a blue rectangle. The circle and car toy indicate the estimated viewpoint.

result, a large number of discriminative and specific classifiers are computed. Although these classifiers can be learned quite efficiently, testing each of them over an image is computationally expensive.

In this work we propose an *efficient strategy* for testing *multiple specific classifiers* for object detection. We study the problem more closely on the detection of cars from multiple views. This category includes challenges such as high inter-class variations, lighting changes, several car sizes and different aspect ratios of the bounding box. In order to address all these issues, we use a decoupled approach consisting of (i) a pose estimator and (ii) a set of pose-specific classifiers. The estimator acts as a filter and prevents having to evaluate all the specific classifiers at each position. Furthermore, we use feature sharing for the estimator and all classifiers. Both these characteristics yield remarkable computational efficiency with high detection rates. Fig. 7.1 depicts some detection results and the corresponding estimated poses.

Several strategies have been proposed in the past for this object detection and pose estimation problem. Some of them mainly rely on (*i*) features that can be computed very fast over images, and thus, increase the speed of the sliding window classifier that is evaluated at multiple scales and locations [64, 90, 98]. Other methods use specific cascaded classifier structures eg. [82, 98] which allow rejecting background windows at early stages and hence, also reduce the computational effort. In other words, these approaches aim for (*ii*) reducing the search space during the detection phase. This is also achieved by means of branch and bound techniques [38, 42], using object priors [2] or splitting the process in two consecutive phases of object estimation and specific detection [51, 65, 71, 92]. Finally, other works (*iii*) have proposed to share features across object classes or views [87, 93, 100].

We propose an efficient method that integrates synergically the strategies reviewed above. More specifically, our method computes Random Ferns (RFs) [64] over local histograms of oriented gradients. They are basically Boosted Random Ferns which were introduced in Chapter 4 for the recognition of specific object categories. In addition, we use shared Random Ferns –Chapter 6– that are used for the computation of the two-step detection approach, that is, the pose estimator and the set of pose-specific classifiers. Unlike other previous works that use Hough-based approaches as object classifiers [6, 26, 51], we use a novel Hough-RFs for building an efficient and robust 3D pose estimator. This estimator uses the Hough transform to learn and map the local appearances of objects –encoded by RFs– into probabilistic votes for the object center. This methodology overcomes previous works which compute rough estimators or predict the object size first [65, 92].

The resulting method is able to learn and detect objects in a straightforward and efficient manner. In particular, the estimator and specific classifiers can be learned in a couple of minutes, while the object detection is performed in about 1 second, using a non-optimized code based on Matlab. In addition, this efficiency is accompanied with high detection rate, comparable and even better than existing approaches.

7.2 Overview of the Method

The main ingredients of our approach are (i) a shared feature representation and (ii) an object pose estimator that limits the search space for (iii) object pose specific classifiers.

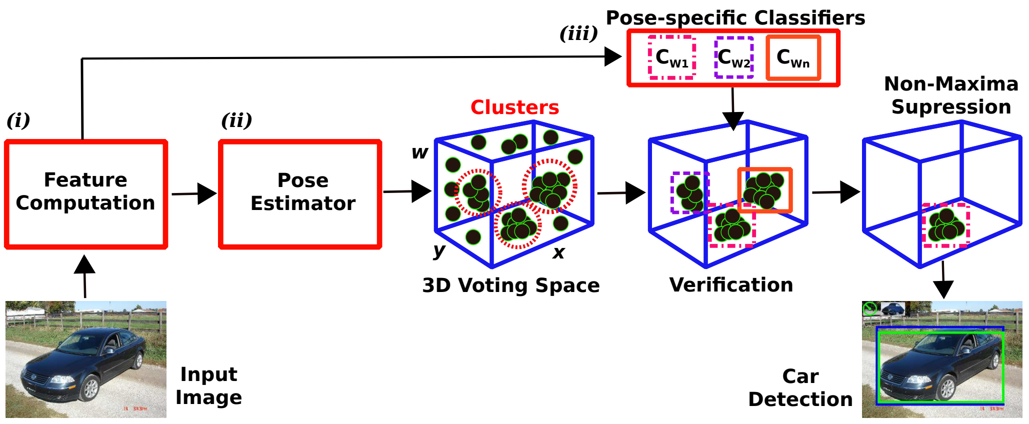


Figure 7.2: Overview of the proposed approach. To detect the 3D pose, given an input image we initially compute a set of shared RFs –Feature Computation–. We then apply the pose estimator to generate several object/pose hypotheses which are verified by the pose-specific classifiers. Non-maximal potential detections are finally filtered out.

Fig. 7.2 depicts an overview of the method, which we describe in detail in the following sections.

Since features are shared among classifiers, their computation is performed in an initial step that is pose independent. This allows an efficient computation of both the pose estimator and the classifiers. For the pose estimation step, each feature is evaluated over the entire image, and casts probabilistic votes for the object/pose center. This yields a set of potential hypotheses –clusters within the voting space–, which are then validated according to a set of specific classifiers. Finally, multiple detections are removed using non-maxima suppression.

7.3 Feature Computation: Random Ferns

The first key element of our approach are the kind of features we use: the Random Ferns. They consist of sets of binary features resulting from simple comparisons on the intensity domain, refer to Sec. 2.1. Yet, and drawing inspiration from [92], we compute RFs over local histograms of oriented gradients –HOGs–, that is, our binary features are simple comparisons between two bins of HOG. The co-occurrence of all feature outputs encodes different image appearances that are used for building the estimator and each one of the classifiers. More formally, each Random Fern $\mathcal{F}$ captures the co-occurrence of M binary features, whose outputs determine the Fern observation z. Therefore, each

Fern maps the image appearances to an $Z = 2^M$-dimensional space, $\mathcal{F} : x \to z$ where x is an image sample and $z \in \{1, 2, .., Z\}$.

In addition, in order to gain in efficiency we share the same RFs among different classes. This was already proposed in the previous chapter, although as we will show in the results section, this previous work does not scale properly for a large number of classifiers since every classifier is independently tested.

7.4 The Pose Estimator

Based on the response of the RFs on an input image, the pose estimator will provide image regions with a high probability of object/pose. For that, we will need to map from the feature domain of the RFs to spatial image locations. This is achieved by means of what we call *Hough-RFs*.

