Hierarchical Multiresolution Models for fast Object Detection

A dissertation submitted by **Marco Pedersoli** at Universitat Autònoma de Barcelona to fulfil the degree of **Doctor en Informàtica**.

Bellaterra, April 2012

Director	**Dr. Jordi Gonzàlez i Sabaté** Centre de Visió per Computador Universitat Autònoma de Barcelona.
Co-director	**Dr. Xavier Roca** Centre de Visió per Computador & Dept. de Ciències de la Computació. Universitat Autònoma de Barcelona.
Thesis Committe	**Dr. Tinne Tuytelaars** Department of Electronic Engineering. Katholieke Universiteit Leuven. **Dr. Francesc Moreno Noguer** Institut de Robòtica i Informàtica Industrial. Universitat Politècnica de Catalunya. **Dr. Manuel Jesús Marín Jiménez** Informática y Análisis Numéricos. Universidad de Córdoba. **Dr. Rodrigo Estéban Benenson** Department of Electronic Engineering. Katholieke Universiteit Leuven. **Dr. Oriol Pujol Vila** Dept. Matemàtica Aplicada i Anàlisi, Facultat de Matemàtiques. Universitat de Barcelona.
European Mention Evaluators	**Dr. Marcin Grzegorzek** Institute for Vision and Graphics. University of Siegen. **Dr. Thomas Baltzer Moeslund** Department of Architecture, Design and Media Technology. Aalborg University.

This document was typeset by the author using LaTeX 2ε.

The research described in this book was carried out at the Computer Vision Center, Universitat Autònoma de Barcelona.

ISBN

Printed by Ediciones Gráficas Rey, S.L.

Acknowledgements

Esta tesis se ha podido llevar a cabo gracias al esfuerzo, dedicación y determinación de muchas personas, y es por tal motivo que tengo el más sincero deseo de agradecelas. Indudablemente quiero agradecer mis directores de tesis, Dr. Gonzàlez, Dr. Xavier Roca, Dr. Juan José Villanueva. Sin sus confianza no hubiera podido ni empezar este trabajo. Quiero agradecer a los membros tribunal de esta tesis por haber aceptado con gusto de evaluar mi trabajo. También quiero agradecer Dr. Andrea Vedaldi por su constante ayuda y al Prof. Andrew Zisserman por aceptarme de estancia en su grupo, así como todos los compañeros del lab. En los tedios días en el sotano del CVC he podido compartir los malos y los buenos momentos con muchos compañeros extraordinarios: Bhaskar, Ariel, Murad, Pep, Fahad, Shida, Ogi, Piero, Francesco, German, Shell, Adela. Agradecer también a los pasados y presentes miembros del grupo ISE: Dany, Ignasi, Pau, Carles, Andy, Miguel, Marc, Noha, Wenjuan, Natalia, Zhanwu. Agradecerle por supuesto, al director de esta institución Dr. Lladós y hacer extensible dicho agradecimiento a todos los integrantes del CVC. Como no agradecer a Montse, Mari y Gigi que constantemente están solucionando mil problemas burocráticos. Agradecerles a las chicas de administración Anita, Raquel, Eva y Claire que siempre están dispuestas a ayudar. Sin lugar a duda, mis principales agradecimientos les van a mis muchas familias. La de aquí: Maurcio, Ana, Lena, Andrea, Roberto con los cuales simepre he tenido amistad y confianza. La de los que estan lejos pero no se olvidan: Stefano, Manz, Gabri y Manu, disfrutando de las noches de vela. La de siempre: Anna, Piero, Valeria, Mauro, que aunque no estamos muy en contacto, siempre nos entendemos con pocas palabras. Y por fin a la que me aguanta cada día, Cristina.

Abstract

Day by day, the ability to automatically detect and recognize objects in unconstrained images is becoming more and more important. From security systems and robots, to smart phones and augmented reality, every intelligent device needs to know the semantic meaning of an image.

This thesis tackles the problem of fast object detection based on template models. Searching for an object in an image is the procedure of evaluating the similarity between the template model and every possible image location and scale. Here we argue that using a template model representation based on a *multiple resolution hierarchy* is an optimal choice that can lead to excellent detection accuracy and fast computation. As the search of the object is implicitly effectuated at multiple image resolutions to detect objects at multiple scales, using also a template model with multiple resolutions permits an improved model representation almost without any additional computational cost. Also, the hierarchy of multiple resolutions naturally adapts to a search over image resolutions, from coarse to fine. This leads to a double speed-up due to: an initially reduced set of coarse locations where to search for the object; a lower cost of evaluating the template model.

The search over resolutions can be effectuated by using a *cascade of multiresolution classifiers*, which saves computation by early stopping the search at coarse level when finding easy negative examples. An alternative approach is to locally but uniformly selecting the most promising detection locations at coarse level and, then, iteratively propagate only these ones to the finer resolutions, saving computation. This procedure, that we call *coarse-to-fine search*, has a speed-up similar to the multiresolution cascade, but a computational time independent of the image content. The coarse-to-fine search is then extended to *deformable parts models*. In this approach, as increasing the model resolution, the hierarchy of models is recursively separated into deformable subparts. In this way, each part can be aligned to the object in the image, producing a better representation and, therefore, an improved detection accuracy with still a reduced computational cost.

We validate the different multiresolution models on several commonly used datasets, showing state-of-the-art results with a reduced computational cost. Finally, we specialize the multiresolution deformable model to the challenging task of *pedestrian detection on moving vehicles*, that requires both high accuracy and real-time performance. We show that the overall quality of our model is superior to previous works and it can lead to the first reliable pedestrian detection based only on images.

Resum

Dia a dia, la capacitat de detectar i reconèixer objectes en imatges automàticament es fa cada vegada més important. Des dels sistemes de seguretat i robots, als telèfons d'última generació i la realitat augmentada, tot dispositiu intel·ligent necessita conèixer el significat semàntic de la imatge.

Aquesta tesi aborda el problema de la detecció ràpida d'objectes a partir de models basats en patrons. La cerca d'un objecte en imatges s'implementa evaluant la similitud entre el model i cada ubicació i escala possibles en una imatge. Aquí s'argumenta que utilitzar una representació d'objectes basada en una *jerarquia de múltiples resolucions* és una opció adequada que pot conduir a una excel·lent precisió i un càlcul molt ràpid. Com, per detectar a múltiples escales, la cerca de l'objecte s'efectua de forma implícita a múltiples resolucions, el fet d'utilitzar un model en múltiples resolucions permet una millor representació de l'objecte, gairebé sense cost computacional addicional. A més, un model multiresolució s'adapta de forma natural a una cerca també en múltiples resolucions en la imatge, des de baixes a altes. Això ens porta a un conjunt d'acceleracions importants, degut a que es poden limitar el conjunt d'ubicacions on fer la cerca de l'objecte a nivells baixos de resolució, el que comporta un cost més reduït en l'avaluació del model.

Una cerca jeràrquica de baixes a altes resolucions es pot fer utilizant una *cascada de classificadors multiresolució*, que elimina facils hipòtesis neagatives utilizant la baixa resolució. Un mètode alternatiu es basa en seleccionar localment, però de manera uniforme, les ubicacions de detecció a resolució baixa y propagarles fins a la resolució més alta. Aquest enfocament alternatiu, que llamem *cerca coarse-to-fine*, té una acceleració i rendiments semblants a la cascada de múltiples resolucions, però en un temps de computació independent del contingut de la imatge. La cerca coarse-to-fine s'ha estès a *models deformables amb partes*. En aquest enfocament, la jerarquia dels models se separa de forma recursiva en les subparts deformables de l'objecte a mesura que augmentem la resolució del model. D'aquesta manera, cada part s'ajusta a l'objecte en la imatge, produint una millor representació i, per tant una millor precisió en la detecció, juntament amb un temps computacional molt reduït.

S'han validat els diferents models de multiresolució en diverses bases de dades conegudes i d'ús comú, mostrant que els resultats arriben a l'estat de l'art, però amb un cost computacional molt reduït. Finalment, es presenta una especialització d'aquest model multiresolució deformable per la tasca de *detecció de vianants des de vehicles en moviment*, que requereix tant una alta precisió com que el rendiment sigui en temps real. S'ha demostrat que la qualitat global del model proposat és superior als treballs anteriors i que té un grau de detecció de vianants fiable i ràpid utilitzant únicament informació de la imatge.

Resumen

Día a día, la capacidad de detectar y reconocer objetos en imágenes automáticamente se hace cada vez más importante. Desde los sistemas de seguridad y los robots, a los teléfonos de última generación y la realidad aumentada, cada dispositivo inteligente necesita conocer el significado semántico de la imagen.

Esta tesis aborda el problema de la detección rápida de objetos a partir de modelos basados en patrones. La búsqueda de un objeto en una imagen es el procedimiento de evaluar la similitud entre el modelo y cada ubicación y escala posible de la imagen. En esta tesis se argumenta que utilizar una representación del modelo de objetos basada en una *jerarquía de resoluciones múltiples* es una opción adecuada que puede conducir a una excelente precisión y un cálculo rápido. Como, para detectar a múltiples escalas, la búsqueda del objeto se efectúa de forma implícita en múltiples resoluciones, utilizar también un modelo de objetos con resoluciones múltiples permite una representación mejor del modelo, casi sin coste computacional adicional. Además, el modelo multiresolución se adapta de forma natural a una búsqueda sobre multiples resoluciones en la imagen, desde bajas a altas. Esto conduce a una doble aceleración debida a: un inicialmente reducido conjunto de ubicaciones en baja resolución donde realizar la búsqueda del objeto; un coste reducido de la evaluación del modelo.

La búsqueda sobre multiples resoluciones puede efectuarse utilizando una *cascada de clasificadores multirresolución*, que elimina los ejemplos negativos en la resolución baja. Un método alternativo se basa en seleccionar localmente, pero de manera uniforme, las mejores detecciones a resolución baja y, luego, propagar estas hipothesis a los siguientes niveles de resolución. Este método, que llamamos *búsqueda coarse-to-fine*, tiene una aceleración parecida a la cascada de multiples resoluciones, pero el coste computacional es independiente del contenido de la imagen. La búsqueda coarse-to-fine se extiende a *modelos deformables con partes*. En este enfoque, la jerarquía de los modelos se separa de forma recursiva en las subpartes deformables a medida que aumenta la resolución del modelo. De esta manera, cada parte puede ajustarse al objecto en la imagen, produciendo una mejor representación y, por tanto, una mejor precisión en la detección con un tiempo computacional muy reducido.

Se han validado los diferentes modelos de multirresolución en varias bases de datos de uso común, mostrando que los resultados alcanzan el estado del arte, pero con un coste computacional reducido. Por último, se presenta una especialización del modelo de multirresolución deformable para la tarea de *detección de peatones desde vehículos en movimiento*, que requiere tanto una alta precisión como un rendimiento en tiempo real. Se ha demostrado que la calidad global de nuestro modelo es superior a los trabajos anteriores y que puede producir una detección fiable de peatones basada solamente en imágenes.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

We define and contextualize the task of object detection. In this chapter we explain motivations, applications and objectives of this work. Next, we discern between object detection and other similar tasks, like segmentation and pose estimation. Also, we introduce the main paradigm, based on learning from examples, used from the current algorithms for object detection. Finally, we present some standard ways to evaluate an algorithm for object detection.

1.1 Motivation

Every day, in every instant of our lives, our bodies are surrounded by a reality that we need to know and interact with. For doing that, even before being born, we begin a process of learning a discrete set of objects or agents, generally with a well defined spatial extent and semantic characteristics that make reality and reasoning about it simple and possible.

Imagine to describe a scene, for instance an office, without using any commonly known object, like chair, desk, monitor, window, etc... For us it is very hard, if not impossible, any useful reasoning without these objects. Thus, one of the main challenges and first step for the artificial intelligence is to learn the basic blocks of our world. We can think about it as a child learning the words of a language, but in a more abstract level, a kind of mental words.

We can see it as a clustering, where the basic aim is to obtain a better understanding of the world, to be able to better perform and, in terms of evolution, to survive. Being able to learn the best representation of a certain world for a certain aim is a key-point for every intelligent algorithm. With the right representation every problem can be solved easily and with simple reasoning.

Currently, we are still far from letting machines learning the basic agents of our world in a quite unsupervised way as we do. However, pattern recognition and,

more specifically, object detection aim to this in a more restricted world and in more supervised settings.

In object detection we restrict the domain of application to sight, which is still the most complex sense we own, and we specifically define the low level task of detecting what we define as our agents. We make machines learning what is the appearance of a human for instance, but not as induction of the fact that it is useful for its task, but as emulating our own representation of the world.

In a practical sense, even if the task is easier, at short time it is still useful for several applications and it can reduce the semantic gap between our representation of reality and what is generally used in machines to reason about.

1.1.1 From motion to appearance

Up to five years ago one of the most important tasks for understanding images was considered to be tracking [107, 20, 147]. That is, given a sequence of images being able to keep track of the moving object, maintaining for each a unique identity. Of course, also nowadays tracking is fundamental for computer vision, however, for the particular task of image understanding, a new paradigm has been introduced: instead of perceiving *where* the objects move, first of all we want to know *who* are the objects.

Here appears object detection, which is able to find, in a single image, thus without any temporal information, the presence and position of object categories. In this sense, knowing that an object moved on a certain trajectory gives a limited information, which is useful only on very specific context, like a person crossing a street or entering in a building. In both cases, the street and the building are given a priori. In contrast, when using object detection, if enough detectors are available, the algorithm can infer dynamically the scene content without any external help. In this case, for example, street and building can be two of the detected class and the algorithm can infer the action of crossing and entering even without motion information.

We can think about tracking as following the position of a certain object over time. Thus the algorithm has essentially the task of localizing the given object or target in a position "close" to the previous. In contrast, object detection has the more challenging task of finding the object everywhere in the scene, without any prior assumption apart the object appearance. For doing this it is necessary a more sophisticated machinery, which is learning from examples (see section 1.3.1).

From a practical point of view and due to its generality, object detection can also be used as likelihood of the target in a tracking framework [13, 2]. In this way it can help to solve many problems that generally affect tracking algorithms. If the algorithm is based on background subtraction, the method works only with static camera and smooth-changing light conditions. If the algorithm is based on appearance tracking, the object initialization is a problem as well as the appearance drift. Most of these problems are easily solved with detection: no need of static camera, image conditions can change abruptly without any problem, no appearance drift because the object model is built off-line.

1.1.2 Challenges

Figure 1.1: Examples of issues that has to face object class localization from [35]. (a) scene illumination, (b) camouflage, (c) rotation and viewpoint, (d) body articulation, (d) intraclass variability, (e) occlusions.

If you ask to some non experts to rank the difficulty of human tasks they will most probably consider classification and detection as easy tasks. We think recognizing and localizing objects is very easy because we use them every instant of our life, so that they look like it is an innate capability. In reality, object class localization in unconstrained environment is a very difficult task. In the following list we present the class of problems one has to solve when dealing with object classification and detection.

- **illumination**: the same object should be recognized under different illumination conditions, from low ambient light to backward and direct light for instance. As we will see in Chapter 2, often illumination issues are solved at feature level, using an illumination invariant descriptor.

- **camouflage**: in many situations the object we are interested in detect and the background have very similar colors and texture. This makes the detection task even harder and specific features and learning should be applied in order to overtake these issues. A typical example of camouflage are animals and insects, that often tend to be as similar as possible to their surrounding environment to avoid possible predators.

- **rotation and viewpoint**: the object class should be recognized from every possible rotation and point of view. While for some objects, like a ball, the point of view is almost irrelevant, for others, the appearance from different

point of view can be very different. Notice that, changing the point of view modifies also the geometrical properties of the objects, producing perspective distortions.

- **deformation and articulation**: certain categories of objects, e.g. animals, are composed of a skeleton which allows them to move limbs and produce complex movements, like running, sitting, jumping etc.. Also, other categories can have different kind of deformations, for example televisions can have different proportions between high and width (aspect ration) and therefore they can be thought as the same object, but with an affine deformation.
- **intraclass variability**: while for some object categories the appearance of different instances are similar, for others, the main feature that makes the class distinctive is semantic and recognition based only on appearance would certainly fail. For instance, a chair is defined for its use more than for its appearance, that can extensively vary among instances. Also in classes where the appearance is more distinctive, like cars, there can be appearance variations. For instance, although quite different, a pick-up as well as a cabriolet belongs to the class car. Logically, using more specific object class can help to reduce the intraclass variability.
- **occlusions**: considering the fact that our vision system projects a 3D world into a 2D surface, it is logical to expect that parts of the objects would be occluded or only partially visible. Also, an object extent can be wider than the field of view of the viewer, therefore part of the object will not be visible. A robust method for object detection should be able to deal with such kind of partial information.

1.2 Applications

The number of tasks where object detection can be worthily employed is huge. Among these tasks, only few of them are ready for real applications, many of them need further technological improvements and some of them have not been even thought yet. In the following we give a general overview of the principal fields of application where object detection can be helpful.

- **Security**: Current applications of object class detection in security are for recognition system. In particular, an effective real-time system for faces detection was proposed in [132], and since that time many security systems used face detection as a preliminary stage of an identity recognition system. In particular, face detection can help to use identification system based on face features, like iris scan, without the need of a specific positioning of the person we want to identify.

  Also person detection in the field of security camera networks is useful. Waiting for an era when the full control of a network of cameras is given to an automatic system, already now, knowing when some humans are in the camera field

is pre-filter used to increase the number of cameras a single person can control. In contrast to face detection, person detection is not mature enough for reliable and real-time systems. This is due to the more complex structure shape and colors that a person can assume. However, a person detector can already be included in modern security systems together with motion segmentation: motion segmentation selects the candidate regions, and the detector distinguishes whether the moving region is a human or not.

- **Intelligent Systems**: For intelligent system we include all systems that can take advantage from vision. One of the first applications of computer vision (also called industrial vision) was for industrial processes. Thank you to the controlled environment, primitive shape detection and segmentation were efficiently used for quality control and piece alignment for industrial robots.

  Today, in the era of the smart devices, the use of detection is starting to find new applications. For instance, the last generation of digital cameras have already included a face detector to tune the camera settings knowing the location of one or more faces.

  The most promising market for possible applications of object detection in the next few years is the mobile phone market. The so called smart phones come with powerful processors, often even GPUs and a high resolution camera, the optimal hardware for computer vision and more specifically object detection applications. For instance, mixing reality with computer graphics (the so called augmented reality) is a new trend that can invade the video-games market (see Kinect) as well as user-phone interaction and advertising. Recognizing and localizing places and objects is the basic step to create a virtual world.

- **Driving assistance**: Driving assistance is one of the fields where object detection can really change and improve our safety. Automatic pedestrian detection is in fact very important for reducing the number of collisions between vehicles and pedestrians. While in other applications like security, object detection can be used together with motion-based segmentation, in a moving vehicle, systems based on motion do not work, thus the entire problem should be solved by detection. This makes the challenge even more difficult, because it has been estimated that a system for pedestrian detection to be practically useful in a vehicle should have 1 false positive per day. In contrast, current detection systems have a false positive every frame for a detection rate of the 90%. In current technology, detection systems are associated with other sensors, like infrared vision and range radar. However, it is expected that in a few years detection technology would be mature enough for being used alone.

- **Robot Vision**: Robot vision is one of the first topics where computer vision has been applied, and object detection is the main capability expected from a robot. Currently robotics has not reached yet a point where prototypes can be used for real applications. Apart from industrial robots that are designed for very specific and repetitive tasks, robots that can emulate human activities are not available yet.

In this sense object detection, with its recent advances, can contribute to give new capabilities to the robots. For example, for a robot that has to be a museum guide, it is fundamental to recognize humans to interact with them. Also, any kind of autonomous vehicle can take advantage of the capability of recognizing object, for avoiding obstacles as well as using them as landmarks and reference point for simultaneous localization and mapping.