7.4.1 Hough-RFs

In the spirit of Hough-Forests [26, 51], our Hough-RFs encode the local appearance captured by RFs and cast probabilistic votes about the possible location of object poses. Specifically, each Fern output ($\mathcal{F}_i = z$) represents a specific image appearance that has associated a list of distances d_i^z where that appearance has occurred. These distances have been extracted during the learning phase as those ones with higher occurrence over training samples.

The computation of the Hough-RFs is done by evaluating a fixed set of R RFs over the training samples for each object view W_j and an additional background class B. Assuming probability independence among Ferns [64], we define the estimator E_{W_j} as:

$$E_{W_j}(x) = \log \frac{\prod_{r=1}^{R} P(\mathcal{F}_r(x) = z, g|W_j)}{\prod_{r=1}^{R} P(\mathcal{F}_r(x) = z, g|B)} = \sum_{r=1}^{R} \log \frac{P(\mathcal{F}_r(x) = z, g|W_j)}{P(\mathcal{F}_r(x) = z, g|B)}, \qquad (7.1)$$

where g refers to image locations where the Fern $\mathcal{F}_r$, with observation z, has occurred. These locations are always measured from the image center of the object pose j.

The aim is to compute the estimator that maximizes the ratio of probabilities between the object view and background classes –Eq. 7.1– with the objective of selecting the most important image appearances and their locations for the current pose. This is done by selecting the most discriminative locations against the background samples.

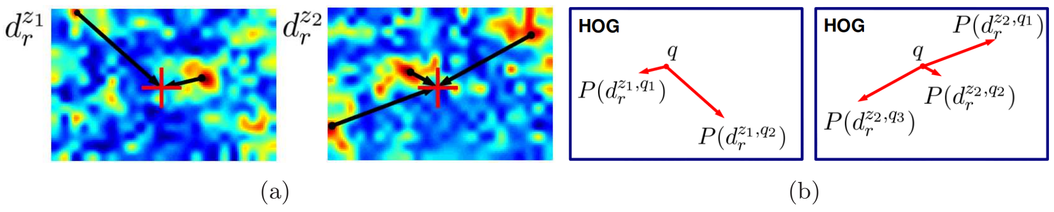


Figure 7.3: The Hough-RFs estimator. (a) The computation of the estimator is carried out by selecting the most discriminative appearance locations against the background category B. Each Fern output ($\mathcal{F}_r = z$) describes a specific image appearance that has associated a list of distances d_r^z indicating where this appearance has occurred over training samples. (b) In runtime, each Fern is evaluated in every image location q to cast probabilistic votes for diverse image locations according to its output. The result is a voting space where its maximum values correspond to possible object instances.

Fig. 7.3(a) shows a simple example where discriminative locations for Fern outputs z_1 and z_2 are chosen. These locations form the lists of distances $d_r^{z_1}$ and $d_r^{z_2}$ which are used to cast probabilistic votes in runtime. These votes are weighted according to their occurrences over training images,

$$P(d_r^{z,q}) = \log \frac{P(\mathcal{F}_r = z, g = q | W_j)}{P(\mathcal{F}_r = z, g = q | B)}. \tag{7.2}$$

Once the estimator has been constructed for every object pose, it is evaluated in runtime as follows: given an input image, a HOG is computed over the whole image for then to test the R RFs. For each image location q –in the HOG space–, each Fern $\mathcal{F}_r$ casts votes for different image locations according to its observation z and its voting list d_r. This voting procedure is illustrated in Fig. 7.3(b). The result of evaluating all RFs is a 3D voting space where their maximum values correspond to object/pose candidates.

7.4.2 Efficient Pose Estimation

As it was exposed in the previous section, the cost of the estimator depends on the number of poses given that each RF must cast votes for the different views. In order to speed up the process, similar to the work of [3], to deal with pose variations, we propose to evaluate the estimator in two consecutive steps. For each Fern $\mathcal{F}_i$, the first step predicts the most likely object pose according to its observation. The second step casts votes only for the estimated pose. In this way, the cost of evaluating the estimator for multiple poses is reduced considerably.

The most likely object pose W_*^r for a Fern $\digamma_r$ is computed with

$$W_*^r = \arg\max_j \log \frac{P(\digamma_r(x)|W_j)}{P(\digamma_r(x)|B)}, \quad j = 1, 2, .., J \tag{7.3}$$

where J is the number of poses. Using the definition of conditional probability, the estimator can be defined as:

$$E(x) = \sum_{r=1}^{R} \log \frac{P(\digamma_r(x)|W_*^r)}{P(\digamma_r(x)|B)} + \sum_{r=1}^{R} \log \frac{P(g|\digamma_r(x), W_*^r)}{P(g|\digamma_r(x), B)}, \tag{7.4}$$

where the first part of the equation is the probability ratio of Fern observations, whereas the second part is the probability of appearance locations given the Fern observations.

In order to reduce the possible locations where a time-consuming pose-specific classifier –see Sec. 7.5– has to be evaluated, we look for the most remarkable hypotheses. This is done by filtering the estimator output, $E(x) > \beta_e$, being β_e a sensitivity parameter. The choice of this parameter is, however, a trade-off between speed of the approach and an increment of false negatives. For instance, if $\beta_e = 0$ each pose-specific classifier is tested on every image position. In this case, the object is not missed but it implies a high computational cost given that all classifiers are tested. In contrast, for increasing values of β_e we speed up the detection phase but with the risk of filtering likely object locations. The estimator in this case reduces the search space and may yield false negatives –missed objects–. The effects of this parameter are evidenced in more detail in Sec. 7.6.

7.5 The Pose-specific Classifier

Each one of the pose-specific classifiers is built independently using a boosting combination of RFs –the BRFs proposed in Chapter 4–. Hereby, a classifier is a set of weak classifiers, where each one of them is based on a Fern selected from the common pool of RFs. This pool is constructed at random and is shared by all classifiers in order to reduce the cost of computing a large number of pose-specific features and to reuse features for constructing different weak classifiers. This idea was used in Chapter 6 to increase the efficiency of BRFs when multiple classes were considered.

The specific classifier $H_{W_j}(x)$ is built to find the Ferns $\mathcal{F}_r$ and locations g_r that most discriminate the positive class from the background. The positive class corresponds to a collection of image samples extracted from the specific object view W_j, whereas the background images B are used for extracting negative samples. The classifier computation is done by means of the Real AdaBoost algorithm – Sec. 2.3–, that iteratively assembles weak classifiers and adapts their weighting values focusing all its effort on the hard samples, which have been incorrectly classified by previous weak classifiers.