1.3 Object detection

Object detection is the task to find the presence and location of an object class or category in an image. In the following subsections we explain the general idea of how object detection is performed, we better define the concept of class or category and, finally, we comparatively discriminate localization from other similar tasks.

1.3.1 Learning vs Rules

At the beginning of artificial intelligence, the task of classification was mainly performed using pre-defined rules. For instance, defining a face like an object composed by two eyes, two ears, a nose and a mouth and, subsequently, defining the appearance of each of these parts makes sense. However, as the complexity of the detection problem increases, moving from simple, environment-controlled images, to real and cluttered images, defining valid rules becomes almost impossible. Therefore, nowadays the most used strategy for many computer vision and artificial intelligence tasks, and in particular for object detection, is example based learning. In practice, a set of examples is given and, from these, a learning algorithm has the task of finding the most relevant characteristics to discriminate the object from the rest of the world.

Still, modern systems do not learn directly from pixel level [1]. A modern detection system is composed of a set of predefined rules (like which features to extract or the parts the object is decomposed) together with a set of parameters that need to be learned (like feature importance). From our point of view, as many parameters can be learned from the examples without external supervision or prior rules, more of the systems would be general and powerful.

1.3.2 Level of supervision

When dealing with example-based learning there can be different levels of supervision. The simplest one is the completely supervised problem. In this case all examples have an assigned label. For instance, in Fig. 1.2 (a) we want to classify whether the image represents an elephant or a rhino. In this case each image used for training has an associated label. In Fig. 1.2 (b) we consider a group of training images with full

[1] In this regard deep learning [52] is pushing towards the ambitious aim to learn everything from scratch, although for the moment we think it is not mature enough.

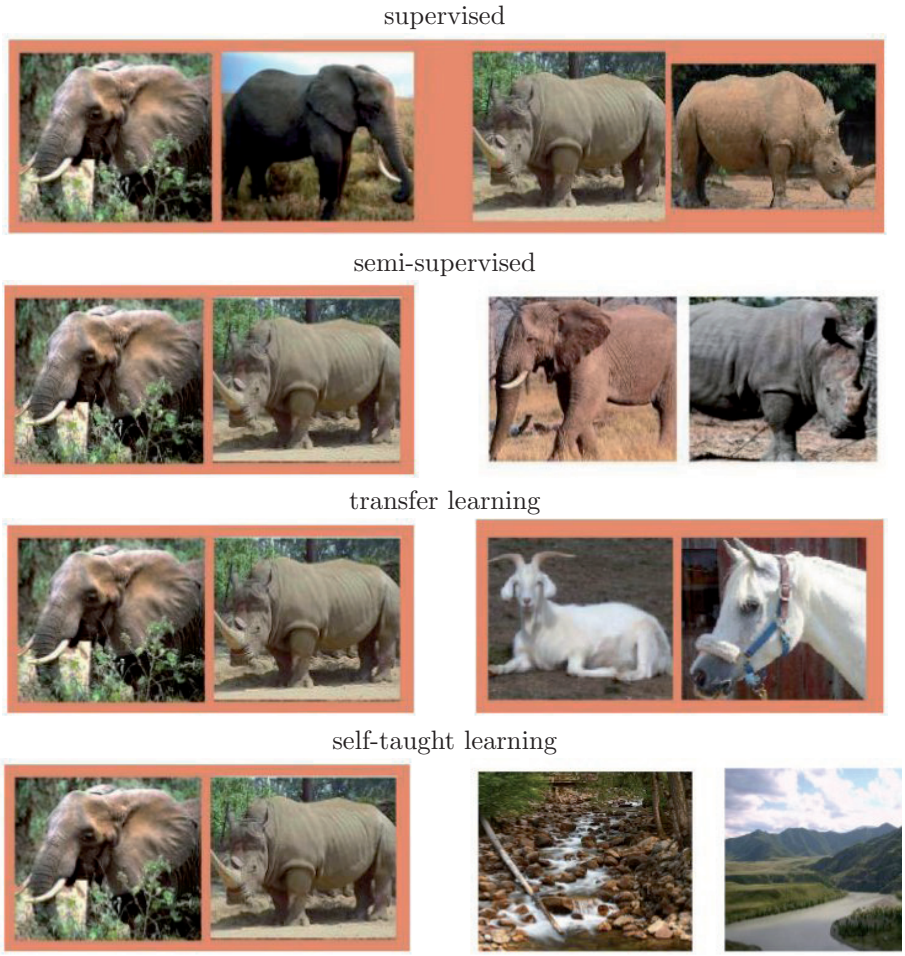
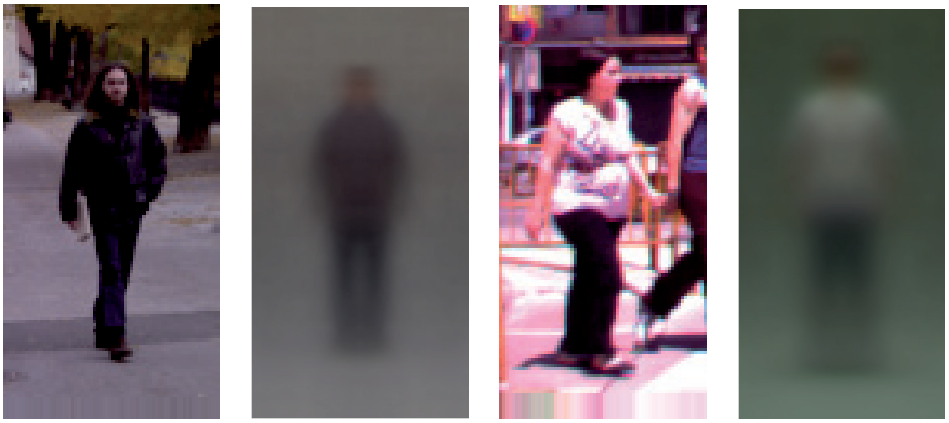


Figure 1.2: Level of supervision: (a) supervised learning, (b)semi-supervised learning, (c) transfer learning (d) self-taught learning. Colored frame represent labeled data. Images from [90].

labeling and others without it. A semi-supervised algorithm can take advantage of the unlabeled data (which is unlabeled but still coming from the same distribution of examples, i.e. each image is elephant or rhino) to better learn a classification function. Transfer learning is another type of learning where training data for a certain class can help to classify another class. In Fig. 1.2 (c) we see that, knowing how to classify elephants from rhino, for instance, can improve the classification of goats from horses. Finally, the weakest level of supervision is when an algorithm can take advantage from data that has still some relation with the original data, but not even the same distribution. For example, in Fig. 1.2 (d) is shown that images of landscapes can still have some statistical information to help to better discriminate

between elephant and rhino. An example of this is sparse coding [139, 90], where low level features are learned to best reconstruct images from unlabeled data. Then, these features are used in a discriminative way using a supervised learning.

1.3.3 What is an Object class



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

Figure 1.3: Comparison of Inria and CVC02 pedestrians. (a) A sample of the INRIA dataset, (b) Average of all samples of INRIA, (c) A sample of the CVC02 dataset, (d) Average of all samples of CVC02.

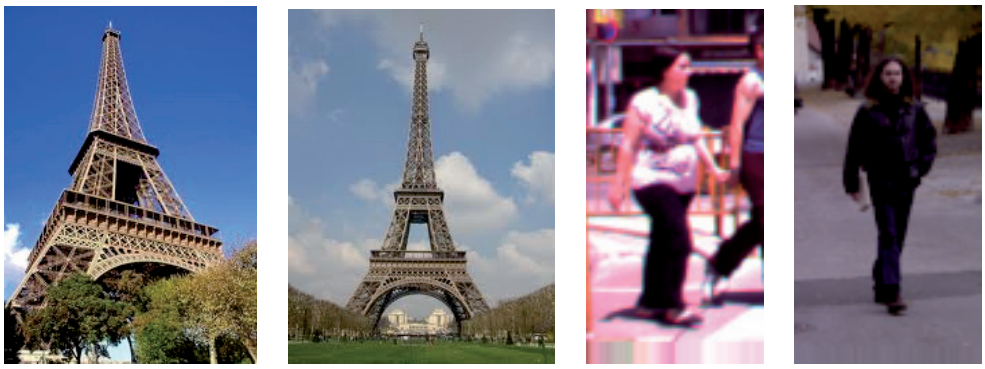
For object class we intend the infinite set of all instances of a certain category. For example, for the class pedestrian we consider all the images of human beings standing in an urban scenario. Unfortunately, this definition is quite loose: i.e. in different collections of images containing humans the appearance model can be very different. As example, in Fig. 1.3 are shown examples of pedestrians from INRIA and CVC02 dataset. Although the definition of pedestrian should be quite clear, it is evident that the two datasets have a different model of human. From the average image, for instance, we can guess that images from INRIA have been mainly taken in winter time or, at least, in cold places because the torso color is dark as generally coats are, while for CVC02 images are probably taken in summer time because the torso color is clear, as usually people in hot places tend to wear white shirts.

This has been investigated in [116], where is shown that the same object category can produce very different models, and testing a model trained on a database on another database can produce very poor results. The authors connect this to the poor quality and generality of the current databases. Although this is partially true, we believe that the main reason of the issue is the fuzzy concept of object category. Let ask to different people what do they consider to be a pedestrian. Although the general idea can be similar, when you ask to identify pedestrian in real images, the answers can vary a lot, exactly as for detectors trained on different databases. This

derives from the general belief that nouns are the optimal way to separate object categories. However this idea is naive and can work only for very specific and reduced number of classes. A possible way to alleviate this problem is to build a hierarchy of classes: from the most general one, i.e something like object, to the most specific one, i.e. a woman wearing blue trousers, red t-shirt, standing close to a horse. In this sense, ontologies try to build-up a hierarchical categorization of "everything". In this direction, some recent works on object detection can be found in [94, 93].

In our work, to avoid such problems, we do not consider the semantic meaning of a certain category; instead, we consider a category as the set of images given as training data. In this way, we separate the problem of object detection in two disjoint tasks: one is the selection of a category, which is performed when the database is created, and the second task is the ability to find instances of objects most similar to the training data, which is the problem we tackle. Assuming that a good detector is able to perform relatively well independently of the object category, if our detector obtains excellent results on some different handcrafted categories is likely to expect that the detector will perform well also on more meaningful categories.

1.3.4 Single instance vs Object class



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

Figure 1.4: Single instance versus class detection. (a) and (b) are two images of exactly the same *instance* of the object. In contrast (c) and (d) are two images of different instances of the same *class*.

The first attempts of object detection were applied to a similar but different problem, which is single instance detection. Single instance detection consists on finding the location of an already seen object. Fig. 1.4 shows the difference between single instance detection and object class detection.

Single instance detection has some similarities to tracking, whose task is to follow or track a certain object over a sequence of consecutive frames. Tracking assumes a certain temporal and therefore spatial coherence between frames and uses this as-

sumption to simplify the object search. In contrast, single instance detection does not assume any spatial coherence between the image where the target object is given, and the image where the target has to be found.

However, single instance detection is still a simplification of object class detection. In single object detection, the object to search for is exactly the same given as target and therefore it has the same characteristics of the given target; changes on the object appearance are only due to illumination and point of view changes. Therefore, if the detection method or the image features are invariant to light condition and viewpoint (at least partially), the method is nothing else than a likelihood search over the possible object locations. Examples of these methods are [66, 77], where the authors employ hough transform and SIFT features to localize flat objects over affine or projective deformations.

In object detection, besides the problem of searching the location of an object in the image, we add also the problem of learning a good summary of the relevant features that are common to all the instances of the class. For example, single instance detection for a book can use features that are specific of the given book to localize it; for example the characters of the title. In contrast, class detection cannot use the characters in the title because it is probable that other books would have other characters. In this case, a book detector should learn more general book features, like their shape, texture, color, etc..

In this sense, in object class detection the learning algorithm has to be able, with a limited set of examples, to extract all the characteristics common to the elements of a given class. In the following, if not differently specified, we will refer to object class detection as object detection.

1.3.5 Localization vs Classification

Another task highly connected with object detection is classification. In classification, given an image, the algorithm has to find to which of a set of given categories the object belongs. In contrast to detection, in classification it is not required to localize the position of the object in the image. Although detection and classification have to solve similar problems, historically they have been approached with quite different techniques: classification is generally based on bag of words techniques, while localization is mostly based on template matching. Both methods are explained in detail in chapter 2. From another point of view we can say that the problem of detection involves the concurrent solution of a classification problem, to distinguish object of a certain class from all the rest and a single instance localization, where the task is to find the location of the object in the image.

As it will be explained in chapter 2, classification can be seen as a sub-problem of object detection. In fact, a possible way to detect an object is to apply the learned classifier on all possible locations of the image, and select those regions where the likelihood of belonging to a certain class is high. In this case therefore, localization is like a classification algorithm where a latent variable representing the object location

has been added.

From this point of view it is easy to understand why the two tasks have been solved in very different ways as mentioned before; while in classification the learned classifier is applied only once per image, in localization it has to be applied from hundreds to many thousands (depending on the method) times. Thus, the computational costs of the two methods are very different and different solutions should be used. In particular, for detection very fast classifiers and techniques to reduce the number of locations where to search should be employed. This is one of the points that we investigate in this thesis. An example of classification and localization are shown in Fig. 1.5 (a) and (b) respectively.

1.3.6 Localization vs Segmentation

Another task that has some similarity with object class localization is *semantic segmentation*. First of all, semantic segmentation must be distinguished from *unsupervised segmentation*. The latter is the task of separating an image into regions (set of connected pixels), based on color, texture, shape or any kind of combinations of these features, without using any semantic supervision i.e learning from examples. These regions, in general, do not have any semantic meaning, however, often they are then used for other and different tasks among which also semantic segmentation. In contrast, semantic segmentation learns from examples how to separate or segment classes of objects. In this sense the task is very similar to object class localization.

At first glance, the most relevant difference is that segmentation expects a localization of the object at pixel level, while localization is generally happy with a bounding box. In this way, semantic segmentation appears more strict than localization. However, often (although it is not very standard in literature yet), when speaking about segmentation, the distinction among instances of objects of the same class is relaxed, i.e. two objects close each other for segmentation are represented as a big blob loosing the information that there are two instances objects of the same class. In this sense, localization gives more relevant information about the scene. In Fig. 1.5 (c) we show an example of pixel-level class segmentation.

1.3.7 Localization vs Pose Estimation

Pose estimation is the task of finding the location of the object parts. Often it is associated with localization because, for localizing the object parts, we need the localization of the whole object. Thus, pose estimation implies the object localization, but not the other way.

Recently, state-of-the-art methods for localization [39, 127] showed that localizing object parts helps to increase localization accuracy. However, while in pose estimation the object parts are given as ground truth, in localization object parts are not given, so they are estimated as latent variables. In this sense, object localization with latent parts and pose estimation look very similar, but there are some differences. As in

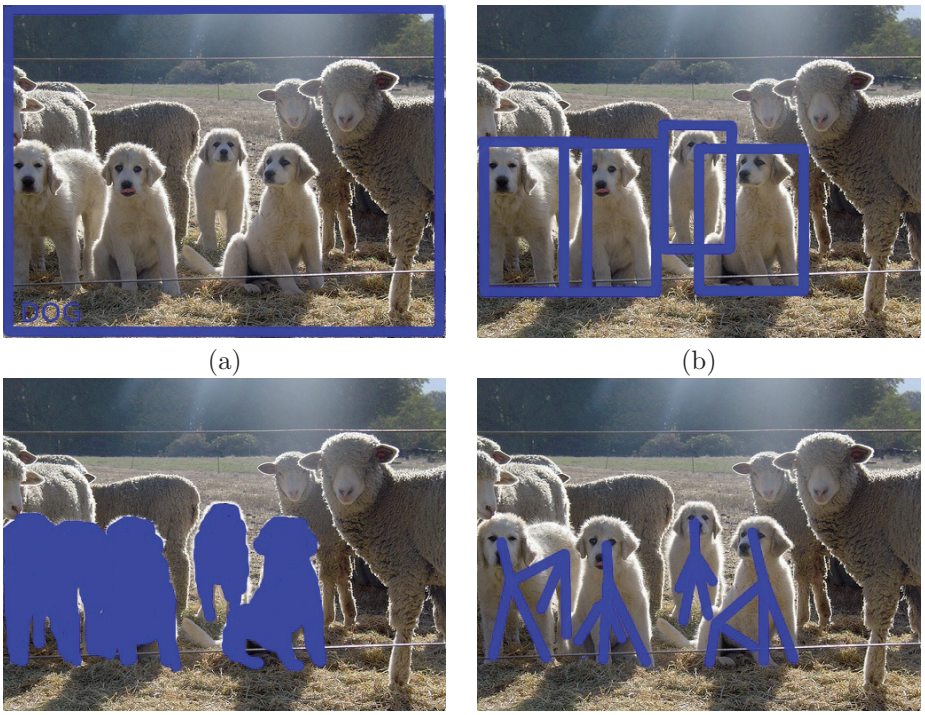


Figure 1.5: Different levels of annotation for the class dog. (a) classification, (b) bounding boxes, (c) semantic segmentation, (d) pose estimation.

localization the object parts are not given a priori, it is necessary an additional task of "parts discovery". This can go from some simple heuristic as splitting the object in a grid of parts, to learning the parts and relative connections (the structure) that best improve localization, but maybe they do not have any semantic meaning. Also the optimization is different for the two tasks. While in localization we aim at reducing the number of examples incorrectly classified, in pose estimation we aim at reducing the localization errors between the estimated pose and the ground truth. Interestingly, in recent works [140] it has been shown that optimizing for the localization task gives better results in both problems. In Fig. 1.5 (d) we show an example of pose estimation.

1.4 What is a good detector

A correct evaluation of an object class localization algorithm is quite challenging but very important to improve the state-of-the-art in the field. Often, a bad evaluation can lead to focus the attention of the research community on bad or not very general methods, spending effort and time in the wrong direction. In this part we consider the

evaluation of a detector at two different levels: detection accuracy and computational cost.

1.4.1 Detection accuracy

The general idea for the evaluation of an algorithm is based on a test set. The test set is a set of images (different form those ones used for training but drawn from the same distribution) together with the corresponding ground truth annotations, that are used to evaluate the algorithm. The evaluation, generally, consists in assigning a score that represents how close the method has performed with respect to the ground truth. In this way, having many different methods, they can be ranked based on that score, and an evaluation of the effectiveness of the method can be given.

An evaluation for human detection is proposed in [24]. Inspired by sliding windows, the authors decompose an image in the set of all possible detection windows. In this way the problem is converted to classification, and standard evaluation methods for classification can be used. In particular, they compare the false positive per window (FPPW) with the miss rate (MR). This method has the advantage to factorize out everything that is not classification to really see which feature-classifier combination is the best preforming. However, during the years it has been shown that this method has many drawbacks. Not considering the full detection framework implies to not evaluate some very important factors (like sampling rate, non maximal suppression, image or feature crop), that can highly distort the final detector performance [39, 30].

Methods that evaluate the full detection system are detection rate versus false positive per image [30] and precision recall [35]. The first evaluates the miss rate of a detector versus the average number of false positive it will produce each image. This gives a good idea of how many errors you should expect for a certain point in the FPPI curve. In contrast, the precision recall curve evaluates a detector in terms of $precision = \frac{TP}{TP+FP}$ (how many detections are correct over all of them) and $recall = \frac{TP}{TP+FN}$ (how many bounding boxes of the ground truth have been correctly detected), where TP is true positive detections, FP is false positive detections and FN is the false negative detections. Using precision recall we can compute the average precision, which resumes the global quality of a detector.

1.4.2 Computational Cost

Many powerful classification techniques are still not applied to object detection because of the higher amount of time they take, especially considering that, for detection, the classifier has to be applied to every possible region of the image. Also, many applications need to run at the speed the camera is providing the images, what is generally defined as real-time. Imagine, for example, a car with a pedestrian detection system to prevent from knocking the pedestrian down. If the pedestrian is not recognized in a few milliseconds, the application is not very useful. In this sense, methods that are able to reduce the computational burden of the object localization search are very

important for research.