The boosted classifier is then

$$H_{W_j}(x) = \sum_{t=1}^{T} h_{W_j}^{(t)}(x) > \beta_{W_j} \ , \tag{7.5}$$

where β_{W_j} is its threshold and $h_{W_j}^{(t)}$ is a weak classifier computed by

$$h_{W_j}^{(t)}(x) = \frac{1}{2} \log \frac{P(\mathcal{F}_t|W_j, g_t) + \epsilon}{P(\mathcal{F}_t|B, g_t) + \epsilon} \ , \tag{7.6}$$

where $\mathcal{F}_t$ is the selected RF that is evaluated at fixed location g_t, measured from the image center, and the parameter ϵ is a smoothing factor. At each boosting iteration t, the probabilities $P(\mathcal{F}_t|W_j, g_t)$ and $P(\mathcal{F}_t|B, g_t)$ are computed using a distribution of weights D over the training samples. This is done as follows,

$$P(\mathcal{F}_t = z|W_j, g_t) = \sum_{\substack{n:\mathcal{F}_t(x_n)=z \\ y_n=+1}} D_t(x_n), \quad P(\mathcal{F}_t = z|B, g_t) = \sum_{\substack{n:\mathcal{F}_t(x_n)=z \\ y_n=-1}} D_t(x_n) \tag{7.7}$$

being x_n and $n = 1, 2, .., N$ the set of training samples. To select the most discriminative weak classifier at each iteration we use, as other previous works, the Bhattacharyya coefficient. In this way, the weak classifier $h_{W_j}^{(t)}$ that minimizes this distance is chosen.

In the present work all pose-specific classifiers are learned using the same parameters, that is, 300 weak classifiers and 10 shared RFs. Since they are learned independently to extract the most relevant features for each pose, the resulting classifiers are very discriminative for each pose and focus on the most relevant object parts.

7.6 Experimental Results

We validated the proposed method using two public and recent datasets: the Savarese car dataset [72] and EPFL car dataset [65]. They are described in more detail following.

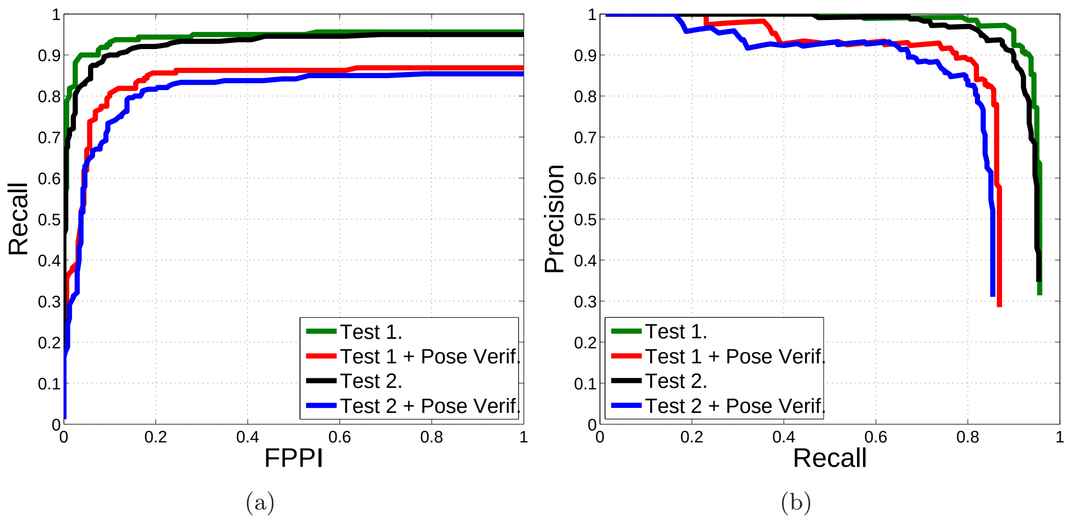


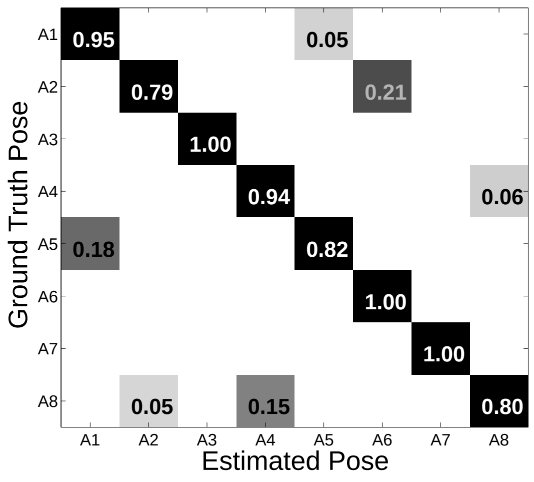
Figure 7.4: Savarese dataset. Car detection using different evaluation approaches. (a) ROC curves. (b) Recall-Precision curves. We see that pose verification reduces detection rates because some pose estimations are incorrect, due mainly to confusions of the true with the symmetric pose pose.

Savarese car dataset [72]. This dataset has multiple views of 10 cars in outdoor settings. For each car, images under 8 different angles, 2 camera heights and 3 distances are available. We train and test our detector using two different sets of images, Test 1 and Test 2. Test 1 is made by 320 car images without the largest distance, whereas Test 2 contains the entire set of 480 images. The first 5 cars of each set are used for training and the rest are used for testing.

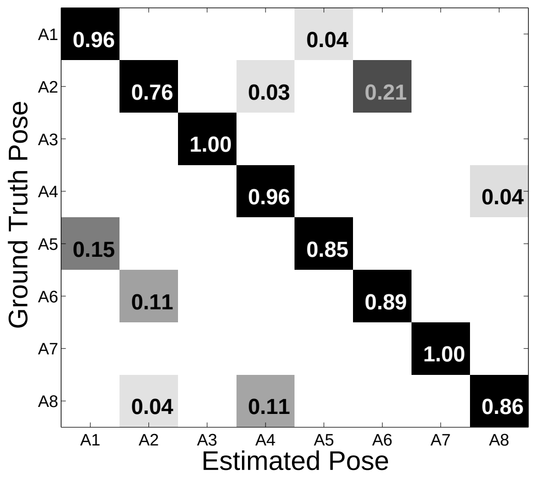
EPFL car dataset [65]. This dataset contains cars under multiple views, light changes and varying backgrounds. The set of images is formed by images collected from 20 specific cars rotating on a platform. The first 10 cars are used for learning the estimator and each one of the pose-specific classifiers. The remaining 10 cars are used for testing [65]. For this dataset, 32 pose-specific classifiers corresponding to 16 views and 2 different aspect ratios have been learned.

7.6.1 Savarese car dataset

Detection performance of our approach on this dataset is shown in Fig. 7.4. We depict the results of just detection –Test 1,2– and detection plus pose estimation –Test 1,2 + Pose Verif–. Note that for images with the largest distance, Test 2, the detection rates

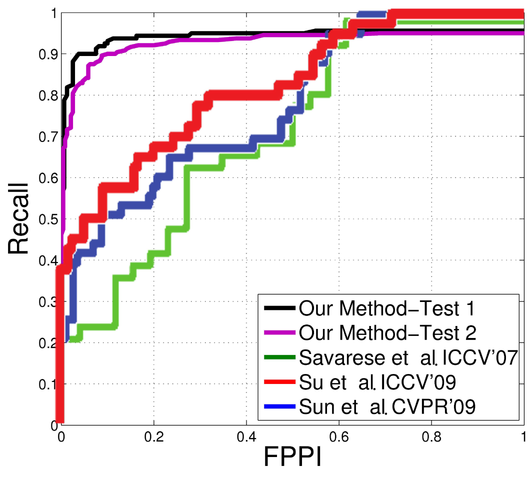


(a)

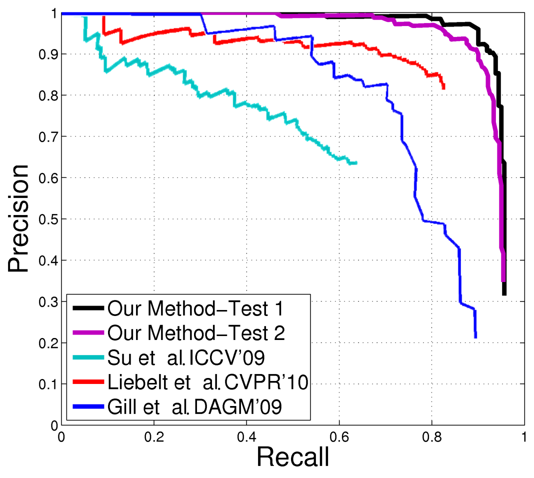


(b)

Figure 7.5: Savarese dataset. (a,b) Confusion matrices for Test 1 and Test 2.



(a)

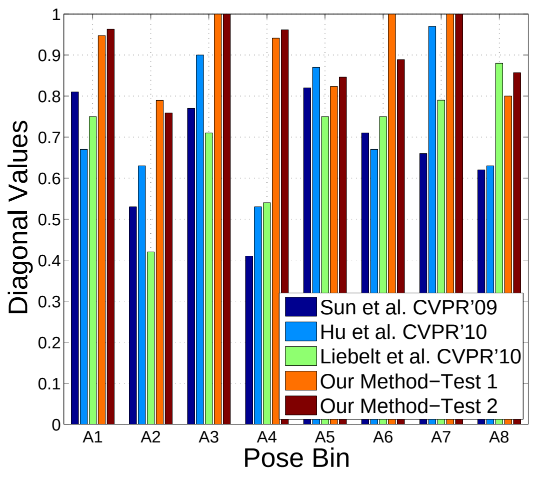


(b)

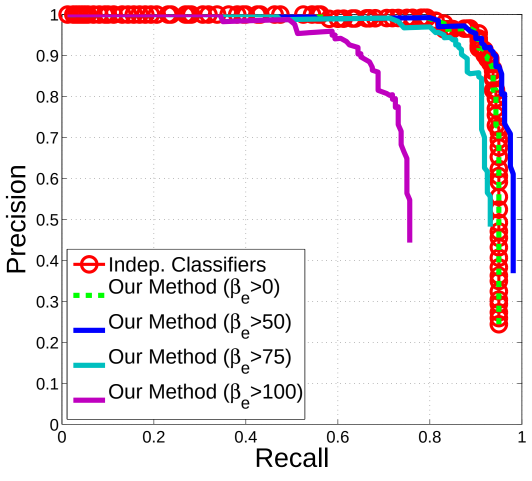
Figure 7.6: Savarese dataset. Comparison against state of the art. (a) ROC curves for our method and some recent works. (b) The comparison is done using the Recall-Precision plots. (c) Comparison in terms of the diagonal values of the confusion matrix.

are lightly reduced. This is because this test contains cars at multiple scales, including the largest distance. Fig. 7.5(b,c) show the confusion matrix both for Test 1 and Test 2, respectively. Observe that only a small fraction of detections are incorrect, and usually correspond to confusions only of symmetry.

The ROC curves of Fig. 7.6(a) and the Recall-Precision plots of Fig. 7.6(b) compare

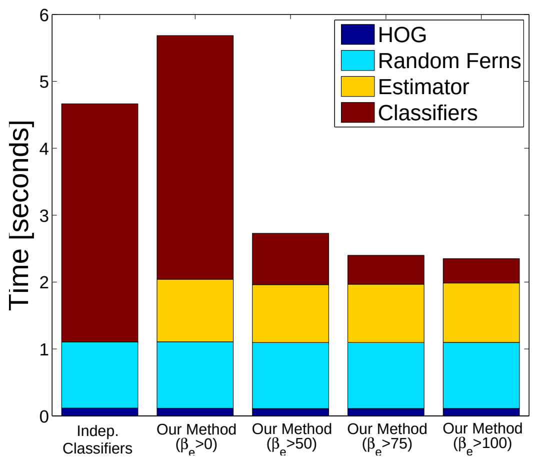


(a)

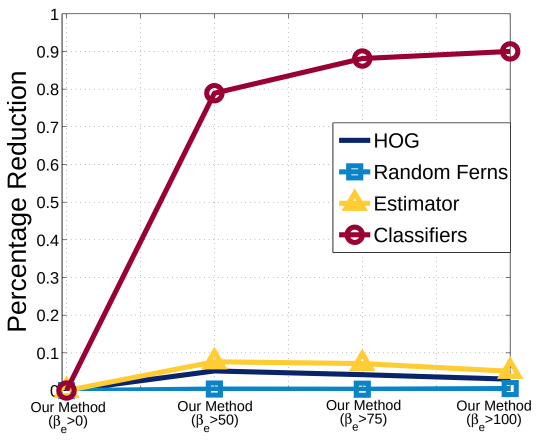


(b)

Figure 7.7: Savarese dataset. (a) Comparison with other methods in terms of the diagonal values of the confusion matrix. (b) Detection according to the sensitivity parameter β_e.