The computational cost of object detection is $O(CL)$, where C is the cost for classifying an image region and L is the number of regions to be evaluated. This cost is too high for practical use. Without any constraint, the number of possible regions in an image is $L = 2^P$, where P is the number of pixels of the image, because each pixel can be either part of a region or background. Assuming that the support region of the sough object can be correctly approximated with a connected rectangular region (or box), the number of possible regions is reduced to P^2 because, for each pair of pixels, there is a different box. Considering that, in a normal image, there are millions of pixels, even if the classification cost is very small, this is repeated for each location, which makes the computation still unfeasible.

For this reason, for object detection reducing the computational cost is very important. To this end, it can be reduced either C, using fast classifiers, or L, using strategies to reduce the number of locations to evaluate. In Chapter 2 most of the common techniques for reducing C and L are reviewed.

1.5 Objectives of this thesis

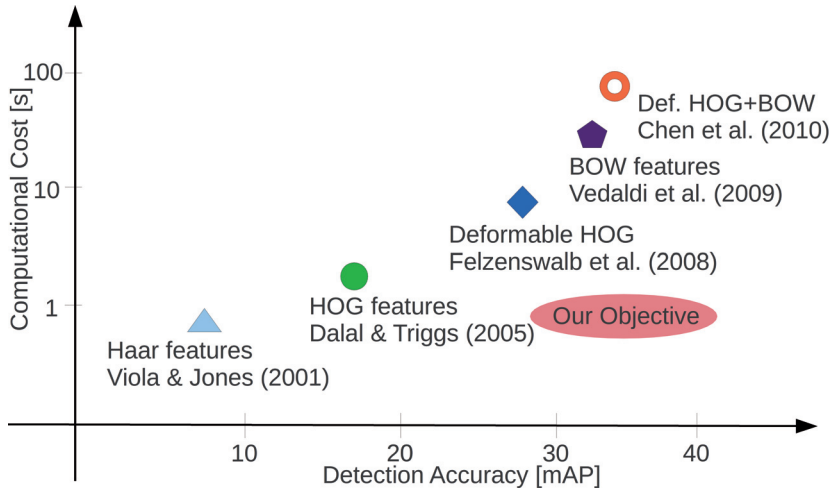


Figure 1.6: Object Detection trend. Newer and more powerful methods for object detection come with an exponential increase of their computational cost (Note that y axis is in log scale) . Our aim is to brake this trend and build a method that is able to achieve state-of-the-art results but far less computation.

As we have seen in the previous sections, a good detector should be the right

trade-off between accuracy and speed. In the last years we have seen a progressive improvement of the accuracy a detector can reach. However, this improvement did not come for free. The computational cost of modern detectors has been continuously increasing in an exponential way. In Fig. 1.6 we plot some of the most relevant methods for object detection (for a complete list go to chapter 2) over detection accuracy, measured as mean average precision on the VOC challenge [35] and estimated computational cost in seconds. Although we can assume that also the computational power of our machines is growing exponentially (Moore's law or, more in general, the law of exponential growth refkurzweil06the-singularity), it is easy to see that this is not enough.

Let's consider, for instance, the famous Viola and Jones detector [130]. In 2001 it was considered the state-of-the-art and it was practically real-time with then available computational power. In contrast, today, state-of-the-art methods like Chen et al. [18] with current computational power are far from being real-time. Thus, our primal objective is to elaborate an algorithm for object detection that can perform similarly to the state-of-the-art, but requiring much lower computation. More specifically we delineate a framework based on multiple resolution that can dramatically speed-up object detection without affecting its accuracy.

We can briefly summarize the objective of our work in terms of the computational costs involved in detection: C the classifier computational cost, L the number of locations where to apply the classifier and F the feature computation. In chapter 3 we introduce our first contribution where we show that, properly using a multi-resolution model, leads to a reduction of not only C, using a cascade as in previous works [146], but also L, changing the sliding window stride accordingly to the model resolution. In chapter 4 we propose a new and specific way to take advantage of multi-resolution in the context of object detection. In this case, both factors C and L are reduced in a way that do not depend on the image content and therefore leads to a fix computational saving. Subsequently, in chapter 5 we show that, in deformable parts models considering limited deformations, the cost for estimating the parts deformation is negligible, while the dominant computation is still in L and C. In this regards, we extend the method proposed in chapter 4 to work with deformable and multi-resolution models, that can lead to enhanced detection accuracy. Finally, in chapter 6 we show that using our deformable multi-resolution model and the previously (4) proposed inference, the bottle-neck is F, the feature computation. To tackle this problem, we propose to use GPU parallel computation to speed-up the feature computation. With this further improvement we finally obtain a *real-time* system with *state-of-the-art* detection accuracy.

1.6 Contributions

Through this thesis we propose several contributions in the field of object detection:

- **Multi-resolution Model**: we show that a multi-resolution model can be used not only for improving the detector accuracy like in [39], but also for speed-

up. In particular, we demonstrate that, taking special attention in the way the model is defined, the same features can be used for both scale search and multi-resolution representation. Also, using multiple resolutions allows the sliding window stride to be proportional to the used resolution, which produces a further speed-up.

- **Coarse-to-Fine search**: we define a new method for fast image scan. The method is based on the fact that for each object in the image it exists a local neighborhood where detection confidence forms a basin of attraction. Thus, instead of the expensive sliding window search a local and fast search over feature resolution can be used. The coarse-to-fine procedure has a similar speed-up as cascade approaches, but in contrast to those ones, it needs a fixed computation time that is independent of image content.

- **Coarse-to-Fine search on deformable models**: we extend the coarse-to-fine framework to deformable models. Considering that the real object deformations are limited, we define a deformable model that has almost the same computational cost as rigid model, but a much better accuracy. Also, we introduce in the model sibling constraints, that can better drive the CF search.

- **Combination of Coarse-to-Fine search with other techniques**: we show that CF search is flexible enough to be combined with other standard methods. We combine it with a cascade of classifiers to obtain a further speed-up as the two techniques are based on orthogonal speed-up cues. Also we show that, adding a final detection refinement based on dynamic programming, can boost performance with a little increment in the computation time.

- **Real application**: we build a real application of the CF inference for the case of pedestrian detection in moving vehicles. We show that using some problem specific restrictions and GPU computation we are able to get state-of-the-art detection accuracy in a real-time application.

1.7 Outline

This thesis is the result of a long process of studying, experimenting and developing new techniques for object detection.

In chapter 2 we first propose a taxonomy to properly arrange the previous work on the different topics that are needed for object detection. From image representation and feature extraction, to optimization and learning. This can help further studies to properly address the focus on the main techniques that are valuable in the filed as well as to give a general overview of object detection.

After that, in chapters 3, 4 and 5 we present a set of incremental approaches for enhancing object detection. The order of the chapters represents the chronological evolution of the work.

In chapter 3 we introduce a cascade of object models at multiple resolutions to speed-up object detection. There we show that the same features can be used for scanning the image at different scales as well as to search for the object at different resolution levels from coarse to fine. This does not produce any additional cost in terms of feature computation, and it provides an image scan up to 20 times faster than normal sliding window.

Considering that, in sliding window, most of the information between two close windows is similar, in chapter 4 we propose a new way to scan an image. By dividing the image into a set of local neighbors where only one object can be found, we can scan the image using a coarse-to-fine procedure that reduces more than 12 times the cost of the scan. In contrast to normal pruning cascades like the one presented in chapter 3, the method is based on the inherent structure of the image, therefore it does not need to learn classifier-dependent thresholds and the speed-up is independent of the image content.

In chapter 5 the coarse-to-fine procedure is further extended to deformable objects. There we show that the computational cost of a part-based deformable object detector is dominated by the cost of evaluating the object parts on the image and not by cost of yielding the best configuration of the parts. We also show that adding local deformations to the coarse-to-fine procedure can highly improve the detector performance while maintaining almost the same computational cost. We modify the latent SVM procedure to work with the approximate coarse-to-fine procedure giving similar speed-ups as in training. Finally, we explain how the proposed coarse-to-fine procedure is orthogonal to pruning cascade procedures and the union of the two methods can finally produce a global speed-up of over two orders of magnitude the standard procedure with a little loss of performance.

Note that the incremental extension of the work over these chapters is not only in terms of improving the detector speed and accuracy, but also in terms of generality. In fact, we start from a human detector tested on the INRIA dataset in chapter 3, then a single aspect but multi-class detector tested on INRIA and VOC 07 in chapter 4 and finally a multi-aspect and multi-class object detector tested on INRIA, VOC 07 and VOC 09 in chapter 5.

Chapter 6 evaluates the coarse-to-fine method on the challenging task of real-time pedestrian detection for moving vehicles. There, it is shown that the method has better performance of previous approaches in both accuracy and time and that, using a GPU-optimized feature computation, it is suitable for real-time performance. Finally conclusions, as well as future lines of research, are discussed in chapter 7.

Chapter 2

Literature review

In this chapter we give an overview of the different methods and techniques generally used for object detection. The methods are organized using the taxonomy illustrated in Fig. 2.1. At top level, object detection is the composition of two tasks: *image classification* and *object localization*. In fact, object detection can be easily thought as the binary classification of all the possible subregions of an image.

Classification is composed of feature extraction, image representation and learning; related works on these topics are dealt in section 2.1. Localization is composed of the type of quantization of the localization procedure and the technique used for speeding up the computation, if any. Related work on this topic is dealt in section 2.2.

After that, in section 2.3 we give a brief overview of the most relevant methods for object detection and sort them using the taxonomy just introduced.

2.1 Image Classification

Image classification is the capability to establish to which class or category an image belongs to. As shown in Fig. 2.1, the task is decomposed into three fundamental parts. The first is *feature extraction*, where the most relevant regions of the image are selected and represented as local features. The second part is *image representation*, where the local features are joint into a single descriptor of a selected region (which can also be the full image). Finally, the third part is *learning*, and corresponds to learn to categorize the set of descriptors generated from the image representation into classes.

2.1.1 Feature Extraction

Although every possible characteristic of an image can be considered a useful feature, generally in computer vision, and specifically in this work, we refer to image feature as local regions or patches which have some distinctiveness. Local regions generally have more stability and invariance than pixels [65]. For this reason, in computer vision many different features based on a well defined region of the image have been proposed and employed.

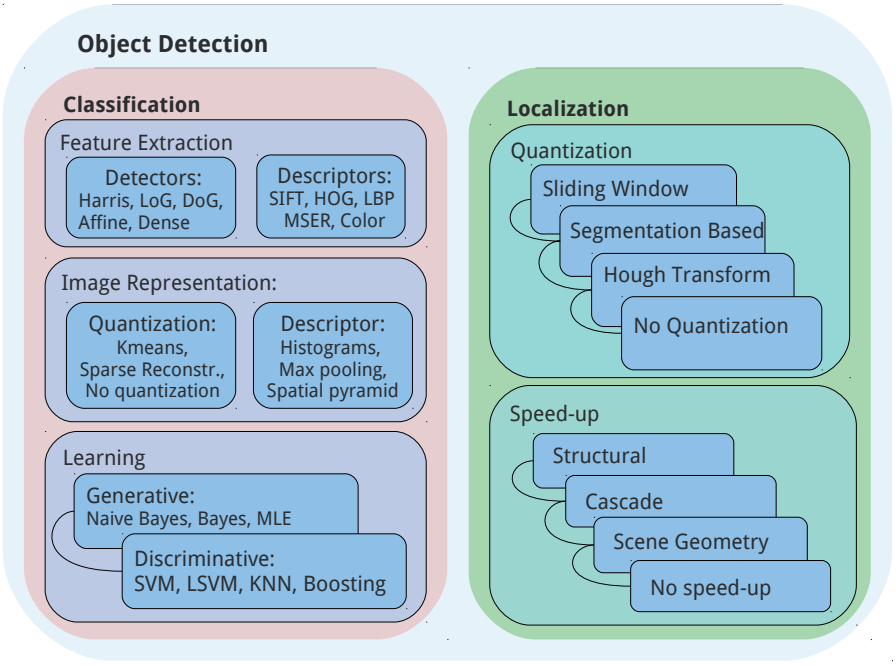


Figure 2.1: Taxonomy of the basic components of an Object Detector. Note that boxes included into super-boxes represent mandatory components, while boxes connected and overlapping each other represent exclusive components. e.g. learning is always present in the classification task, but it can be either generative or discriminative.

The process of feature extraction is therefore divided in two separate tasks, the first is generally named interest point detection, while the second is named feature description.

Detectors Interest point detection is the task of selecting the regions of an image that are distinctive enough to be retrieved under different image transformations. Depending on the application, different detectors have been proposed. In [50], corner are detected evaluating the Hessian of the image. To permit the detection of the same features at different scales, scale-invariant features are introduced. Scale-invariance is generally obtained by searching for extrema of local operators over scale-space images. The most common operators used are the laplacian [64], the difference of gaussians [65], hessian-laplace [72] and saliency [55]. The same procedure is extended to affine-invariant features [120, 73, 56]. A detailed analysis of different feature detectors can be found in [119]. Besides interest points, also edges and segments are used for localizing interesting regions. Edges can be extracted by using the canny operator [16] or by learning a specific boundary detector for natural images [70].

In recent years, due to the increment of computation available, in the context of object classification and detection, often the best strategy is a dense feature extraction

[79]. Dense extraction corresponds to densely select image locations (i.e. every pixel) independently of their content. It will be the following classification stage to assign the importance to the feature. In this regard we consider also HOG features [24] like (histograms of oriented) gradient feature extracted densely.

Descriptors Once a region of the image has been selected, the relevant information contained in it should be stored in a descriptor. The most important feature descriptor is the scale invariant feature descriptor (SIFT) [66, 65]. It is the combination of the difference of gaussians as feature detector and the computation of the descriptor rotated at the dominant gradient orientation. The descriptor is a 4×4 l_2-normalized grid of histogram of oriented gradients at the scale selected by scale-invariant detector. This descriptor is invariant to rotations due to the rotated descriptor, to small spatial movements due to the construction of histograms and to affine light transformation due to the l_2 normalization. More than 10 years later its introduction it is still one of the most used and discriminative descriptors in computer vision. Other similar detectors-descriptors algorithms have been proposed to make the feature extraction faster [48, 4, 114, 15] Also other algorithms improve the descriptor distinctiveness [74] or the descriptor invariance to image deformations [75]. A dense representation of the image is given by the histogram of oriented gradients (HOG) [24] and its extension (ext. HOG) [40], which are very similar to the SIFT descriptor but computed densely and without rotation invariance. A global description of an image is given by GIST [81], which is based on a PCA compression of a set of basic edges and color features.

A different approach is to use a feature descriptor based on boundaries (BFM) [82] or adjacent segments [41]. These descriptors show good performance for shape-based detection. Also descriptors based on self-similarities (SS) [103] show good invariance capabilities for object detection when dealing with highly textured objects.

Features based on colors are also used in the context of image classification and object detection. An overview of SIFT descriptors extended to different color spaces are evaluated in [95].

Invariance A lot of effort has been spent for making the feature as much invariant as possible to all kind of photometric and geometric properties, like illumination, color, local deformation, rotation. This process involves both the detection and the description part. For instance, in the SIFT algorithm the detector is invariant to rotations, in the sense that it is anisotropic, i.e it does not privilege any image orientation, while the descriptor is rotation covariant in the sense that it is computed on the main orientation of the local patch. The combination of these two steps makes the SIFT algorithm rotation invariant.

In the last years it has been shown that, often, a total invariance to a certain image property can have a negative impact in the task of classification [145]. In fact, often, it is much discriminative to know the degree of variance of a feature. For example, back to the case of SIFT, it has been seen that, adding to the commonly used SIFT descriptor a part that represents the rotation of the patch, can be useful to improve its discriminative capability.

2.1.2 Image representation

Once features have been extracted from an image, they have to be joint in a single, general enough description. In this sense, many different techniques have been proposed, and almost all of them can be further described in two subtasks: feature quantization and final description.

Quantization The local patch descriptors presented in the previous section are generally laying in a manifold, those standard statistical analysis cannot be applied. To deal with this, the space of the patch descriptor is divided in local regions. If the local region is small enough, the characteristics of all the descriptors in that region are similar, thus they can be considered a constant. In this way, we can associate many different locations of the manifold to different elements of a set and, subsequently, use them with standard tools.

A common way for the quantization of local features is using clustering, that is an unsupervised grouping of elements which have some common features. In literature there are many different algorithms for clustering depending on the criteria for grouping and the optimization strategy [138].

The most simple but effective is kmeans [68]. Given a set of examples $\mathcal{X} = (x_1, x_2, .., x_N)$, kmeans minimizes the following objective function:

$$\min_{V} \sum_{i=1}^{N} \min_{u_i} |x_i - u_i V|, \quad s.t.\ |u_i|_0 = 1\ ,\ |u_i|_1 = 1, \tag{2.1}$$

where V is a matrix containing in each raw the center of a cluster and u_i is an indicator vector that indicates which cluster the associated element x_i belongs to. This can be seen as the minimization between the real image and its reconstructed version using the clusters as a set of bases. As the cluster centers are latent variables, the problem is not convex and the optimization is generally performed using expectation maximization [92]. Recently, a sparse coding representation has shown better performance than standard kmeans [139, 135, 3]. In this case the algorithm has the same objective function, but it relaxes the condition on u_i, so that a feature sample can be represented as a linear combination of clusters, therefore reducing the reconstruction error.

One drawback of quantization is its computational cost. While the learning of the clusters, even if slow, it is performed only once, the assignment of a new point to the correct cluster is often the bottleneck of many methods based on features quantization. Assignment consists on finding the closest cluster to each point which is a nearest neighbor problem [21]. In this sense, the clusters V are given and u_i should be found:

$$\min_{u_i} |x_i - u_i V|, \quad s.t.\ |u_i|_0 = 1\ ,\ |u_i|_1 = 1. \tag{2.2}$$

From Eq. 2.2 we can see that the computational cost is proportional to the number of cluster $O(N)$; logically, up to a certain point, also the quality of the quantization is proportional to the number of clusters. Generally, for classification between hundreds and thousands clusters are used, which makes the assignment very slow. To tackle this problem many approximation techniques have been proposed [104, 76].

Also, uniform approximation of the feature space has been proposed [111]. In this case, to obtain a reasonable representation, the number of clusters grows to millions. However, in this case the assignment has a cost $O(1)$ and hash tables can be used. For other kind of features that are not local like GIST [81] and HOG [24], no quantization is performed.

Descriptor Describing an image or a region of an image corresponds to find a way to summarize the information contained in the extracted features. Depending on the properties and invariance we want to apply, different descriptors have been proposed in literature. The most used descriptor is the normalized histogram of the clusters occurrence. The association of a local feature quantization and the histogram computation is generally known as bag of words technique because it was initially used for text classification. In bag of words descriptor is a vector with dimensionality equal to the number of clusters used in the quantization procedure. Each feature sum a point to the clusters it belongs to. Then, the histogram is l_2 normalized. Finally, every bin of the descriptor contains the average number of times a feature has been found in the image:

$$\frac{\sum_i u_i}{|\sum_i u_i|_2}. \tag{2.3}$$

In this sense, the procedure used to build this histogram is known as average-pooling.

The computational bottleneck of the BOW representation is the need of non-linear kernels for best performance. For linear kernel the evaluation of a test image is very fast because it is just the scalar product of the BOW histogram with the corresponding learned weights. Thus, its computational cost is proportional to k, the vocabulary size. In contrast, in non-linear kernel spaces, the evaluation of an image using the representer theorem [22, 102] is the weighted sum of all support vectors, which in general is on the order of the training examples n. Therefore, the computational cost is now kn which can be thousands times slower than for the linear case. In the last years, for certain kernels, explicit approximations of their kernel have been found [69, 126], which allows to convert the non-linear learning to a linear one.