(a)



(b)

Figure 7.8: Savarese dataset. Efficiency in terms of the sensitivity parameter β_e. (a) Detection times. (b) Computational reduction of the proposed approach.

our approach with state of the art methods [27, 33, 46, 72, 83, 84]. In both cases, our method outperforms the detection rates of other approaches. Fig. 7.7(a) compares the methods in terms of pose classification. Note again that the proposed method yields better results. A few sample results are shown in Fig. 7.1.

Fig. 7.8(a), depicts the detection times of our approach for different values of the

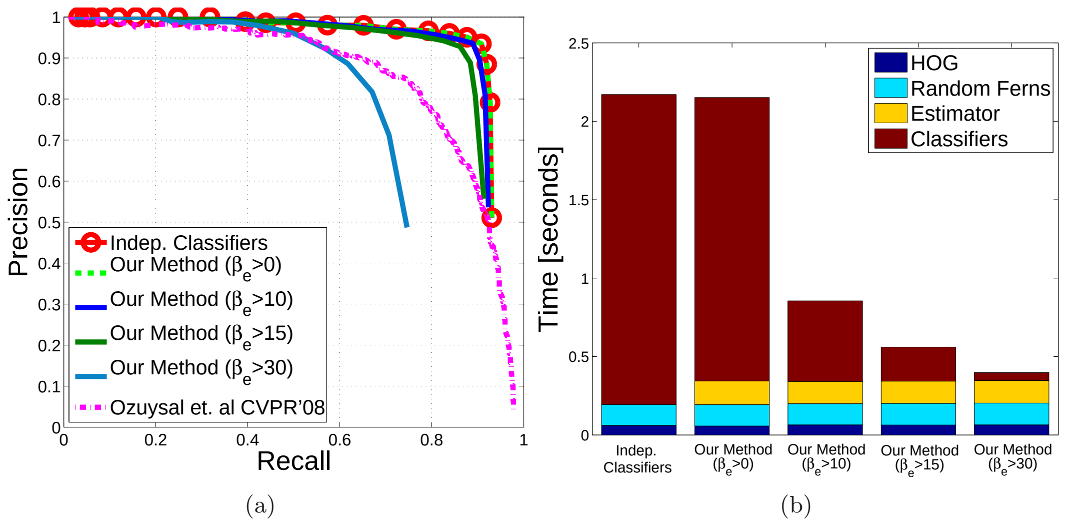


Figure 7.9: EPFL dataset. Detection and efficiency. (a) Recall-Precision plots of the proposed method and the state-of-the-art [65]. (b) Detection times for several values of the parameter β_e.

sensitive parameter β_e. We also show an additional method that would test all the pose-specific classifiers –*Indep. Classifiers*–. It can be seen that the efficiency of our method is increased for larger magnitudes of β_e, see Fig. 7.8(b). However, this is at expense of a reduction in the recall rate, because the estimator misses correct object hypotheses. This can be observed at Fig. 7.7(b).

7.6.2 EPFL car dataset

Detection rates for this dataset using several values of β_e are shown in Fig. 7.9(a). Also the performance curve reported by [65] is depicted for comparison purposes. We see that our method consistently outperforms this work. On the other hand, Fig. 7.9(b) plots the detection times, and shows the efficiency of our method by reducing the time of evaluating the set of pose-specific classifiers.

To measure the viewpoint estimation accuracy of our approach on this dataset we build again the confusion matrix –Fig. 7.10(a)–. We can see that most estimations are correct, showing a diagonal line. This performance is similar to the results reported in [65], where most of incorrect estimations appear on symmetric points of view. This is because there is a strong similarity among these views. This issue is represented in

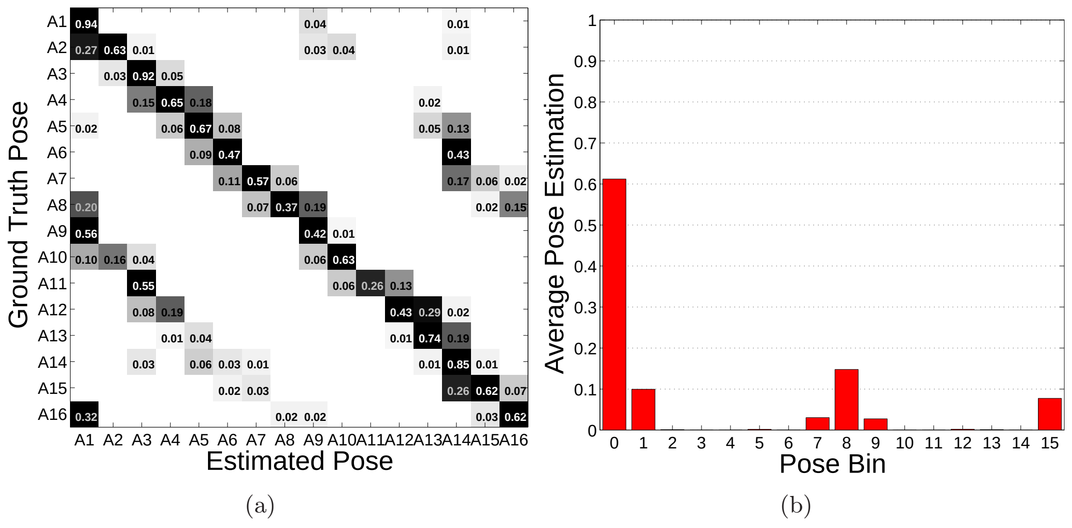


Figure 7.10: EPFL dataset. Car pose estimation. (a) Confusion matrix. Incorrect pose estimations occur mainly at opposite views because of their strong similarities. (b) Distribution of error for each pose bin.

Fig. 7.10(b), where the distribution of error among pose bins is shown. Most of pose estimations are correct –i.e., they belong to bin 0–, but a few of them appear at opposite and adjacent pose bins. Fig. 7.11 shows some sample detections on this database.

7.7 Summary

In this chapter we introduced an efficient strategy to test multiple classifiers with a decoupled approach consisting of a pose estimator and a set of pose-specific classifiers. This method has reported high detection rates and efficiency for the problem of detecting cars from multiple vantage points. The estimator filters out image locations to yield potential candidates where specific classifiers are then evaluated.

To increase efficiency, we propose to compute all the specific classifiers and the estimator using a reduced set of features –Random Ferns–. This allows to reduce the cost of evaluating a large number of features and to decompose the detection process into two stages. The first one tests the RFs over the input image, whereas the second one computes the estimator and the corresponding specific classifier. The benefit is that feature computation is independent of the number of classifiers.