Another way to use linear classifiers with BOW is sparse coding. In association with sparse coding has been empirically found that, substituting the average-pooling with the max-pooling, increases the final classifier performance [139]. In practice this correspond to changing the sum of Eq. 2.4 with the max operator:

$$\frac{\max_i u_i}{|\max_i u_i|_2}. \tag{2.4}$$

Surprisingly, this configuration associated with linear kernel can reach similar or better performance than the typical average-pooling with intersection kernel, which is computationally much more expensive. The intuition behind bag of words methods is that they represent the most interesting local elements of the image discarding the global scene geometry. In some cases, where the global scene geometry is very poor, or very complex, this is still the best representation. However, in other cases, it is useful to consider global information.

The most successful extension of the bag of words model that takes into account the global geometry of the scene is the bag of words pyramid [49, 60, 8]. It consists on considering the descriptor as a pyramid of occurrence histograms with different

levels of resolutions. Starting from level 0, that is a classical bag of words, then level 1 divides the image in a regular grid of 2×2 parts, and for each one, a histogram of bag of words is computed. The procedure can continue to the level r, where the regular grid would have a resolution of $r \times r$. This approach, although very simple, can boost classification accuracy, especially for categories with regular global shape. The main drawback of the pyramid is the memory occupancy, which is $k \sum r^2$, where k is the number of clusters (vocabulary size) used. For a commonly used pyramid of 3 levels, for instance, the memory consumption is 14 times the bag of words model. In this sense, several technique have been proposed to handle this problem [44, 67].

Finally, in case features are not quantized, like in HOG, the image descriptor is generally computed as the concatenation of the local features.

2.1.3 Learning

Given a set of training samples and a predefined mode, learning is the task of finding the most appropriate parameters of the model that best performs on a defined criteria. More specifically, we are most interested in learning binary classifiers that are able to best discriminate or separate negative and positive examples (i.e in an image distinguish a car form other objects). A model is generally represented by a family of functions, where a set of parameters establishes which instance fit best to the training data. The most simple model is a linear function. In spite of the simplicity, when dealing with high dimensional spaces, linear functions are still competitive with more complex models. This is because, often, using a too complex model leads to overfitting the data.

In the following we discern two main families of models: generative models and discriminative models [78]. Generative models are models that encode the entire complexity of the process that generates the training data. In contrast to generative models, discriminative models want to approximate a function that best distinguishes between positive and negative examples without caring about the real process that generates the data. The general idea that has emerged in the last years in the computer vision community is that generative models are more flexible and can easily be adapted to different tasks, while discriminative models are specific for the discriminative task, but generally they perform better [78].

Generative Models Generative models are generally based on *Bayes* rule:

$$P(Y|X) = P(X|Y)P(Y), \tag{2.5}$$

where X is a random variable representing the input data and Y is a random variable representing the output, $-1, 1$ in case of binary classifiers. $P(X|Y)$ is often referred as likelihood, because it represents the distribution of the data X given the output Y. $P(Y)$ is the prior probability of a certain input. If we consider $P(Y)$ uniform, then we need to estimate only $P(X|Y)$ and this is referred as *maximum likelihood estimation* (MLE). Estimating $P(X|Y)$ is often impractical because it would require a high number of samples, especially if X has high dimensionality. An alternative approach is considering all dimensions of X like conditionally independent. In this case we use what is normally called a *Naive Bayes* model [31].

Discriminative Models In discriminative models, the focus is directly on $P(Y|X)$. K nearest-neighbors [21, 14, 7] is a very powerful discriminative classifier. It consists on classifying a new example based on the K closest in the training data. Although its simplicity, it can generate state-of-the-art results when enough training data is given. On test, its computational time is proportional to the number of training samples. As for the assignment in the bag of words model, also in this case techniques for speeding-up the neighbor search have been developed [104, 76].

Often, in discriminative models, it is defined an energy or *objective function* that optimizes explicitly the errors the model produces on training data [61]. In this way we define a loss function that encodes the way errors are penalized in the objective function. For classification the typical loss to optimize is the zero-one loss, which gives a penalty of 1 for each example misclassified and 0 for the right classification. Unfortunately, optimizing the zero-one loss is NP hard, so surrogate losses are substituted to the real loss to make the optimization feasible.

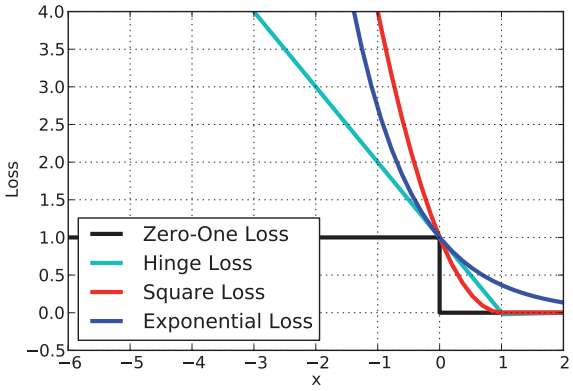


Figure 2.2: Example of different loss functions.

We can group the methods depending on the objective function. In vision, a common technique to build discriminative classifiers is based on boosting [98]. The optimization procedure is based on an iterative procedure that at each iteration refines the classifier based on the errors committed on the previous iteration. More specifically, the classifier f is a linear combination of so called weak classifiers. At the beginning, a set of weak classifiers are learned to minimize the error in the training images using the exponential loss:

$$\sum_{i=1}^{N} \exp(-y_i f(x_i)). \tag{2.6}$$

The importance of each training example is weighted based on the loss produced and new weak classifiers are added with the re-weighted training set. The iterative procedure is repeated for several iterations.

Another very popular technique for learning a model in image classification is support vector machines (SVM) [123, 22]. The classifier is a linear model on the

feature space $f(x) = \langle w, x \rangle$, while the loss function is the surrogate hinge loss.The model parameters w are learned minimizing the following objective function:

$$\frac{1}{2}|w|^2 + C\sum_{i=1}^{N} \max(0, 1 - y_i f(x_i)). \tag{2.7}$$

The first term of Eq. 2.7 is the regularization, while the second is the hinge loss generated by each example. The trade-off between regularization and loss is controlled by C, which is generally estimated through cross-validation. The objective function is convex in w, so its minimization leads to the global and unique solution. For more sophisticated spaces, the linear classifier f can be substituted to a non linear one using the kernel-trick [22]. Eq. 2.7 can be optimized with different strategies [101, 11].

Recently, binary SVMs have been generalized to structured output SVMs [112, 117]. In this case, the output can be any structure like a parsing tree, a string or, in general, any kind of graph. The main difference with normal SVMs is the scoring function $f(X, Y)$ that now is also function of the structured output Y. The new objective function is:

$$\frac{1}{2}|w|^2 + C\sum_{i=1}^{N} \Delta(y, y_i), \tag{2.8}$$

where Δ is the loss function that measures how close is a certain solution y to the ground truth y_i. The loss Δ can have any shape which would produce a NP hard problem. To still have a convex problem, as for the binary case, we create a surrogate loss $\hat{\Delta}$ that bounds the real loss in this way:

$$\Delta(\hat{y}, y_i) = f(y_i, x_i) - \Delta(y, y_i) + f(y, x_i), \tag{2.9}$$

so that the margin between a given output y and the ground truth is equal to the loss (margin rescaling). In contrast to binary SVM, in this case, each example can generate multiple constraints (each for each possible configuration of y). For solving this is generally employed cutting-plane [54].

Recently, structured output SVMs have been used for object detection, considering the bounding box of the sough object the SVM output [5, 125].

In the context of object detection, latent SVM has shown excellent results [39, 141]. Latent SVM is the extension of SVM to consider latent variables, i.e. variables that are unknown in training and they should be estimated to enhance the final classifier accuracy. For instance, in [133, 39] it is shown that a good alignment between image and object model is fundamental for good performance. In this regard, assuming the object location, or even local object parts, as latent variables, improves the image alignment and, therefore, the final detector accuracy. Although the benefit in terms of performance, considering latent variables in the learning procedure makes the object function not convex. However, in [141] it has been shown that the new objective is the composition of a concave and a convex function, therefore the CCCP [143] can be used. In this way, if the initialization of the latent variables is good enough, the final solution will be close to the optimal.

Finally, a quite recent technique for classification is random forest [12]. A random forest is a combination of tree predictors, such that each tree depends on the values

of a random vector sampled independently and with the same distribution for all trees in the forest. The main advantage of this method is that each tree predictor selects only a small amount of the input features, which makes it suitable for real-time applications [36, 142].

2.2 Object Localization

In this context, with object localization we intend the task of localizing the searched object given a region classification algorithm. We separate the set of possible algorithms based on two properties: quantization and speed-up.

2.2.1 Region Quantization

Quantization represents the way the regions of an image are selected. Without any approximation there is a combinatorial number of possible regions to be selected. Generally to reduce the complexity of the problem, for detection only rectangular regions are considered. Furthermore, to make the algorithm computationally tractable, among all possible locations, aspects and scales of every rectangular region, only quantized subsamples are considered. Depending on the strategy of quantization different algorithms have been proposed with different advantages and disadvantages.

Sliding window Sliding window is the simplest but effective technique for scanning an image and it is currently used in the most successful methods for object detection and localization [24, 121, 39, 124]. It consists on a uniform sampling of the detection window over scale and position. The sampling step over spatial locations and scale determines the trade-off between speed and accuracy. Note that, if it is necessary to detect objects with different bounding box aspects, a sampling over the window aspect is also needed.

Segmentation Based Recently, it has been shown that, just looking at some image properties like boundaries and colors, can give a good estimation of the regions where an object is probable to be. Alexe et al. [1] propose to distinguish generic objects from background. Using features based on characteristics of objects, such as appearing different from their surroundings or having a closed boundary, it can highly reduce the number of bounding box hypotheses to consider. In contrast to a normal detector, the aim of this algorithm is to select the minimum number of hypotheses that includes the maximum number of objects in the image.

The same aim is also pursued by Van de Sande et al. [122]. Initial bounding boxes are generated from segmentation as the minimum ones that contain a segment. Then, different bounding boxes are fused in a greedy fashion, based on an appearance distance. The set of initial bounding boxes plus the ones generated by their fusion are the hypotheses to use for applying a class specific detector. Due to the fact that the segmentation is already assuming some priors about the object boundaries, with a relatively limited number of bounding boxes the algorithm can obtain quite high recall.

Hough Transform Hough transform is a popular technique for finding primitives in images like lines and circles [32]. The main idea of the method is to collect the number of occurrences the sought primitive over the primitive parameters. In case of a

line for instance, the hough space is composed by line orientation and line intersection with the x axis. A slightly different version of this have been applied for object detection. In this case the hough space is composed by position and scale of the center of the object bounding box (assuming to deal with a single aspect ratio object). During training the correspondence of a visual word to the object center is learned. Then in testing, every new feature is associated with the closest visual word and a vote on the possible object center is added. The locations with more votes are the final detections. The advantage of this method is that is not necessary to really evaluate the score of the detector on each bounding box. However, it has been noticed that the hough space is quite noise because a single hypotheses can collect votes from different objects. Thus, generally the hough procedure is finally improved with a final refinement often based on standard SVM classification. Examples of hough based detectors are [63, 80, 45].

Similarly to hough transform, the so called jumping window [124, 127] has revealed a useful strategy to sample image windows for detection. As in hough transform detection hypotheses are generated from visual words: the words with best discriminative power and stable location in the bonding boxes of the training samples are used to generate detection hypotheses. In [124] it is shown that with a reduced set of hypotheses jumping windows can retrieve more than 90% of the objects in the dataset.

2.2.2 Speed-up techniques

To further speed-up the localization process other techniques different than region quantization have been proposed and are analyzed in this section. We can roughly separate these techniques into *structural*, *filter-based* and *scene geometry based*.

Structural Structural speed-ups are those techniques that take advantage of the image structure to make the image scan faster. In many cases the quantization technique inherently introduces this kind of speed-up. For example, the trivial sliding window considers that close-by detection regions have an high overlapping area. Thus, most of the features found in one region would be repeated in the other region, therefore it is not necessary to really evaluate the regions at pixel level; a small stride can high speed-up the image scan without affecting the detection rate.

Another technique that takes advantage of the image structure is the efficient sub-window search [58]. In this case, bounding boxes are parametrized as intervals of the top-left and bottom-right coordinates. For each interval, using the bag of words model, a bound of the minimum and maximum classification score is estimated. In this way, using a branch-and-bound algorithm only the intervals with highest bound are further decomposed until reaching an interval containing only one element, which is the bounding box with highest score. Using this method it is not necessary to evaluate every bounding box in the image because the bound already gives an estimation of the minimum and the maximum score in the interval. A similar strategy, but based on regions extracted form a segmentation of the image is presented in [128].

We can also classify in this category the coarse-to-fine strategy proposed in chapter 4 and 5 of this thesis. In this case, the sliding window procedure is initially computed at a coarse level. From this, knowing that, for a small enough region, only one object (chapter 4) or a part of it (chapter 5) can lay there, an hypothesis per region is

generated and locally propagated over finer levels of representation.

Filtering Cascade-based techniques are based on the fact that, decomposing an image into the set of regions to evaluate, the number of regions that contain the searched object are fewer than the number of regions where the object is not present. This asymmetry can be exploited substituting the single region classifier with a hierarchy of classifiers. If the hierarchy is sorted, from fast but inaccurate classifier to slow but accurate classifiers, there exists a set of pruning thresholds that can highly speed-up the computational cost of the image scan without any loss in accuracy. In practice, in contrast to the normal classification method, the time spent on a certain region depends on its difficulty. If the region is classified as no object with high accuracy from the first level of the hierarchy, the classification is already finished with a high computational saving; otherwise, when the region is uncertainly classified, the final decision is passed to the next and more accurate classifier. The procedure is repeated until discarding the region (no object) or reaching the last and most accurate classifier that will finally decide whether the region represents the searched object or not.

Often, the cascade of classifiers is obtained from boosting where, at each stage of the cascade, a more accurate classifier is generated adding more and more weak classifiers [131, 33, 106, 9, 149]. Recently, cascade of classifiers are also employed in SVM-based classification. For instance, in [146] as well as in chapter 3 and 5 of this thesis, a cascade is build on model representations associated to feature at different resolutions from coarse-to-fine. In [51] and [124], instead, a cascade of classifier is built changing the SVM kernel, from the fast linear one to the slower, but more accurate, intersection and RBF kernels. Finally, in [37] a cascade is built on the object structure. That is, by using an object model composed of different parts, each part represents a stage of the cascade.

Scene Geometry This technique consists on (sometimes partially) inferring 3D or semantic structure of the scene and using it to speed-up the search focusing the image scan only to those regions that are likely to contain the object. For instance, if we can estimate that a certain region of the image represents the sky, it is not necessary to search there for cars. Also, if we can estimate that a certain region of the image contains an object with a certain volume, the same region could not contain any other volumetric object.

In the automotive field it is quite common the estimation of the ground-plane for reducing the object search. This can be estimated off-line and then fixed with a certain margin due to possible changes in the camera pitch [46], or estimated on-line. For instance, in [47] the ground-plane is estimated taking advantage of stereo images, while in [144] the ground plane is obtained by estimating the horizon line.

Notice that reasoning about the scene geometry is not only useful for saving time, but also for improving the detector accuracy. In fact, selecting a limited amount of regions where the object can be and assuming that these regions are correctly selected (no object is there), can help to avoid false positive detections. Examples can be found in [53, 84, 25, 110].

2.3 Methods for object detection

In table 2.1 we list the most relevant methods for object detection of the last years. Each method is described using the fields of the taxonomy previously presented.

Object detection on static images started in the seventies. Fishler at al. [42] introduced the pictorial structure model that proposes to detect an object as a set of independently-learned parts that have spring-like geometrical constraints. Currently, most of the state-of-the-art methods are still based on that deformable model [39, 40, 18].

In the 1990s and early 2000 object detection mostly focused on face detection mainly because faces, in contrast to other objects have a quite stable appearance that can be easily recognized. Methods based on the PCA decomposition showed some discriminative power [118], although the break-through happened with the famous Viola and Jones work [132], where the authors showed that a cascade of Haar feature classifiers is able to reach real-time performance and quite accurate detection for faces.

Unfortunately, the same method for pedestrians does not work very well due to the highest degree of variability that a pedestrian can have depending on the clothes and the pose he is assuming. In 2005 Dalal and Triggs [24], renouncing to real-time performance, obtain much better detection accuracy for pedestrians using HOG features and SVM learning. Subsequently, several improvements of the original detector have been proposed [149, 146, 51, 136, 100].

A further evolution in the filed of object detection is due to the introduction of the PASCAL VOC [35], that every year proposes an object detection challenge, where the best methods are applied on a large dataset of 20 different object categories. As the test data annotation is not available and the number of allowed evaluations is limited, data can not be overfitted, so we can consider the VOC a fair comparison of state-of-the-art techniques for object detection.

From VOC 2007 to current days, the most promising technique in terms of both speed and accuracy is the deformable parts model proposed by Felzenszwalb et al. [39, 40]. This method is based on moving parts that allow the model to best align with the current image, which gives a boost in the HOG discriminative capability. Also the method take advantage of the distance transform to reduce the cost of searching for the parts location. As this method, together with [24] is the starting point of most of this work, most of its parts will be explained in detail in the following chapters.

The other method that proved to be competitive in the VOC challenge is the bag of words representation already detailed at the beginning of this chapter. Bag of words representation discards geometrical information; it was initially and successfully used for classifying the entire image where no concrete shape was present [105, 23, 59, 145]. Later on, Vedaldi et al. [124] proved its validity on detection, when used in a relatively fine pyramid representation and with kernel SVM learning. Since then, it has been noticed that HOG representation is better for shape defined classes, like bicycles, cars, humans, horses, etc.., while BOW gives better performance on highly deformable classes, like cats and dogs. Similar results were also obtained by [58] using the ESS search and BOW.

An important step forward for enhancing detection accuracy is the work of Harzallah et al. [51], where the authors in one side combined HOG and BOW features, and

in the other they showed that classification and detection are highly correlated tasks, and they can help each other. Since 2010, all VOC best methods are based at least on the combination of deformable HOG, BOW and classification.

			Classification								
			Feature Extraction		Image Representation		Learning	Localization			
Method	Year	Ref.	Det.	Descr.	Quant.	Descr.		Quant.	Speed-up	Other
Haar	2004	[132]	Dense	Haar	None	Concat.	Adaboost	SW	Cascade	Real-time
HOG	2005	[24]	Dense	HOG	None	Concat.	lin. SVM	SW	None	Bootstrapping
Fast HOG	2006	[149]	Dense	fast HOG	None	Concat.	Boost of SVMs	SW	Cascade	HOG int. image
Hough MC	2006	[71]	Har-Lap	SIFT	Hier. Clust.	Sum	MLE	Hough	Struct.	
R. Manifold	2006	[121]	Dense	Covariance	None	Covar.	Logitboost	SW	None	
BFM	2006	[82]	Edges	BFM	None	BFM	Adabost	SW	None	
kAS	2006	[41]	Edges	kAS	kmeans	pyramid	SVM	SW	None	
Multires.	2007	[146]	Dense	HOG	None	Concat.	lin. SVM	SW	Cascade	
HOG+BOW	2008	[51]	Dense	BOW+HOG	kmeans	pyramid	lin. SVM	SW	Cascade	+classification
Hough	2008	[63]	Harris	SIFT	Hier. Clust.	Sum	MLE	Hough	Struct.	
Def. Parts	2008	[39]	Dense	ext. HOG	None	concat.	LSVM	SW	None	Distance Transf.
ESS	2008	[58]	Harris	SIFT	kmeans	histogram	SVM	None	Struct.	branch-bound
HOG-LBP	2009	[136]	Dense	HOG+LBP	None	Concat.	lin. SVM	SW	None	Occlusions
PLS	2009	[100]	Dense	Color+HOG	None	Concat.	PLS+SVM	SW	None	
Geometry	2009	[47]	Dense	fast HOG	None	Concat.	lin. SVM	SW	Geometry	
Multires.	2009	Chap 3	Dense	HOG	None	Concat.	lin. SVM	SW	Cascade	
struct ESS	2009	[5]	Harris	SIFT	kmeans	histogram	str. SVM	None	Struct.	branch-bound
BOW	2009	[124]	Dense	PHOG,SIFT,SS	kmeans	pyramid	RBFχ^2 SVM	SW	Cascade	MKL
3D model	2009	[108]	Dense	SIFT	kmeans	histogram‘	SVM	SW	None	3D models
struct HOG	2009	[125]	Dense	HOG	None	Concat.	lin. SVM	SW	None	Def.+Occl.
Hier. struct	2009	[99]	Dense	PHOG+BOW+SS	None	pyramid	???	SW	None	CRF structure
CF	2010	Chap 4	Dense	ext. HOG	None	Concat.	LSVM	SW	Struct.	Coarse-to-Fine
New features	2010	[134]	Dense	HOG+SS+Col.	None	Concat.	inter. SVM	SW	None	
ESS Cascade	2010	[57]	Harris	SIFT	kmeans	histogram	kern. SVM	None	Str.+Casc.	branch-bound
Mixt. Parts	2010	[40]	Dense	ext. HOG	None	Concat.	LSVM	SW	None	Mix. Aspects
FDW	2010	[27]	Dense	Grad+Color	None	Concat.	Boosting	SW	None	integral image
Casc. Parts	2010	[37]	Dense	ext. HOG	None	Concat.	LSVM	SW	Cascade	
Rot. Ferms	2010	[129]	Dense	HOG	None	Ferms	Ferms	Hough	Cascade	
CF+Parts	2011	Chap 5	Dense	ext. HOG	None	Concat.	LSVM	SW	Str.+Casc.	Coarse-to-Fine
Segment	2011	[122]	Dense	col. SIFTs	kmeans	pyramid	inter. SVM	Segment.	Struct.	
Sparse	2011	[127]	Dense	SIFTs	sparse coding	max pooling	lin. SVM	Segment.	Struct.	Fixed Parts
BOW+Parts	2011	[18]	Dense	HOG+SIFT	kmeans	pyramid	inter. SVM	SW	None	Active masks

Table 2.1

Recent methods for object detection. The principal characteristics of recent methods for object detection are listed based on the taxonomy presented at the beginning of this chapter.

Chapter 3

Multiresolution Cascade

In this chapter we propose the first approximation of our model. We present a human detector based on a multiresolution cascade. The algorithm is based on a early rejection of negative hypotheses using coarser representation of the object. Going down in the cascade, the number of hypotheses is reduced and the model resolution increases. In this way, at the last and more expensive stage of the cascade, only few hypotheses are evaluated. Compared with boosting-based cascades, the use of an SVM-based cascade using multiple object resolutions has several advantages: (i) no new features need to be computed during the cascade traverse, as the same features are used for the search at multiple scales as well as in the cascade, (ii) the search speed-up is produced by the reduced cost of the classification due to the coarse object representation as well as by the high stride of the sliding window, which is chosen proportional to the model resolution. Finally, depending on the setting of the rejection thresholds, the detector, compared with the sliding windows, can achieve a speed-up on the object search between 10 and 20 times with a marginal loss in accuracy.

3.1 Introduction

Within object class detection, human detection is very challenging, since it is one of the most difficult classes. This is due to the fact that, differently than many other categories, humans are not rigid bodies and, furthermore, they can wear different kinds of clothes with varying shapes, dimensions and colors. This implies that humans have a very high intra-class visual variation that actually makes their detection an even more difficult problem. Restricting the problem to standing people (but still observed from all possible directions: frontal, side, backward) makes it possible to tackle it.

Considering that in sliding window approaches most of the evaluated windows can be easily recognized as negative examples (i.e. non-textured parts like sky or walls), the use of a system that can calibrate its computational time based on the difficulty of the samples can highly speed-up the full process. In our method we propose a

cascade of sliding window detectors. Each detector is a linear model (trained using SVM) composed of HOG features at different resolutions, from coarse for the first level to fine for the last one. Using this representation, most of the object location can be discarded by the first level of the cascade, which is coarse and therefore very fast to be evaluated. The filtering is repeated until the last and finest resolution level of the model. Here, the evaluation is more expensive but also more discriminative because it is effectuated on the finest resolution. However, only a reduced set of hypotheses reaches the last stage. In this way the final detection cost is much inferior than evaluating directly the fine object representation everywhere in the image.

Also, unlike previous methods based on Adaboost cascades, we adapt the sliding window stride to the features resolution: higher the resolution, smaller the spatial stride. This reflects that the speed-up of the cascade is not only due to the low number of features that need to be computed in the first levels, but also to the lower number of detection windows that needs to be evaluated.

The rest of the chapter is divided into the following parts: section 2 is dedicated to the concept of multiresolution cascade highlighting its advantages. Section 3 and 4 explain training and detection procedures used in the experiments. A comparison of the performance of the detector in different configurations is presented in section 5. Finally, section 6 is a final discussion about the method.

3.2 Overview of the method

In Adaboost based methods the trade-off between speed and performance is accomplished by adding at each stage new weak classifiers. In contrast, in our model, the use of a cascade of SVMs entails many different options to balance speed and accuracy. A possible way is to use different kernels, starting from the fastest linear one up to the slowest Gaussian one. A similar work, based on histogram of word features has been presented in [124]. However, as already shown in [24], in the case of HOG features, the use of a non linear kernel does not improve the results very much but it makes the computation tremendously slow.

Another possibility would be the selection of a small subset of features in the first level of the cascade, and then add more and more features for the following levels, until all relevant features are considered. This solution has two problems. First, there is not a clear way of selecting features. Second, by selecting sparse features we lose the global and dense representation of the object, which can be useful in many circumstances (e.g. detection of occlusions).

3.2.1 Multiresolution cascade

Our method represents the object that we aim to detect by using several feature resolutions: from few big features which represent the whole object in a coarse way, to many small features, where each one represents a small portion of the object in a

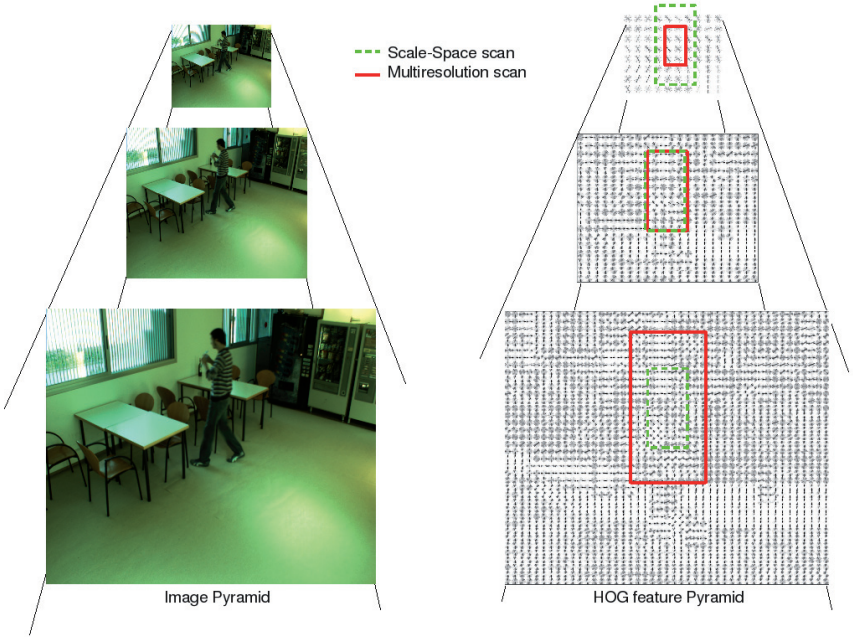


Figure 3.1: HOG feature pyramid. It is used for both Scale-space search (green dashed detection window) and Multiresolution Cascade (red continuous detection window). The scale-space search uses a fixed size window, while the multiresolution cascade doubles the size at each level.

more detailed way.

In contrast to previous methods, where the concept of cascade is associated to Adaboost classifiers, in this work we propose a cascade of SVM, where for each level a different feature resolution is used, from coarse for the first levels to fine for the last ones. The fact that no feature selection has been applied in the cascade implies three important consequences: (i) the feature size of every level is known and this is used to decide the sliding window stride: in this way in the first level it is possible to use a high sliding window stride which reduces the number of window to scan, while in the last level a small sliding window stride is used which produces better localization; (ii) the training time is highly reduced (from days to hours in a standard PC) because the expensive process of feature selection is substituted by a faster linear SVM training; (iii) features always keep a dense distribution which can be used for additional reasoning, like observing the feature response distribution looking for possible partial occlusions or also neighborhood coherence.

3.2.2 HOG pyramid

Feature computation is pre-calculated for each scale s, resulting in a pyramid of features H_s, as represented in Fig. 3.1. In practice, the original image is subsampled by using bilinear interpolation and a dense grid of features is extracted. This is repeated for all levels of the pyramid. The scale sampling of the pyramid is established by a parameter λ defining the number of levels in an octave, that is the number of levels we need to go down in the pyramid to get twice the feature resolution of the previous one.

The pyramid is used for scanning the image at different scales, as well as for the different resolutions of the cascade. If we move across the pyramid levels maintaining the same number of features per detection window, we move over scale; if we move across the pyramid varying the number of features per detection window, we move over resolution.

In contrast to [146], where each feature resolution level needs to be calculated as a supplementary step, we use the same features for both scale-search and multiresolution cascade. If the multiresolution levels and the sliding windows scale-search use the same scaling stride or even a multiple one, it is possible to adopt the same features for both processes. This means a high save of computational time considering that feature computation is one of the most time-expensive tasks in the object detection pipeline.

The basic block of the pyramid is the HOG feature which has reveled very effective for object class detection tasks (see [24], [39]). The computation of HOG is the following. First, for each sub-sampled image $I(x, y)$ at a certain scale s, gradient magnitude m and orientation θ are computed as follows:

$$m(x,y) = \sqrt{(I(x+1,y)-I(x-1,y))^2+(I(x,y+1)-I(x,y-1))^2}, \tag{3.1}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{I(x,y+1)-I(x,y-1)}{I(x+1,y)-I(x-1,y)}\right). \tag{3.2}$$

After that, a weighted histogram of orientations is computed for a certain square region called cell. The histogram is computed by summing up the orientation of each pixel of the cell weighted by its gradient magnitude. The histogram of each cell is smoothed using trilinear interpolation, in space and orientation. In our implementation we use a cell size of 8×8 pixels and an orientation bin of 20 degrees obtaining a cell descriptor of $360/20 = 18$ dimensions. Finally the cells are associated into blocks of four adjacent cells and normalized using L_2 norm obtaining a total of $18 \times 4 = 72$ dimensions.

The use of orientation histograms over image gradients allows us to capture local contour information, that is the most characteristic information of a local shape. Translations and rotations do not influence HOGs as long as they are smaller than the local spatial and orientation bin size, respectively. For this reason the use of

different HOG resolutions helps to better represent an object. Object parts that highly move like human legs are best represented by low resolution features, while object parts with a more stable representation are best represented by high resolution HOGs. Finally, local contrast normalization makes the descriptor invariant to affine illumination changes which improves detections in challenging lighting conditions.

To make the HOG computation faster, we decide to use an approach similar to [149] in which the Gaussian smoothing process is skipped reason of efficiency. However, in contrast to that, we benefit from the fact that we already know the position and size of the features that is necessary to compute. So, instead of using an integral histogram which needs a time of $2N$ memory accesses (where N is the number of pixels of the image) for the integral propagation and 4 memory accesses per bin for the feature computation, we use a direct feature computation instead which takes a similar time for the pre-computation, but it needs only 1 memory access per bin because the feature value is already saved in memory.

3.3 Training algorithm

The training of the multiresolution cascade consists of learning separately the linear SVM detectors. In contrast to Adaboost, each detector is trained independently from the previous one in the cascade. The selection of the negative examples is similar to the method proposed by [39] although in our case we do not use latent variables for object parts. Each detector is initially trained using (i) cropped and resized images of human as positive examples and (ii) randomly cropped and resized image regions not containing human as negative examples. After that, the learned detector is refined in an iterative process by selecting the most difficult negative examples (hard examples) from images not containing human. This helps to better populate the sampling space of the negative examples without increasing the SVM memory requirement and also improves the discrimination capability of the final detector.

The detection score $g_r(W)$ for each level r of the cascade and for a certain window W with associated features x, is computed as:

$$g_r(W) = \sum_{i=1}^{n} \alpha_i K(x_i, x) \tag{3.3}$$

where x_i and α_i are the support vector and corresponding weights learned in the training process respectively and $K(-,-)$ is an appropriate kernel. As we deal with linear SVM, we can substitute the kernel by scalar product rewriting Eq (3.3) as:

$$g_r(W) = \sum_{i=1}^{n} \alpha_i \langle x_i, x \rangle = \langle \sum_{i=1}^{n} \alpha_i x_i, x \rangle \tag{3.4}$$

This allows us to compute the score as a single scalar product which is independent of the number of support vectors so it can highly speed-up the detection process.

Table 3.1
Detection algorithm using the Multiresolution Cascade. Up-sample operation is used for propagating the detections to the next resolution level and is defined in Equation 1.

Given image I, resolution level R, SVM classifier g_r,
  threshold t_r at resolution r
Calculate the HOG pyramid H_s of I (in section 2.2)
for each scale s
  resolution $r \leftarrow 1$
  $W_s \leftarrow$ valid detection windows of H_s
  while $r < R$ **and** $W_s \neq \emptyset$
    $W_s \leftarrow g_r(W_s) > t_r$
    $W_s \leftarrow upsample(W_s)$ (see Eq. (6))
    $r \leftarrow r + 1$
W_s are the final detection windows at scale s

For the cascade pruning a score threshold t_r is learned for each resolution level r. This establishes a trade-off between speed and accuracy. In practice, if at a certain cascade level r the score $g(W)$ is smaller than t_r, the detection is pruned and no further evaluation will be necessary. Otherwise the evaluation will go to the next cascade level and so on until reaching the last level, which will give the final detection score. To associate the threshold t_r to a corresponding amount of correctly detected positive examples, we fit the detection score of the positive examples to a Gaussian distribution $f(x; \mu_r, \sigma_r^2)$, where μ_r and σ_r are mean and variance of the detection scores for the r level. Thus, we obtain an estimation of the percentage of positive examples correctly detected by a certain threshold t_r based on the value of threshold that reaches a certain value of the cumulative density function $F(x; \mu_r, \sigma_r^2)$. Considering that F is, by definition, an increasing function from $[0, 1)$, its inverse can be used to obtain the optimal threshold

$$t_r = F^{-1}(p; \mu_r, \sigma_r^2) \tag{3.5}$$

given an expected percentage p of correct detections.

3.4 Detection algorithm

The algorithm for the detection search using the multiresolution cascade is shown in Table 3.1. For each scale s, all possible window positions W_s at the lowest resolution are scanned to evaluate the score $g_r(W_s)$ of the SVM classification. Those windows with a score higher than the threshold t_r will be propagated to the $r + 1$ level of the

cascade. This is done using the function $W'_s = upsample(W_s)$ defined as:

$$\begin{aligned} W'_s(2x,2y) = W_s(x,y)(1-x)(1-y) + W_s(x+1,y)x(1-y) \\ + W_s(x,y+1)(1-x)y + W_s(x+1,y+1)xy \end{aligned} \tag{3.6}$$

which is a bilinear up-sampling by a factor of two of the set of valid windows. Therefore, we map each detection score to the corresponding one in the next cascade level which has double resolution. In this way, a full search of the object over all the image is done only at the coarsest resolution. After that the next detectors in the cascade are applied only to the locations with high detection score.

3.5 Experimental results

We run our experiments on the INRIA person dataset. The dataset is divided into training and testing images. Training images are divided into 614 images containing a total of 1208 pedestrian instances and 1218 images not containing any pedestrian. Test images are divided in 288 images containing a total of 563 pedestrians and 453 images not containing any pedestrian. For comparison purposes, we use the same configuration of training and test data as proposed in [24]. Training images are used for training a linear SVM detector and for the selection of hard examples, while test images are used for the detector evaluation.

Fig. 3.2 summarizes the characteristics of the three detectors used in the multiresolution cascade. Each column represents a detector, from the coarser to the finer one. The first row shows an example image of the cascade process, where in each level the valid windows are drawn with different colors until reaching the final detection. The second row shows the HOG feature weights learned in the training process for each detector level. By increasing the feature resolution more details of a human silhouette can be observed. Finally, the detection performances are represented on the third column using the ROC curve which represents the number of false positives per window in the X axis and the percentage of correct detections in the Y axis.

Experiments of different combinations of the three detectors are shown in Table 3.2. The first row in the table represents the use of the finer resolution detector without any cascade, which corresponds to the original human detector presented in [24]. The detection rate of this detector is slightly lower than the original one because in our implementation we do not use Gaussian smoothing in the feature computation, which makes the features slightly less discriminative, but faster to compute. This detector is taken as reference to verify the increment of speed that one can get using exactly the same configuration but substituting the single detector with the multiresolution cascade. It is important to remark that the gain in the scanning time presented in the last column of the Table 3.2 only accounts for the gain in speed due to the cascade model. It is then excluded the gain due to the faster feature computation and due to

	Level 1			Level 2			Level 3			Cascade		
	Acc.	Rej.	Cost	Acc.	Rej.	Cost	Acc.	Rej.	Cost	Det.	Time	Gain
1	-	-	-	-	-	-	83	99.99	100	83	135	1
2	99.5	56	2.3	99.5	88	28.8	85	99.95	68.9	82	21.2	13.1
3	95	64	4.2	95	93	40	90	99.9	55.8	80	6.87	23.4

Table 3.2

MULTIRESOLUTION CONFIGURATIONS. THE EXAMPLES SHOW THREE DIFFERENT TRADE-OFF BETWEEN SPEED AND DETECTION PERFORMANCES: *Row 1* IS THE DETECTOR WITHOUT USING THE CASCADE, *Row. 2* IS USING THE CASCADE WITH HIGH ACCEPTANCE RATES AND *Row 3* IS USING THE CASCADE WITH LOWER ACCEPTANCE RATES. *Acc.* IS THE ACCEPTANCE RATE FOR EACH CASCADE LEVEL; *Rej.* IS THE REJECTION RATE; *Cost* IS THE PERCENTAGE OF TIME USED FOR EACH DETECTOR; *Det.* IS THE PERCENTAGE OF DETECTION RATE OF EACH DETECTOR AT 10^{-4} FALSE POSITIVE PER WINDOW; *Time* IS THE AVERAGE TIME IN μs NECESSARY TO SCAN A WINDOW; *Gain* IS THE ESTIMATED GAIN IN SPEED TO SCAN AN ENTIRE IMAGE CONSIDERING THAT THE CONFIGURATION 1 IS TAKEN AS REFERENCE.

the fact that we do not need any further feature computation for the multiresolution level.

The second and third rows of Table 3.2 show two different configurations of the multiresolution cascade. The second row represents the conservative case, where the cascade thresholds are very loose. This means that the detectors in the cascade are less selective and accept almost all positive examples to reach the final detector. This cascade configuration obtains a gain in speed of the scanning process of around 13 times the configuration without the cascade together with a reduced detection rate of around 1%. In the third row of the table, the detectors are tuned with a more restrictive threshold which allows the cascade to reach an increase of speed of more than 23 times with a reduction of the detection rate of around 3%.

In the table is also shown (see *Cost* column of Table 3.2) that, in contrast to [146], the computational load of the three detectors is not uniformly distributed. This is due to the constrain that we impose in the use of the multiresolution: fixing the resolution factor to two (every feature level has a size that doubles the size at the previous level) does not allow one to choose the computational load of each detector, but it allows the use of different values for the spatial search stride. The stride is high for coarse feature resolution which allows a high speed search, but it is low for finer feature resolution which means a better localization.