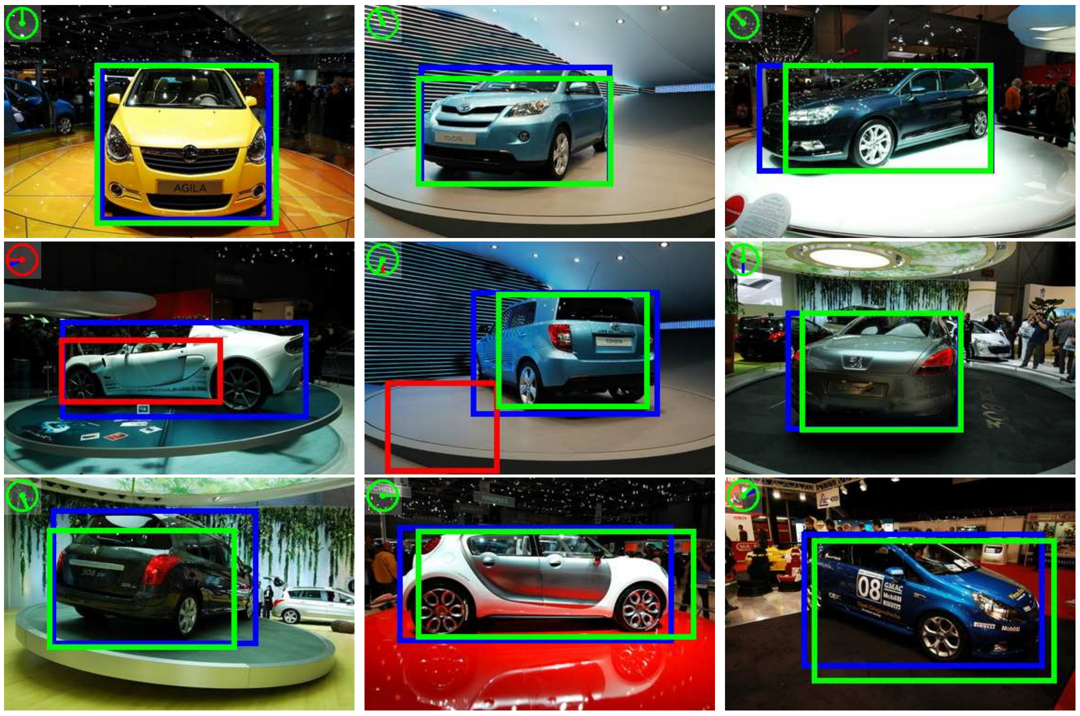


Figure 7.11: EPFL dataset. Sample Results. Correct detections are depicted by green rectangles, whereas false positives are indicated by red ones. The ground truth is shown by a blue rectangle. The circle located at top and left indicates the estimated viewpoint.

With respect to the ideas presented in the previous chapters in this dissertation, this work integrates previous ideas for the problem of detecting 3D objects in an efficient way. In detail, we use a two-step detection strategy that combines BRFs –Chapter 4– and feature sharing –Chapter 6– to compute a pose estimator and a set of pose-specific classifiers.

Chapter 8

Conclusions and Future Work

This chapter provides the conclusions and contributions of this dissertation. The conclusions, as well as the future work, are organized by chapters.

- **Chapter 2**. This chapter presented some computer vision techniques and algorithms for the computation of efficient features and robust object classifiers. These techniques have been studied and used through this dissertation to build novel approaches for object detection in challenging conditions. This chapter, in summary, gave the background for the following chapters.

- **Chapter 3**. In this chapter, a novel method for object detection under in-plane rotations was proposed. This method is based on Haar-like features for fast feature computation. However, and in contrast to other works, the method integrates the steerable filters property to steer efficiently a set of Haar-based features. Besides, an estimator that predicts locally the object orientation in images was also introduced. This particular combination of ideas resulted in an efficient and robust approach that allows to detect specific objects using a more representative set of features. The results of this chapter were published in [95, 96, 97].

  Future work. As the orientation estimator was based on a fixed –rectangular– support region, the method is sensitive to changes in object size. This estimator is also conditioned to global object statistics, such as, the mode on the distribution of oriented gradients. This limits the ability of the estimator for predicting correctly the object orientation. Therefore, a feasible line of research is to create a new and more sophisticated estimator –i.e, based on the Hough transform–, where local features may cast probabilistic votes about the object center and its orientation. This methodology would allow detecting objects and computing their orientations from local estimations using a voting space.

- **Chapter 4**. In this chapter, the Boosted Random Ferns –BRFs– were proposed to compute efficiently and robustly object classifiers that in difficult imaging conditions achieve remarkable detection rates. In comparison with original Random Ferns, the proposed method computes, using AdaBoost, the most robust and discriminative features on local histograms of oriented gradients -HOGs- in order to increase the robustness of the method against lighting changes and intra-class variations. This chapter was the bulk of [92, 94]. Furthermore, BRFs are at the base of the following chapters. In Chapter 5, for instance, BRFs were used to detect objects under planar rotations, and Chapter 6 proposed BRFs to compute multiple object classifiers.

  Future work. We are considering an online version of BRFs for object detection and tracking in videos. The idea is to compute an object classifier that is updated and improved during the image sequence by tracking and detecting the object, and by bootstrapping the classifier [28, 29, 37]. This idea would be of great interest for robotics applications, automatic object modeling, human-robot interfaces, etc.

  Also, more recent and sophisticated boosting algorithms could be assessed and used with the aim of improving the resulting object classifier [4, 21, 100, 82].

- **Chapter 5**. An efficient approach to cope with the detection of object categories that might have rotations in the image plane was proposed in this chapter. Particularly, a two-step detection approach, consisting of an object estimator and classifier, was presented to reduce the computational cost of the detector. This method showed reducing considerably the search space during the detection phase using the object estimator, while kept high detection rates. Both the estimator and classifier were computed using BRFs. This chapter also presented a new database specially designed to evaluate the proposed method. This database contains motorbikes with in-plane rotations. The method reported remarkable detection rates on this dataset with low rates of false positives. This work was presented in [92].