From Table 3.2 it is evident that our method improves in terms of speed more than one order of magnitude over Dalal and Triggs with little loss of accuracy. The most similar method to ours is [146] which also uses a multiresolution features to make the detection process faster. A quantitative comparison with this method is not really possible because no public implementation of the methods is available and the speed-up is given in terms of time (in contrast to our evaluation), which is totally dependent of the testing platform used for the experiments.

3.6 Conclusion

In this chapter we have shown that it is possible to speed-up an object detector using a multiple resolution model in a hypotheses-pruning cascade.

The proposed approach shows that in contrast to normal cascades of classifiers, using a multiresolution representation has many adavantages: (i) the first level of the cascade is represented as a coarse resolution which is faster to evaluate than the fine resolution and therefore easy negative hypotheses can be discarded with lower computation (ii) the coarse-resolution establishes the sliding window stride which is higher at coarse resolution and smaller at fine resolution and therefore in contrast to normal cascades also the variable stride contributes to the speed-up (iii) as features at multiple resolution are used for scale and cascade-level, the same features are computed only once for both spaces without any additional cost. Experimental results show that our method compared with a sliding window approach obtains an increment of speed up to 20 times, depending on the tuning of the cascade thresholds, but maintaining comparable accuracy.

Still, using a cascade-based scheme has some drawbacks: (i) in contrast to a sliding window method like [24] the computational time of the detector is not fixed, but it depends on the complexity of the image, (ii) the cascade should learn the pruning thresholds, so the speed-up can be used only when the training is finished and not during the training, for instance for selecting the hard negative examples (iii) the method do not take advantage of the image structure because the pruning depends only on the detection score. In the following chapter we present a new method to solve the aforementioned problems.

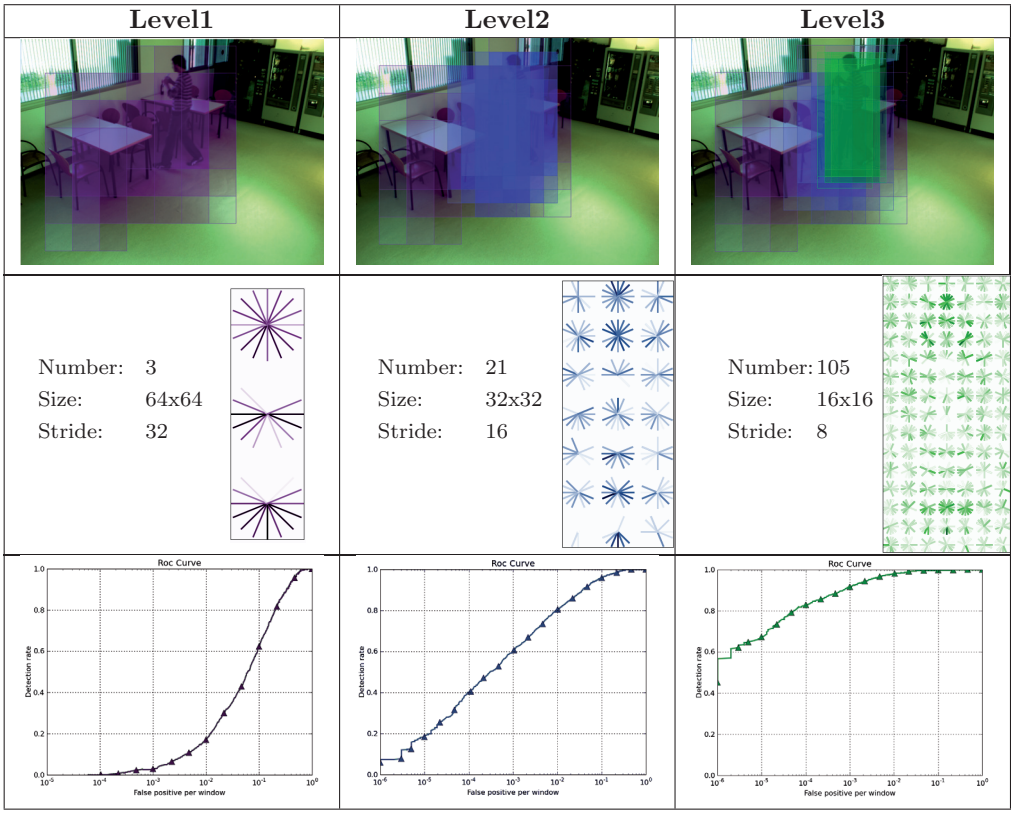


Figure 3.2: The three detectors composing the multiresolution cascade (best seen in colors). Each column represents a level of the multiresolution cascade: from left to right the coarser to the finer resolution. The first row shows an example image, where only the detections that passed the detector threshold are shown; the second row represents the weights associated to each HOG feature in the detector which have been learned by means of a linear SVM; the third row shows the ROC curve of the corresponding detector.

Chapter 4

Coarse-to-Fine Search

In this chapter we extend the use of a multiresolution model by introducing a new procedure to search for an object in the image. We call this procedure coarse-to-fine search becasue it searchs the object jointly over locations and scales in a coarse-to-fine manner. In contrast to the previous algorithm based on a multiresolution cascade, in the coarse-to-fine procedure the selection of the location hyphoteses to propagate the the next resolution is done using a local non maximal suppression. This avoids the explicit set of pruning thresholds. Throughout the chapter we empiracally show for different classes and datasest that the method has performance comparable to a cascade of classifiers in both speed-up and accuracy, but it has a constant speed-up that is independent of the image content.

4.1 Introduction

Although many improvements and enhancements have been made, the state of the art for detection is still far from the level necessary for real applications in terms of both speed and accuracy [30]. These two aspects are highly correlated: the newest and best performing methods for object detection, where multiple features [100, 124, 137, 136], multiple and non-linear kernels [51, 124] or deformable models [39] are employed, rely on a high computational power. All these approaches are based on the sliding window model, which is based on the concept of appling a classifier around over all possible scales and positions (in a brute-force way), scanning the image and searching for maximal detection responses. Optimizing sliding window (SW) search improves detection efficiency, allowing the use of more powerful and better performing techniques.

In this chapter we propose a new method to greatly speed-up the sliding window procedure (or image scan) based on a coarse-to-fine (CF) search. A simple illustration of the method is shown on Fig. 4.1. Instead of evaluating the object model (green box) everywhere over the feature space (Fig. 4.1 (a)), we decompose it as well as the

feature space into a hierarchy of resolutions, from coarse-to-fine (Fig. 4.1 (b)). In this way it is possible to realize a local search for maximas over resolutions. Correctly defining the sampling rate and the number of hypotheses to propagate the hypotheses over resolution can produce an enormous saving in computations (blue cells) while maintaining a high probability of finding all maximas.

Our implementation of the coarse-to-fine procedure based on HOG features runs twelve times faster than standard SW using exactly the same configuration. In contrast to cascade approaches, the speed-up is constant and independent of the capability of the detector to discriminate objects as well as the complexity of the image and the number of objects in the scene. This is a very important point, especially for real-time applications, where the detection time must be a very short and constant value, and can not vary from image to image as with cascades.

Using the variety of object classes of PASCAL VOC 2007 we evaluate the optimal detector configuration for the usage of multiple resolution features. We then show that on average our method performs similar or better than a threshold-based cascade. We also compare the performance of our model with state-of-the-art methods for object detection on the INRIA pedestrian dataset. Results show an excellent trade-off between accuracy and speed.

The rest of the chapter is organized as follows. Section 4.2, starting from standard SW, presents our coarse-to-fine approach in a well defined formulation. The learning process together with implementation details are given in sections 4.3 and 4.4. Section 4.5 discusses advantages and drawbacks of the new method. Finally, section 4.6 presents evaluations and comparison of our method with other ones in terms of both detection performance and speed. 4.7.

4.2 The Image Scanning Approach

In this section we first describe the standard SW as a vectorial convolution between an object model and image features. Suqbsequently, this formulation is extended to describe our coarse-to-fine procedure.

4.2.1 Sliding Windows

In SW, as described in [24], an object model is scanned over a pyramid of features representing an image. The pyramid of features is a set of matrices $H_s(x, y)$, where each element is an f-dimensional feature vector (see Fig. 4.2(b)). Each matrix H_s is built from a smoothed and sub-sampled version $I_s(x, y)$ of the original image at a certain scale s, as shown in Fig. 4.2(a).

The object model for a linear classifier is an $h \times w$ matrix $M(x, y)$, where each element is an f-dimensional weight vector, as shown in Fig. 4.2(c). The scale sampling of the pyramid of features is established by a parameter λ defining the number of levels in an octave, that is the number of levels we need to go down in the pyramid to get

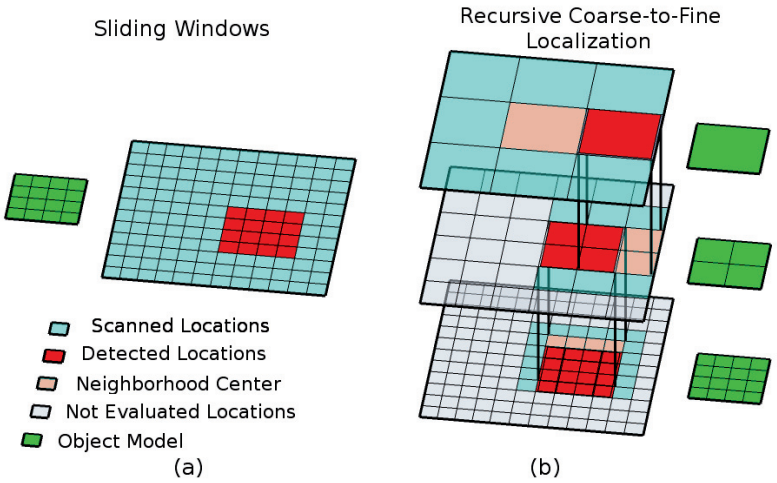


Figure 4.1: Sliding Windows versus Recursive Coarse-to-Fine Localization. (a) In the standard SW search, the object model (green) must be evaluated at all image locations (cyan) to detect the object (red). (b) In the coarse-to-fine procedure the object model as well as the image are decomposed at multiple resolutions and the search is done in a coarse-to-fine manner. The computational cost of our method is much lower than SW because of the higher stride and the simpler model used at coarse resolution.

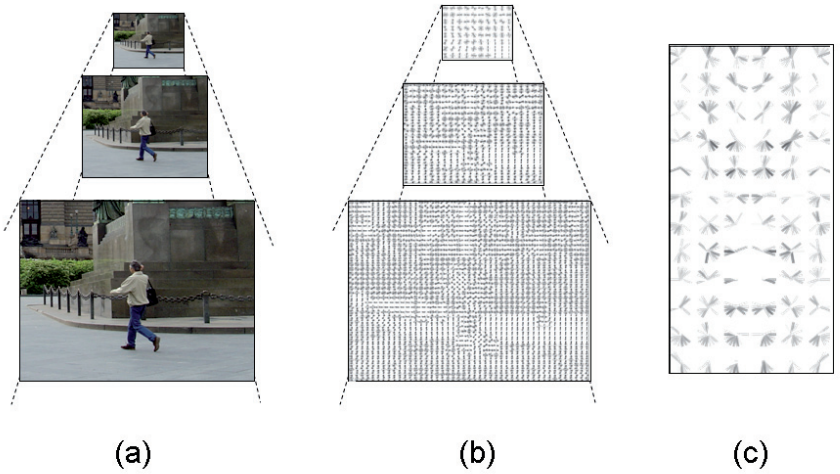


Figure 4.2: Sliding windows components: (a) Pyramid of images I_s: computed by repeated smoothing and sub-sampling of the original image. (b) Pyramid of features H_s: from every scale of the pyramid of images, the corresponding matrix of features is extracted. (c) Object model M: an $h \times w$ matrix of f-dimensional weight vectors.

twice the feature resolution of the previous one.

The response D_s^{sw}, or score, of the object model centered at position (x, y) and scale s is defined as:

$$D_s^{\text{sw}}(x,y) = \sum_{\hat{x},\hat{y}} M(\hat{x},\hat{y}) \cdot H_s(\hat{x}+x-w/2, \hat{y}+y-h/2), \tag{4.1}$$

where $\hat{x} \in \{0,1,\ldots,w-1\}$, $\hat{y} \in \{0,1,\ldots,h-1\}$. Note that the symbol $(-\cdot-)$ represents the scalar product because each element of M_s and H_s are f-dimensional vectors. In this way, D_s^{sw} is a pyramid of matrices of the same size as H_s, but where each element is a scalar that represents the response of the object model in the corresponding position and scale. Each element of $D_s^{\text{sw}}(x,y)$ is converted to the corresponding image bounding box center

$$\begin{aligned} B_s(x,y) &= (2^{\frac{s}{\lambda}}kx, 2^{\frac{s}{\lambda}}ky) && (4.2)\\ &\equiv k2^{\frac{s}{\lambda}}(x,y), && (4.3) \end{aligned}$$

where k is the number of pixels a feature occupies in the rescaled image (in our case 8 pixels).

For the sake of simplicity, in the following we will use the notation of Eq. (4.3), i.e. coordinate-wise scalar multiplications, as equivalent to notation in Eq. (4.2). Eq. (4.3) describes SW in terms of image coordinates, which is more natural.

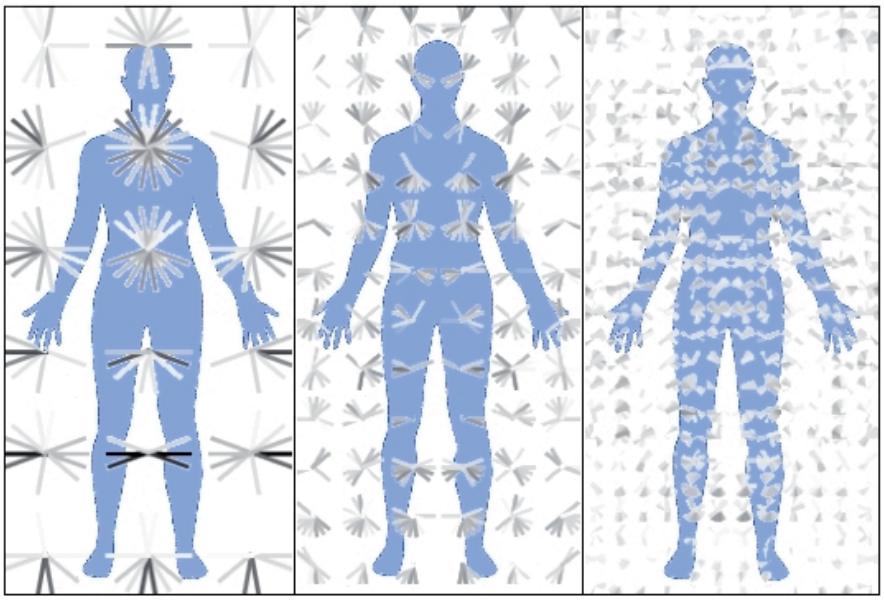


Figure 4.3: HOG pyramid model M for the class person with $w = 3$, $h = 6$ and $l = 3$. The low resolution features ($d = 0$) give a general coarse representation of the human silhouette, while the high resolution ($d = 2$) focuses more on details.

The same conversion of Eq. (4.2) is also applied to the bounding box size (w, h). In this way, we obtain all the necessary information to associate each score $D_s(x, y)$ with the corresponding image bounding box. Applying non-maximum suppression we obtain the bounding box of the final detection.

4.2.2 Coarse-to-Fine Localization

In the coarse-to-fine localization the object is searched in space but at different resolutions, from coarse to fine. The final score of the detector is now the sum of partial scores, one for each resolution. For this reason, the object model is a dyadic pyramid composed of l levels, where each level d is a matrix M_d of weight vectors. An example of a 3-level pyramid model for the class person is shown in Fig. 4.3.

The computation of the partial score R_s^d for a resolution level d of the object model pyramid at a position (x, y) and scale s of the pyramid of features is then:

$$R_s^d(x,y) = \sum_{\hat{x}_d, \hat{y}_d} M_d(\hat{x}_d, \hat{y}_d) \cdot H_{s+\lambda d}(\hat{x}_d + x - 2^{d-1}w, \hat{y}_d + y - 2^{d-1}h), \tag{4.4}$$

where $\hat{x}_d \in \{0, 1, \ldots, w2^d - 1\}$, $\hat{y}_d \in \{0, 1, \ldots, h2^d - 1\}$. When $d = 0$ this is exactly Eq. (4.1). When the resolution level d is greater than 0, it is necessary to move down λd levels in the feature pyramid to reach the corresponding resolution level. Note

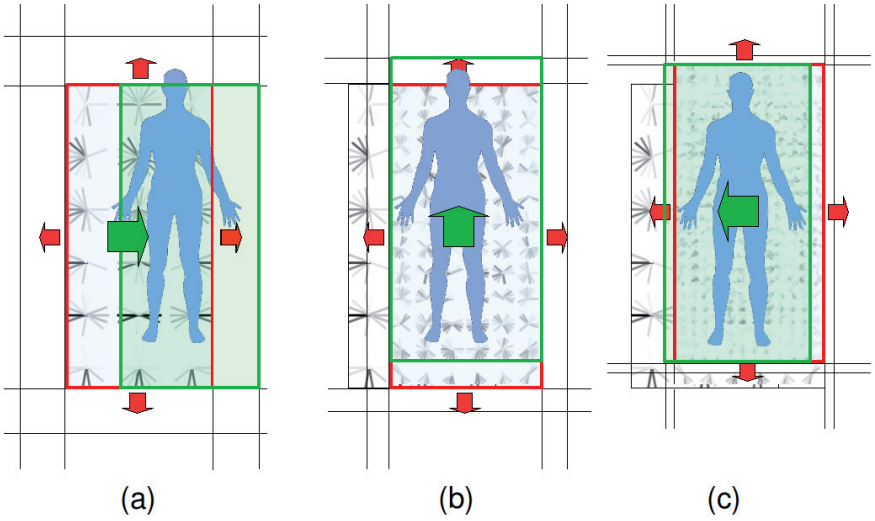


Figure 4.4: Example of a coarse-to-fine detection. In (a), at a certain position (x, y) and scale s (red box) of the pyramid of features H, the best location $\Pi_s^0(x, y)$ (green box) for the low resolution model of the object M_0 is sought in the local neighborhood $\Delta_\delta(x, y)$. In (b), the same procedure is repeated for the next resolution level $s + \lambda d$, using as center of the neighborhood the best location computed at low resolution $\Pi_s^0(x, y)$. The process is recursively repeated for all feature resolution levels. In (c), the location obtained at the finest resolution $\Pi_s^2(x, y)$ is the location of the final detection and can be converted to pixels using Eq. (4.9). This figure is best viewed in colors.

that the same level of features $H_{s+\lambda d}$ is used multiple times for searching the same model M_d at different scales s. In this way we create a connection between scale and resolution, and the coarse-to-fine procedure can be executed without the need of computing any additional feature, but using multiple times exactly the same features used in normal sliding windows.

For each $H_{s+\lambda d}$, the search space is split into neighborhoods $\Delta_\delta(x, y)$ defined as:

$$\Delta_\delta(x, y) = \{(\hat{x}, \hat{y}) | \hat{x} = x + d_x, \hat{y} = y + d_y\}, \tag{4.5}$$

where $d_x, d_y \in \{-\delta, -\delta+1, \ldots, \delta-1, \delta\}$ and δ is the radius of the neighborhood. The neighborhood represents all the locations where a single object can be found. A key difference between SW and the coarse-to-fine localizatin is the number of hypotheses that are evaluated. While in SW the number of hypotheses depends only on the sliding window stride, in our method the number of hypotheses depends also of the size of the initial neighborhood, and more importantly, it is not increased when propagating hypotheses to higher resolutions. We define Π_s^0 for each (x, y) and scale s as the location that maximizes the partial score R_s^0 over the neighborhood Δ_δ :

$$\Pi_s^0(x, y) = \underset{(\hat{x},\hat{y}) \in \Delta_\delta(x,y)}{\arg\max} R_s^0(\hat{x}, \hat{y}). \tag{4.6}$$

Notice that (x, y) is the location of the center of the neighborhood at the coarse resolution at scale s, while Π_s^0 is the location of the object estimated by M_0. Since we optimize the score of R_s^0 over the neighborhood Δ_δ, it is not necessary to compute each (x, y). To select the correct sub-sampling of (x, y) is necessary that all locations be scanned at least once, which implies a sampling of $(\hat{\delta}x, \hat{\delta}y)$ with $\hat{\delta} \leq \delta$. In oreder to avoid the evaluation of the same location multiple times in adjacent neighborhoods we fix $\hat{\delta} = \delta$. The optimal position at levels $d > 0$ is recursively defined as a refinement of the position at $d - 1$:

$$\Pi_s^d(x, y) = \underset{(\hat{x},\hat{y})\in\Delta_1(2\Pi_s^{d-1}(x,y))}{\arg\max} R_s^d(\hat{x}, \hat{y}). \tag{4.7}$$

For $d > 0$ the neighborhood is fixed to Δ_1. This is because, as the model resolution doubles when going from one level to the next, a displacement of 1 is enough to correct for bad localizations at coarser resolution. A bigger neighborhood would require more computation, because more locations need to be evaluated, which is not necessary.

Recall our notational convention for coordinate-wise scalar multiplication, so that $2\Pi_s^{d-1}(x, y)$ represents a doubling of the coordinates for the object estimate at resolution $d - 1$. An example of recursive localization refinement is shown in Fig. 4.4.

Knowing the optimal position of the object model at each level d, we calculate the total score $D_s^{\text{cf}}(x, y)$ as:

$$D_s^{\text{cf}}(x, y) = \sum_{\hat{d}} R_s^{\hat{d}}(\Pi_s^{\hat{d}}(x, y)), \tag{4.8}$$

where $\hat{d} = \{0, 1, \ldots, l - 1\}$. The computation of the bounding box of each score $D_s^{\text{cf}}(x, y)$ is similar to that of standard sliding windows. However, now (x, y) represents the location of the detection at the coarsest level. To obtain the location at the finest level $l - 1$ it is necessary to convert the coordinates at Π_s^{l-1}. The center of the bounding box B for position (x, y) and scale s is thus:

$$B_s(x, y) = k2^{\frac{s+\lambda(l-1)}{\lambda}} \Pi_s^{l-1}(x, y). \tag{4.9}$$

The final detection is computed like in normal SW by applying non-maximum suppression. Note that the local non-maximum suppression applied to each neighborhood during the CF refinement is not enough to avoid multiple detections of the same object. For instance, if in an image an object instance is in between two neighborhoods, both of them will produce an high detection score on the object, which is what must be eliminated with global non-maximum suppression.

4.3 Learning

Given a set of input data $\{x_1, \ldots, x_n\}$ and the associated labels $\{y_1, \ldots, y_n\}$, we find a parameter vector w of a function $y(x_i; w)$ that minimizes the regularized empirical risk:

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\max(0, 1 - y_i y(x; w)). \tag{4.10}$$

In our problem the input data x_i is a set of multiple resolution features (extracted from the pyramid H_s defined in section 4.2.1) associated with an image region, while the output data $y_i \in \{-1, 1\}$ indicates whether the object is present in the region.

The estimated output y depends on the relative position of each feature level with respect to the previous level. This allows us to obtain a better estimate of the object location at each level. We introduce a structured latent variable h that is a vector of tuples $(h_{x,d}, h_{y,d})$ representing the relative position of a certain level d with respect to the previous $d-1$. The estimated output is:

$$y(x_i; w) = \max_h \langle w, f(x_i, h)\rangle \tag{4.11}$$

where $f(x_i, h)$ is a function that maps the input features x_i to the corresponding latent variable h. In our case, for each location (x, y):

$$\langle w, f(x_i, h)\rangle = \sum_d R_s^d(2^d x + \sum_{\hat{d}=0,d}(2^{d-\hat{d}} h_{x,\hat{d}}), 2^d y + \sum_{\hat{d}=0,d}(2^{d-\hat{d}} h_{y,\hat{d}})). \tag{4.12}$$

From Eq. (4.4) we see that w corresponds to the flattened version of M, our object model. Note that in Eqs. (4.12) and (4.11) the scalar product can be substituted with another non-linear kernel by applying the kernel trick. The only restriction imposed by our method is that the final score be correctly computable as the sum of the partial scores generated by the different resolution levels of the model. This is satisfied by other kernels like χ^2, intersection, Hellinger.

Now, we can compute the real maximum of f, evaluating all possible locations for each resolution d or its faster approximation which is the coarse-to-fine recursive refinement of Eq. (4.8): s

$$\max_h \langle w, f(x, h)\rangle \approx D_s^{\mathrm{cf}}(\hat{x}, \hat{y}) \tag{4.13}$$

where $(\hat{x}, \hat{y})$ and s corresponds to object location and scale at the lowest resolution.

We use the real maximum of f for training to avoid problems due to the approximate estimation of the latent variable h, while during testing we use the CF approximation. We leave for feature work the analysis of the effects of using the CF approximation during training.

In contrast to normal SVM optimization, y is no longer linear in w, therefore the empirical risk is no longer convex and standard optimization techniques can not be used. However f is still convex in w since it is a maximum of linear functions. Thus, the empirical risk is convex for $y_i = -1$ but concave for $y_i = 1$. In order to optimize this function we use the latent SVM optimization proposed in [39]; learning is divided

into two iterative steps: optimization of w with fixed h for the positive examples and the estimation of the best h for the positive examples using the computed w.

Another problem with learning is the number of negative examples. While positive examples are costly to obtain and thus their number is always quite limited, the number of negative examples can be very high and can be obtained from images not containing the object to detect. Using a large number of negative examples can help boost performance, but it can make the learning process prohibitive in terms of time and memory. To solve this we use cutting-plane [54] that consists of an iterative algorithm that first estimates w using a subset of the full training set and then selects the most violated constraints that will be added to the training set of the next estimation of w. This yields much faster learning and assures that the algorithm converges to the solution obtained with the full set of examples.

4.4 Implementation Details

We implement the coarse-to-fine procedure based on HOG features, which are widely used in SW-based object detection [24, 51]. However, the proposed framework is not limited to only HOG and it can be applied to any (and multiple) features, like Haar [130], SIFT [66] or LBP [136].

Features. We use the HOG feature implementation proposed in [40]. The features for each square region are 31-dimensional: 9 contrast insensitive features, 18 contrast sensitive features and 4 representing the overall gradient of four neighbor regions.

Object model definition. The object model has few parameters to tune. The aspect ratio of each object model is chosen based on the mean aspect ratio of the object bounding boxes of the training set. We fix the number of HOGs to use at the lowest resolution object representation. The size at higher resolutions double the previous because we use a dyadic pyramid representation. The number of features for the object representation is a trade-off between better discrimination (many features) and the capability to detect small objects (few features).

Positive examples. We convert an image containing positive examples into a pyramid of features (as described in section 4.2) and then search over space and scale for the best fit between the object model bounding box and the training example bounding box using the overlap ratio defined in [35]. If the overlap o is greater than 0.5, the example is taken as a positive sample and added to T_p, otherwise it is discarded.[1]

Negative examples. Negatives examples T_n are extracted from images not containing the object class. As for the positives, the image is converted to a pyramid of features. The examples are initially drawn using a uniform distribution over both space and scale. Subsequently, they are selected based on the cutting plane technique explained above.

[1] Generally the overlap is less than 0.5 when the object is very small or when the aspect ratio is very different from the one chosen for the detector model.

SVM training. Positive T_p and negative T_n pyramids of features are flattened to vectors and used to train a linear SVM using libSVM [17]. The result of this is a weighted sum of support vectors. Since the kernel is linear, these are summed up into a single vector of weights w. This is then converted back to the dyadic pyramid representation, resulting in our object model M.

Neighborhood size In the coarse-to-fine procedure the only parameter that has to be selected is the size of the initial neighborhood defined by δ. In general, bigger is the neighborhood, faster is the method; however, a very wide neighborhood can produce two collateral effect that can reduce the final detector accuracy: (i) a wide neighborhood implies a more difficult choice of the best hypotheses to propagate and consequently a higher probability of committing localization errors (ii) a wide neighborhood can produce a local region where more than one object can fit at the same time. This condition is opposite to our initial assumptions, thus only the hypothesis of one of the two objects will be propagated to the next levels and finally detected. Nevertheless, we experimentally verified in section 4.6 that using a $\delta = 1$ assures to robustly overcome the previous problems, but still guarantee a great speed-up.

4.5 Discussion

The coarse-to-fine search scans the image in two ways at the same time. It scans the image spatially, searching as a standard SW for the object. It simultaneously scans the image in the resolution dimension, from coarse to fine. The number of hypotheses to scan is established by the first and coarsest level of the pyramid model and is a set of neighborhood regions uniformly distributed over the image. Subsequent levels of the pyramid object model refine the hypotheses to the best location inside each neighborhood.

Our method has some similarities with [39]. Both methods are based on latent SVMs used refine the location of the object model. However, in [39] the latent variables represent object parts that can move with respect to a global model. The model learns the best cost to associate to the displacement of the parts, to make the detector as much discriminative as possible.

In our method we do not have parts, but rather describe the same object at different resolutions and use the latent variables to refine the object localization from coarse-to-fine and therefore to speed-up the search. Logically, not considering local alignment of parts but just a global alignment of the entire object lowers the detector accuracy. An evaluation of this is given in chapter 5, when local deformations are introduced.

In contrast to previous methods based on cascades [130, 146, 149, 27], there is only one classifier to train. The only assumption made is that the object has an appearance that can be modeled in a top-down manner. That is, global views of an object contain most of the relevant information needed to support reliable recognition [115], although specific details may be helpful for further refinement. This is a biologically viable assumption to make, as Rao et al. [91] showed that the human visual search proceeds

in a coarse-to-fine manner, which supports the claim that all the objects that can be easily identified by a human have this top-down appearance property.

In cascade-based approaches, for each sliding window location the score of the classifier is used to evaluate whether to discard the location or continue to the next classifier in the cascade. This means that the score provided by a small number of features must be precise enough to take a difficult choice that will greatly affect overall detector performance. Therefore, to obtain reliable decisions, the detector must be conservative, discarding only a fraction of hypotheses. In contrast, our method does not consider each hypothesis location as independent; it groups locations into small regions (neighborhoods), and for each of these a single choice is propagated to the next level. This is a much easier decision to take, because it reduces to finding a local maxima in a small neighborhood of hypotheses. This is what allows our coarse-to-fine search to perform as fast as cascades without any rejection thresholds.

From a general view, object detection can be seen as finding the local maximas of a scoring function D. What the coarse-to-fine procedure does is to use the additional constraint that each local maxima of D have a minimum distance k to every other local maxima. This is justified from the fact that two objects in the space can not physically occupy the same location. Thus, dividing the search space into neighborhoods of radius smaller than $k/2$ assures that for each neighborhood only one local maxima is found. Consequently, it is possible to find this using a greedy search in a smoothed enough version of D. Considering HOG features at coarse resolution is an approximation of a smoothed version of the fine respresentation, which is exactly what is needed to correctly find the local maxima.

In this regard, our method has some similarities with those based on the Hough transform such as [71, 63, 62]. These methods use a mean-shift or gradient-based estimation of the local maxima. This speeds-up the search and avoids a complete evaluation of the detection space. With our method a greedy search is performed over feature levels to avoid evaluating the finest level everywhere. In both cases, if the sampling (propagation) of the hypothesis is not dense enough, good detections can be missed.

The fact that the method does not use thresholds to prune hypotheses is a great advantage because it does not need any validation phase for estimating thresholds, and it enables its possible use also in the training phase, for learning the latent variables. However, the substitution of score thresholds with a local non-maximal suppression strategy also signifies the inter-exchange of an accuracy guarantee (given by the validation procedure) with a speed-up guarantee (given by the manner the image scan is applied). Still, in our test the coarse-to-fine search reaches the same accuracy level as threshold-based methods.

$l \setminus \delta$	0	1	2	3
1	1.0	1.0	1.0	1.0
2	0.4	3.2	6.6	9.2
3	1,4	**12.2**	31.2	54.8
4	5.4	48.2	131.1	249.3

Table 4.1
SPEED-UP FACTOR $g(l, q)$ FOR DIFFERENT VALUES l AND δ.

4.6 Experiments and Results

4.6.1 Databases

We evaluate our detector on two different and complementary databases. The first test uses the PASCAL VOC07 dataset [35]. We use this database as reference to evaluate the best configuration of our algorithm, as well as to compare the algorithm against a cascade-based approach.

The second test is on the INRIA person dataset [24] In this case the database contains only pedestrians, but almost all methods that use this database are more focused on real applications where speed is also critical. Furthermore, this database contains many humans, often overlapping each other. Hence, the INRIA dataset is also an optimal testbed to show that our algorithm can properly detect multiple and overlapping object instances.

4.6.2 Neighborhood Radius, Resolution Levels and Speed-up Factor

The neighborhood radius δ and number of resolution levels l are the two most important parameters that influence the final performance of the detector. While for resolution levels greater than zero δ is forced to be 1 to ensure coherence of representation of the model over resolutions, for level zero δ is free and greatly affects the speed-accuracy trade-off.

Using a neighborhood of radius δ for level zero corresponds to scanning $q = (2\delta + 1)^2$ locations at the first level and 9 locations for subsequent levels. So, a model of l levels requires $q + 9(l - 1)$ evaluations instead of $q4^{l-1}$ as in standard SW working at the finest level. However, the cost of scanning a location is proportional to the object model resolution which doubles at each level of the pyramid model. So, the ratio between the cost of brute-force search and our recursive localization approach is:

$$g(l,q) = \frac{q4^{l-1}}{\sum_d \frac{9}{4^d} + \frac{q}{4^{l-1}}} \tag{4.14}$$

where $d = \{0, 1, \ldots, l-2\}$. This can be simplified to:

$$g(l,q) = \frac{q4^{l-1}}{12 + \frac{1}{4}^{l-1}(q-12)} \tag{4.15}$$

Table 4.1 shows the speed-up of the image scan for different values of l, the resolution levels of the model and δ, the neighborhood radius. Note that, the speed-up considers only the image scan, the remaining parts of a complete detection: feature computation and non-maximum suppression of the detection are not considered because they remains the same as in normal SW. The computational cost of the coarse-to-fine search, compared to standard SW, is reduced proportionally to the number of levels of the object model l and neighborhood locations q. In experiments l is bounded by the resolutions available in images of the object and the memory needed for training. For the choice of δ we have to consider that a neighborhood must contain a unique hypothesis for an object. Therefore, to correctly localize partially overlapping bounding boxes it is necessary that, within a neighborhood, all possible detections overlap each other enough to be grouped together by non-maximum suppression.

The intra-class overlap is class and relative-position dependent[2]. A maximum overlapping of 0.2 assures fusing 99% of the object instances correctly.

Limiting the minimum resolution for the lower resolution model to at least 3×3 HOG cells assures a minimum overlap of 0.2 for $\delta \leq 1$. For $\delta = 1$ the speed-up factor g of our method with respect to normal SW is shown in the second column of table 4.1. This varies from 1 for $l = 1$ to 48.2 for $l = 4$ levels of resolutions and is totally independent of the image and the object model.

In table 4.2 the average number of SVM evaluation necessary to scan an image is shown, comparing standard SW with our method using different resolution levels. Using the coarse-to-fine search with only 1 level obtains exactly the same results and speed as a normal SW approach. Increasing the resolution levels reduces the number of evaluations. With 3 levels, our method has more than one order of magnitude less evaluations than SW as predicted in Table 4.1.

4.6.3 Levels of Resolution

It is necessary to establish how many levels of feature resolution are the best for a given problem. For our experiments, we evaluate our method for all classes of the PASCAL VOC 2007 database. To speed-up the experiment and because we are interested only on the relative performance of different configurations, we test only on positive examples (i.e. images containing the chosen object class).

[2]That means, (i) humans appear together more often than cats and (ii) besides another human but almost never below or above him.

Method	Evaluations per Image
SW	22,156
CF 1 level	22,156
CF 2 levels	6,722
CF 3 levels	1,583

Table 4.2

Average number of SVM evaluations in the cat class of VOC PASCAL2007 for SW and coarse-to-fine search with different resolution levels.

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
l=1	22.4	39.2	10.5	3.6	17.4	37.5	36.8	23.4	15.5	20.8	33.6
l=2	28.3	43.3	11.5	4.5	29.0	45.7	39.3	28.8	16.0	27.4	36.3
l=3	28.0	37.3	9.6	3.6	22.1	45.8	36.7	26.6	14.8	35.2	31.6
	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mean**	**speed**
l=1	19.2	45.4	36.5	23.6	16.2	19.3	33.3	26.5	44.7	26.3	1.0
l=2	24.7	42.9	38.0	22.1	16.3	27.7	34.1	31.3	47.7	29.7	3.2
l=3	26.7	43.9	37.7	21.5	15.1	27.2	30.6	28.2	46.0	28.1	12.2

Table 4.3

Average-precision computed on positive examples of TEST set of the PASCAL VOC 2007 database. The detectors have been trained using the TRAIN+VAL set. Rows represent results of the coarse-to-fine procedure using a different levels of resolutions, from 1 to 3. Columns represent the 20 VOC object classes plus mean median and speed-up of the image scan.

We test three different coarse-to-fine configuration, with 1, 2 and 3 levels of feature resolution. Increasing the number of features in the detector increases the average-precision score by providing more information about object shape. However, using higher resolution prevents to detect small objects. To make the comparison fair, we fix the number of features (but getting the best bounding box ratio as explained in section 4.4) for the maximum resolution level to be the same for all configurations. So, *CF* 1 *lev.* has one level of resolution of around 240 HOG cells, *CF* 2 *lev.* has two levels of around 60 and 240 HOG cells, and finally *CF* 3 *lev.* has three levels of around 15, 60 and 240 HOG cells.

Detection results and speed-ups of the image scan are reported in table 4.3. Mean and median values show that the best performance in terms of average-precision is the configuration with 2 resolution levels. However, the speed-up of this configuration is only 3.2 times. Moving to 3 resolution levels the performance is still good, but the speed-up is increased to 12.2 times. This makes this configuration an optimal trade-off between performance and speed and is be the configuration used in all the following experiments. Note that the general trend of performance is not identical for all classes. For examples for buses, cows and dogs, 3 levels is the best configuration, while for

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
Exact	24.1	41.3	11.3	3.9	20.8	36.8	35.4	25.5	16.0	19.4	21.2
Cascade	24.1	38.7	12.9	3.9	19.9	37.3	35.7	25.9	16.0	19.3	21.2
Speed	9.3	9.8	9.3	9.9	3.9	18.1	13.8	17.3	9.5	12.1	6.4
CF	23.6	39.4	12.9	2.7	19.7	39.2	34.5	25.9	17.0	21.6	23.1

	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mean**	**speed**
Exact	23.0	42.9	39.8	24.9	14.6	14.3	33.0	22.8	37.4	25.4	1.0
Cascade	23.0	40.2	41.5	24.9	14.6	15.1	33.2	23.0	42.2	25.6	10.9
Speed	3.3	17.6	20.1	3.6	6.4	19.0	15.0	9.8	2.8		
CF	24.1	42.0	41.1	25.3	14.2	15.8	29.6	22.5	41.0	25.8	12.2

Table 4.4

Average-Precision computed on positive examples of TEST of the VOC2007 database. *Exact* shows the results of a brute force method; *Cascade* represents the result of a cascade method with thresholds chosen for obtaining the same performance as exact up to precision equals recall; thresholds are computed using the validation set of VOC2007; *Speed* is the average speed-up per class achieved for Cascade; *CF* is our method using three resolution levels.

horses and person 1 level is instead best.