Future work. Similarly to the estimator presented in Chapter 3, performance of the estimator is conditioned to the object size and its aspect ratio. This is because the estimator was computed using a rectangular and fixed window. This is an important drawback when we consider object categories with larger size variations –i.e, cars– because the choice of the window size becomes critical. In consequence, a potential work is to develop an orientation estimator that considers changes in size. In Chapter 7 was proposed an estimator based on the Hough transform that overcame this problem. However, this idea would require the computation of rotation-invariant features. That is, to create a new type of Random Ferns invariant to orientation changes. Shared features, proposed in Chapter 6 to compute BRFs, are also future work.

- **Chapter 6**. In this chapter, BRFs were used for the multi-object detection problem. More precisely, and aiming efficiency in the detection phase, we proposed to share the same set of features –RFs– to compute each one of the object classifiers. This speeded up detecting multiple object in images since that feature computation was common for all object detectors, and was done only once. The experimental evaluation showed that with a reduced set of RFs, it is possible to learn different object classifiers and achieve, at the same time, high detection rates in some standard databases. The results of this chapter were published originally in [93], and the idea was used in following articles [91, 94].

  Future work. We have seen that a shared and random set of feature allows to compute multiple classifiers. However, we are considering the use of a learning step to construct that feature set. This probably would increase recognition rates since that a more compact and useful feature set might be computed. This also would remove the randomness of our method. Another potential future work is to augment the efficiency of the method by reducing the number of classifier evaluations, given that each classifier is tested by separated. To this end, the JointBoosting algorithm [87] could be a reference work to share not only features but also weak classifiers among object categories.

- **Chapter 7**. This chapter addressed the multi-view object detection using efficient and robust techniques. The proposed method combines sinergically BRFs, sharing features and a two-step detection approach. In contrast to Chapter 6, where all classifiers are tested by separated, a novel pose estimator is presented in this chapter to avoid testing the set of pose-specific classifiers individually. This estimator –Hough-RFs– is efficient and robust to changes in object size, contrary to estimator introduced in Chapter 5. The method showed experimentally to be fast and achieved high detection rates on some popular 3D object datasets. This work was presented in [91].

  Future work. Although this work was focused for the detection of objects from multiple views, we think that it might be extended for multi-view and multi-object detection. However, there are issues that should be considered before, such as, the voting space given by the estimator grows proportionally to the amount of views and objects. A feasible line of research is the computation of hierarchies from object parts [20, 63]. This would allow to transfer information among views and object categories, and would yield a compact and reduced representation with which to describe multiple objects [35].

Bibliography

[1] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *European Conference on Computer Vision*, 2008.

[2] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[3] K. Ali, F. Fleuret, D. Hasler, and P. Fua. Joint pose estimator and feature learning for object detection. In *International Conference on Computer Vision*, 2009.

[4] K. Ali, D. Hasler, and F. Fleuret. Flowboost – appearance learning from sparsely annotated video. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[5] D. Ballard and L. Wixson. Object recognition using steerable filters at multiple scales. In *Workshop on Qualitative Vision*, 1993.

[6] O. Barinova, V. Lempitsky, and P. Kohli. On detection of mutiple object instances using hough transform. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[7] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision*, 2006.

[8] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions Pattern Analylis and Machine Intelligence*, 24(4):509–522, 2002.

[9] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *International Conference on Computer Vision*, 2007.

[10] A. Bosch, A. Zisserman, and X. Munoz. Image classification using rois and multiple kernel learning. *International Journal of Computer Vision*, pages 1–25, 2008.

[11] Q. Chen, N. D. Georganas, and E. M. Petriu. Real-time vision-based hand gesture recognition using haar-like features. In *IEEE Instrumentation and Measurement Technology Conference*, 2007.

[12] F. Crow. Summed-area tables for texture mapping. In *ACM SIGGRAPH*, 1984.

[13] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge 2007 (voc2007) results. www.pascalnetwork.org/challenges/voc/voc2007/workshop. 2007.

[15] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[16] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[17] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[18] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Transactions Pattern Analylis and Machine Intelligence*, 30(1):36–51, 2008.

[19] V. Ferrari, F. Jurie, and C. Schmid. Accurate object detection with deformable shape models learnt from images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[20] S. Fidler and A. Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[21] F. Fleuret. Multi-layer Boosting for pattern recognition. *Pattern Recognition Letters*, 30:237–241, 2009.

[22] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions Pattern Analylis and Machine Intelligence*, 13(9):891–906, 1991.

[23] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[24] Y. Freund and R. E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[25] M. Fritz, B. Leibe, B. Caputo, and B. Schiele. Integrating representative and discriminant models for object category detection. In *International Conference on Computer Vision*, 2005.

[26] J. Gall and V. Lempitsky. Class specific hough forests for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[27] G. S. Gill and M. D. Levine. Multi-view object detection based on spatial consistency in a low dimensional space. In *German Association for Pattern Recognition*, 2009.

[28] H. Grabner and H. Bischof. On-line boosting and vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[29] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *European Conference on Computer Vision*, 2008.

[30] C. Gu and X. Ren. Discriminative mixture-of-templates for viewpoint classification. In *European Conference on Computer Vision*, 2010.

[31] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988.

[32] D. Harwood, T. Ojala, M. Pietikainen, S. Kelman, and S. Davis. Texture classification by center-symmetric auto-correlation, using kullback discrimination of distributions. *Technical Report, CAR-TR-678*, 1993.

[33] W. Z. Hu and S. Zhu. Learning a probabilistic model mixing 3d and 2d primitives for view invariant object category recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[34] C. Huang, H. Ai, Y. Li, and S. Lao. Vector boosting for rotation invariant multi-view face detection. In *International Conference on Computer Vision*, 2005.

[35] S. J. Hwang, F. Sha, and K. Grauman. Sharing features between objects and their attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[36] M. Jacob and M. Unser. Design of steerable filters for feature detection using canny-like criteria. *IEEE Transactions Pattern Analylis and Machine Intelligence*, 8(26):1007–1019, 2004.

[37] Z. Kalal, J. Matas, and K. Mikolajczyk. Pn learning: Bootstrapping binary classifiers by structural constraints. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[38] C. Lampert, M. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[39] I. Laptev. Improving object detection using boosted histograms. *Image and Vision Computing*, 27(5):535–544, 2009.

[40] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Transactions Pattern Analylis and Machine Intelligence*, 27(8):1265–1278, 2005.

[41] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[42] A. Lehmann, B. Leibe, and L. Van Gool. Fast prism: Branch and bound hough transform for object class detection. *International Journal of Computer Vision*, 94(2):175–197, 2010.

[43] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1-3):259–289, 2008.

[44] B. Leibe, K. Mikolajczyk, and B. Schiele. Segmentation based multi-cue integration for object detection. In *British Machine Vision Conference*, 2006.

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

[46] J. Liebelt and C. Schmid. Multi-view object class detection with a 3d geometric model. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[47] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *International Conference on Image Processing*, 2002.

[48] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, 1998.

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

[50] J. MacQueen. Some methods for classification and analysis of multi-variate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[51] S. Maji and J. Malik. Object detection using a max-margin hough transform. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[52] R. Manduchi, P. Perona, and D. Shy. Efficient deformable filter banks. *International Journal of Computer Vision*, 46:1168–1173, 1998.

[53] J. Matas, O. Chum, M. Urba, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference*, 2002.

[54] K. Mikolajcyk and C. Schmid. An affine invariant interest point detector. In *European Conference on Computer Vision*, 2002.

[55] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[56] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

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

[58] T. Mita, T. Kaneko, B. Stenger, and O. Hori. Discriminative feature co-occurrence selection for object detection. *IEEE Transactions Pattern Analylis and Machine Intelligence*, 30(7):1257–1269, 2008.

[59] A. Monroy, A. Eigenstetter, and B. Ommer. Beyond straight lines - object detection using curvature. In *International Conference on Image Processing*, 2011.

[60] J. Mutch and D. Lowe. Multiclass object recognition with sparse, localized features. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[61] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[62] T. Ojala, M. Pietikainen, and D. Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition Letters*, 29(1):51–59, 1996.

[63] A. Opelt, A. Pinz, and A. Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[64] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[65] M. Ozuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multi-view object localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[66] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *International Conference on Computer Vision*, 1998.

[67] P. Perona. Deformable kernels for early vision. *IEEE Transactions Pattern Analylis and Machine Intelligence*, 17:488–499, 1991.

[68] F. Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[69] N. Razavi, J. Gall, and L. Van Gool. Scalable multi-class object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[70] H. Riemenschneider, M. Donoser, and H. Bischof. Using partial edge contour matches for efficient object category localization. In *European Conference on Computer Vision*, 2010.

[71] H. A. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[72] S. Savarese and L. Fei-Fei. 3d generic object categorization, localization and pose estimation. In *International Conference on Computer Vision*, 2007.

[73] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *International Conference in Machine Learning*, 1999.

[74] B. Schiele and J. L. Crowley. Recognition without correspondence using multidimensional receptive field histograms. *International Journal of Computer Vision*, 36(1):31–50, 2000.

[75] F. Schroff, A. Criminisi, and A. Zisserman. Object class segmentation using random forests. In *British Machine Vision Conference*, 2008.

[76] H. Schweitzer, J. W. Bell, and F. Wu. Very fast template matching. In *European Conference on Computer Vision*, 2002.

[77] J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *International Conference on Computer Vision*, 2005.

[78] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[79] E.P. Simoncelli and H. Farid. Steerable wedge filters for local orientation analysis. *Technical Report CS–06–08*, 5(9):1377–1382, 1996.

[80] E.P. Simoncelli, W.T. Freeman, E.H. Adelson, and D.J. Heeger. Shiftable multi–scale transforms. *IEEE Transactions on Information Theory*, 38(2), 1992.

[81] S. M. Smith and J. M. Brady. Susan - a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[82] J. Sochman and J. Matas. Waldboost–learning for time constrained sequential detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[83] 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.

[84] M. Sun, H. Su, S. Savarese, and L. Fei-Fei. A multi-view probabilistic model for 3d object classes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[85] P.C. Teo and Y. Hel-Or. Design of multi-parameter steerable functions using cascade-basis reduction. *IEEE Transactions Pattern Analylis and Machine Intelligence*, 21(6):552–556, 1999.

[86] B.M. ter Haar Romenij. *Front-End Vision and Multi-Scale Image Analysis.* Kluwer Academic Publisher, 2003.

[87] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions Pattern Analylis and Machine Intelligence*, 19(5):854–869, 2007.

[88] A. Toshev, B. Taskar, and K. Daniilidis. Object detection via boundary structure segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[89] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2008.

[90] O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. In *European Conference on Computer Vision*, 2006.

[91] M. Villamizar, H. Grabner, J. Andrade-Cetto, A. Sanfeliu, L. V. Gool, and F. Moreno-Noguer. Efficient 3d object detection using multiple pose-specific classifiers. In *British Machine Vision Conference*, 2011.

[92] M. Villamizar, F. Moreno-Noguer, J. Andrade-Cetto, and A. Sanfeliu. Efficient rotation invariant object detection using boosted random ferns. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[93] M. Villamizar, F. Moreno-Noguer, J. Andrade-Cetto, and A. Sanfeliu. Shared random ferns for efficient detection of multiple categories. In *International Conference on Pattern Recognition*, 2010.

[94] M. Villamizar, F. Moreno-Noguer, J. Andrade-Cetto, and A. Sanfeliu. Detection performance evaluation of boosted random ferns. In *Iberian Conference on Pattern Recognition and Image Analysis*, 2011.

[95] M. Villamizar, A. Sanfeliu, and J. Andrade-Cetto. Computation of rotation local invariant features using the integral image for real time object detection. In *International Conference on Pattern Recognition*, 2006.

[96] M. Villamizar, A. Sanfeliu, and J. Andrade-Cetto. Orientation invariant features for multiclass object recognition. In *Iberoamerican Congress on Pattern Recognition*, 2006.

[97] M. Villamizar, A. Sanfeliu, and J. Andrade-Cetto. Unidimensional multiscale local features for object detection under rotation and mild occlusions. In *Iberian Conference on Pattern Recognition and Image Analysis*, 2007.

[98] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[99] B. Wu, H. Ai, C. Huang, and S. Lao. Fast rotation invariant multi-view face detection based on real adaboost. In *International Conference on Automatic Face and Gesture Recognition*, 2004.

[100] B. Wu and R. Nevatia. Cluster boosted tree classifier for multi-view, multi-pose object detection. In *International Conference on Computer Vision*, 2007.

[101] P. Yarlagadda, A. Monroy, and B. Ommer. Voting by grouping depedent parts. In *European Conference on Computer Vision*, 2010.

[102] J.J. Yokono and T. Poggio. Oriented filters for object recognition: an empirical study. In *International Conference on Automatic Face and Gesture Recognition*, 2004.

[103] W. Zhang, Q. Wang, and X. Tang. Real time feature based 3d deformable face tracking. In *European Conference on Computer Vision*, 2008.

[104] Q. Zhu, S. Avidan, M. Ye, and K-T Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.