4.6.4 Comparison with Cascades

Our method intends to improve threshold-based cascade detectors. In this section we compare these methods, showing the advantages and drawbacks of both. Methods implementing cascades based on HOG have been developed in recent years [149, 146, 37]. However these methods use different HOG implementation, different parameter configurations and different learning strategies so that a fair comparison is impossible.

We implement our own version of a cascade detector. To allow a full comparison with our method we keep the same configuration based on 3 levels of feature resolution. In this sense the cascade is similar to [146], but we improve the learning strategy by joining all features from different levels into a single SVM optimization, exactly the same used for CF and explained in section 4.3.

Using the same learning and features assures that changes in accuracy or speed are totally due to the method, not to implementation details or different learning strategies. To select the optimal thresholds we use the method proposed for the star-cascade in [37]. We threshold the cumulative sum of the partial scores of Eq. 4.4. In practice, for each resolution level of the object model we compare the partial score so far obtained with a learned threshold and if the score is higher than the threshold, the search continues, otherwise it stops. The thresholds are set so the resulting precision-recall curve of the cascade detector reaches the precision-equals-recall point without any pruning. For each class it is thus necessary a separate set of

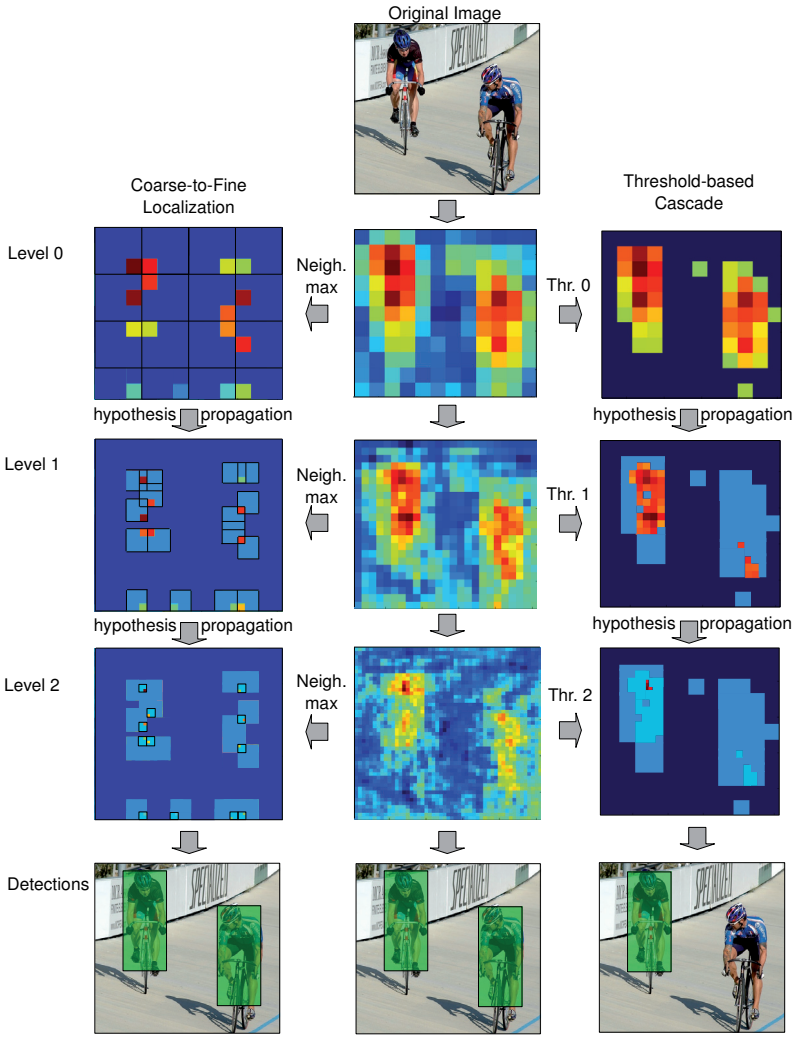


Figure 4.5: Example of a scan over three resolutions using three methods: left column *coarse-to-fine search*, central column *Exact*, right column *Cascade*. Images represent the partial detection score R_s^d at the object scale s and for resolution levels $d = \{0, 1, 2\}$. Red locations represent high detection score. For CF and Cascade dark blue pixels represent locations that are pruned and do not need further computation. In this example the cascade threshold is too high and prunes good hypotheses, missing the second cyclist. In coarse-to-fine both detections are made because the algorithm retains hypotheses at all locations. This figure is best viewed in colors.

examples that serves to learn the thresholds values.

We compare the two methods also with a brute force approach in all classes of PASCAL VOC 2007. In this experiment we train the detectors with only the training set, while the validation set is used for threshold learning. Results are reported in table 5.4. For the cascade we also reported per-class speed-up, while the final speed-up is the average of all classes.

Both speed-up methods not only improve speed, but in some cases also average-precision. This is due to the pruning of false positives. Also, consider that even without any threshold expressly tuned for it, the coarse-to-fine search obtains an average performance better than that of the cascade detector. This gives a clear indication that recursive localization is a efficient strategy for pruning hypotheses: (i) it obtains the same or better performance than cascade on most classes; and (ii) it assures that detector speed does not depend on object class or image conditions which is very important for real-time applications; and (iii) it requires no additional parameter tuning.

Note that the speed-ups shown on Table 5.4 and 4.3 only represent the increase of speed due to the faster image scan and do not take into account the feature computation and the final non-maximum suppression. To give an approximate idea of the final detection time consider that on our machine, using a single CPU, the feature computation for a VOC2007 image takes on average 0.6 seconds, the non-maximum suppression 0.02 seconds and the image scan goes from 1.7 seconds for SW to 0.13 seconds for our 3-level configuration. This means that when using a faster image scan, the time that dominates a detection changes from the SW search to the feature computation, producing a global speed-up of around 3 times. This speed-up can be greatly increased using faster feature computation (i.e. using multicore processors or GPU).

Fig. 4.5 illustrates the pipeline of the pruning strategy of the coarse-to-fine search and cascade-based detectors for the class person at a certain scale. Both strategies use exactly the same detection scores (central column) based on our multiresolution HOG implementation. On the left, our method uses a constant factor of hypotheses pruning which is established by the neighborhood size and the sub-sampling factor (see section 6.2) , while on the right the cascade uses a pruning method that depends on the current image complexity. This suggests that for cascades, learning thresholds based on the partial score information can achieve higher pruning. However, due to the high variability of conditions in images, the partial score is not very reliable, and better information can be obtained by considering local score variation, as in our approach.

4.6.5 INRIA pedestrian dataset

The INRIA dataset is the standard evaluation test for human detection [24]. We found the original evaluation methodology to be prone to errors, as originally pointed out by Dollar et al. [30]. A better evaluation metric was proposed by the same authors,

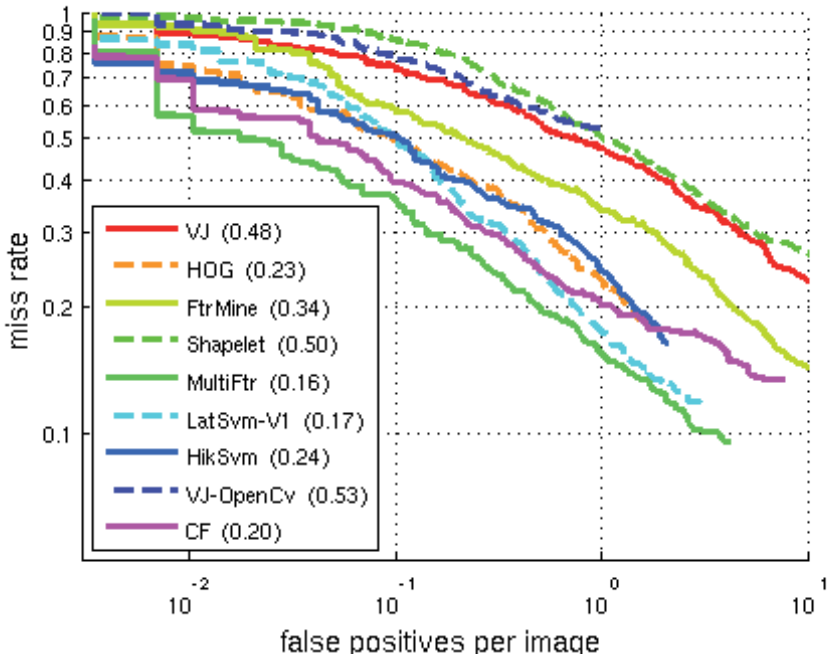


Figure 4.6: False Positive Per-Image on the INRIA database. VJ [132], HOG [24], FtrMine [29], MultiFtr [137], LatSvm-V1 [39] , HikSvm [69], CF is the coarse-to-fine approach.

where the evaluation is done on a per-image basis. This is equivalent to a precision recall curve, but for certain tasks it is preferred because it gives an absolute measure of the average number of false positives to expect per-image (FPPI).

We test our method using the same detector configuration with 3 resolution level as the one used for VOC2007. A comparison of CF HOG with other methods is shown in Fig. 4.6. The coarse-to-fine search reduces the standard HOG miss-rate by 3 points at 10^0 FPPI, by 10 points at 10^{-1} FPPI and by 14 points at 10^{-2} FPPI. Globally, two methods perform better than our approach. However, *MultiFtr* uses multiple and complementary features to improve the HOG results while *LatSvm* learns the object deformations using a latent variables.

In terms of speed, in our machine, our method takes 1.0 seconds to process an image of 640×480 pixels, but around 0.8 second is used to compute the features and only 0.2 seconds for the scan of the image. In contrast to most of the others methods, where a very significant part of the time is used for scanning, with the coarse-to-fine procedure this time is reduced to a small fraction of the total detection time.

4.7 Conclusions

In this chapter we introduced a general multiscale and multiresolution detection framework based on a coarse-to-fine search. It improves the detection of object in still images, yields excellent results on two well-known databases, and is significantly more efficient than comparable methods. In this framework we can use any dense feature descriptor, decompose it into a pyramid of increasing resolutions of features, and use it to scan the image looking for an object in a much faster manner.

The method combines prior information about the search for object location hypotheses with a coarse-to-fine localization to optimally redistribute the computation necessary to detect objects. Compared to cascade approaches, our method obtains similar detection and speed performance, but assures a constant speed-up independent of object class and image conditions and does not require rejection threshold to prune hypotheses. This makes the method also very suitable for real-time applications, where fast but constant frame-rate detection is necessary.

In terms of accuracy, the method is comparable to similar approaches like [24], but it is still far from the best deteecton performance that are obtained by deformable part-based models like [39]. In the next chapter we extend the coarse-to-fine search to deformable models. In this way we boost the detector accuracy but maintaing a similar speed.

Chapter 5

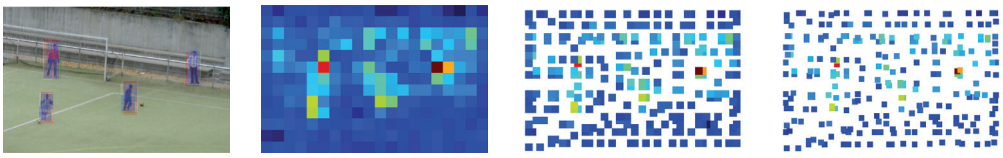
Deformable Coarse-to-Fine search

In this chapter we extend the coarse-to-fine procedure to deformable part models. The method is based on the observation that the cost of detection is likely dominated by the cost of matching each part to the image, and not by the cost of computing the optimal configuration of the parts as commonly assumed. Therefore accelerating detection requires minimizing the number of part-to-image comparisons. To this end we propose a multiple-resolutions hierarchical part based model and a corresponding coarse-to-fine inference procedure that recursively eliminates from the search space unpromising part placements. The method yields a ten-fold speedup over the standard dynamic programming approach and is complementary to the cascade-of-parts approach. Compared to the latter, our method does not have parameters to be determined empirically, which simplifies its use during the training of the model. Most importantly, the two techniques can be combined to obtain a very significant speedup, of two orders of magnitude in some cases. We evaluate our method extensively on the PASCAL VOC and INRIA datasets, demonstrating a very high increase in the detection speed with little degradation of the accuracy.

5.1 Introduction

In the last few years the interest of the object recognition community has moved from image classification and orderless models such as bag-of-words [105, 23, 59, 145] to sophisticated representations that can explicitly account for the location, scale, and deformation of the objects [38, 40]. By reasoning about geometry instead of discarding it, these models can extract a more detailed description of the image, including the object location, pose, and deformation, and can result in better detection accuracy.

A major obstacle in dealing with deformable objects is the combinatorial complexity of the inference. For instance, in the pictorial structures pioneered by Fischler and Elschlager [42] an object is represented as a collection of P parts, connected by springs. The time required to find the optimal part configuration to match a given



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

Figure 5.1: Coarse-to-fine inference. We propose a method for the fast inference of multi-resolution part based models. (a) example detections; (b) scores obtained by matching the lowest resolution part (root filter) at all image locations; (c) scores obtained by matching the intermediate resolution parts, only at location selected based on the response of the root part; (d) scores obtained by matching the high resolution parts, only at locations selected based on the intermediate resolution scores. A white space indicates that the part is not matched at a certain image location, resulting in a computational saving. The saving *increases with the resolution*.

image can be as high as the number L of possible part placements to the power of the number P of parts, i.e. $O(L^P)$. This cost can be reduced to $O(PL^2)$ or even $O(PL)$ by imposing further restrictions on the model ([38], 5.2.1), but is still significant due to the large number of possible part placements L. For instance, just to test for all possible translations of a part, L can be as large as the number of image pixels. This analysis, however, does not account for several aspects of typical part based models, such as the fact that useful object deformations are not very large and that, with appearance descriptors such as HOG [24], locations can be sampled in a relatively coarse manner.

The first contribution of this chapter, is a new analysis of the cost of part based models (Sect. 5.2.1) which better captures the bottlenecks of state-of-the-art implementations such as [24, 40, 148]. In particular, we show that the cost of inference is likely to be dominated by the cost of *matching each part to the image* rather than by the cost of determining the optimal part configuration. This suggests that accelerating inference requires minimizing the number of times the parts are matched.

Reducing the number of part evaluations can be obtained by using a *cascade* [130], a method that reject quickly unpromising object hypotheses based on cheaper models. For deformable part models two different types of cascades have been proposed (Sect. 5.2.1). The first one, due to Felzenszwalb et al. [40], matches parts sequentially, comparing the partial scores to learned thresholds in order to reject object locations as soon as possible. The second one, due to Sapp et al. [96], filters the part locations by thresholding marginal part scores obtained from a lower resolution model.

The second contribution of the chapter is a different cascade design (Sect. 5.2.2). Similar to [43, 96], our method is also coarse-to-fine. However, we note that, by thresholding scores independently, standard cascades propagate to the next level clusters of nearly identical hypotheses (as these tend to have similarly high scores). Instead of thresholding, we propose instead to reject all but the hypothesis whose score is *locally maximal*. This is motivated by the fact that looking for a locally optimal hypothesis at a coarse resolution often predicts well the best hypothesis at the next resolution

level (Sect. 5.2.2). As suggested in Fig. 5.1, and as showed in Sect. 5.2.2, 5.3, 5.4, this results in an *exponential saving, which has the additional benefit of being independent of the image content.* Experimentally, we show that this procedure can be ten times faster than the distance transform approach of [38, 42], while still yielding excellent detection accuracy.

Compared to using global thresholds as in the cascade of parts approach of Felzenszwalb et al. [40], our method does not require fine tuning of the thresholds on a validation set. Thus it is possible to use it not just for *testing*, but also for *training* the object model, when the thresholds of the cascade are still undefined (Sect. 5.5). More importantly, the cascade of parts and our method are based on complementary ideas and can be combined, yielding a *multiplication the speed-up factors* (Sect. 5.4.1). The combination of the two approaches can be more than two order of magnitude faster than the baseline dynamic programming inference algorithm [38] (Sect. 5.6).

5.2 Accelerating part based models

This section analyses the cost of inference in modern deformable part models (Sect. 5.2.1) and leverages on it to introduce a new efficient of coarse-to-fine detection scheme (Sect. 5.2.2).

5.2.1 Accelerating part based models

This section studies the cost of state-of-the-art models for object detection based on the notion of deformable parts. A deformable part based model, or pictorial structure as introduced by Fischler and Elschlager [42], represents an object as collection of P parts arranged in a deformable configuration through elastic connections. Each part can be found at any of L discrete locations in the image. For instance, in order to account for all possible translations of a part, L is equal to the number of image pixels. If parts can also be scaled and rotated, L is further multiplied by the number of discrete scales and rotations, making it very large. Since even for the simplest topologies (trees) the best known algorithms for the inference of a part based model require $O(PL^2)$ operations, these models appear to be intractable. Fortunately, the distance transform technique of [38] can be used to reduce the complexity to $O(PL)$ under certain assumptions, making part models if not fast, at least practical.

The analysis so far represents the standard assessment of the speed of part based models, but it does not account for all the factors that contribute to the true cost inference. In particular, this analysis does not predict adequately the cost of state-of-the-art models such as [40] for the three reasons indicated next. First, the complexity $O(PL^2)$ reflects only the cost of finding the optimal configuration of the parts, ignoring the cost of matching each part to the image. Matching a part usually requires computing a local filter for each tested part placement. Filtering requires $O(D)$ operations where D is the dimension of the filter (this can be for instance a HOG descriptor [24] for the part). The overall cost of inference is then $O(P(LD + L^2))$. Second, depend-

ing on the quantization step δ of the underlying feature representation, parts may be placed only at a discrete set of locations which are significantly less than the number of image pixels L. For instance, [39] uses HOG features with a spatial quantization step of $\delta = 8$ pixels, so that there are only L/δ^2 possible placements of a part. Third, in most cases it is sufficient to consider only *small deformations* between parts. That is, for each placement of a part, only a fraction $1/c$ of placements of a sibling part are possible. All considered, the inference cost becomes

$$O\left(P\frac{L}{\delta^2}\left(D + \frac{L}{\delta^2 c}\right)\right). \tag{5.1}$$

Consider for example a typical pictorial structure of [39]. The part filters are composed of 6×6 HOG cells, so that each part filter has dimension $6 \times 6 \times 31 = 1{,}116$ (where 31 is the dimension of a HOG cell). Typically the elastic connections between the parts deform by no more than 6 HOG cells in each direction. Thus the number of operations required for inferring the model is

$$(1{,}116 + 36)P\frac{L}{\delta^2} \tag{5.2}$$

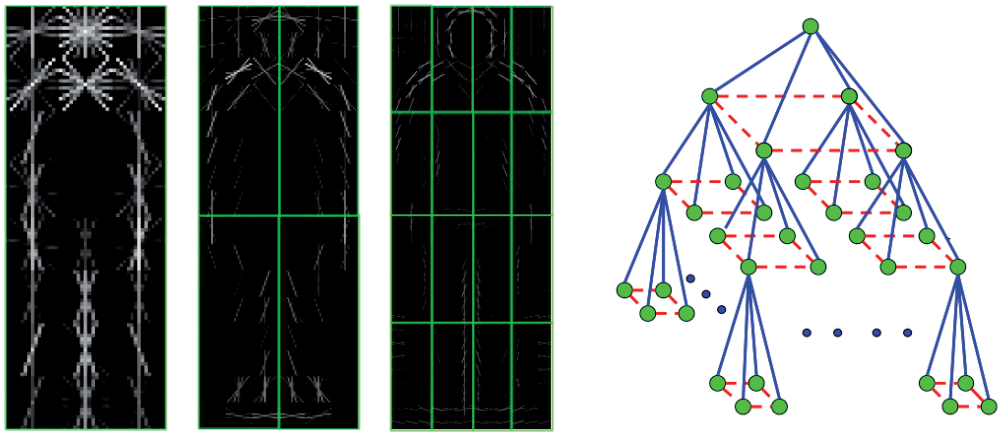
where the first term reflects the cost of evaluating the filters, and the second the cost of searching for the best part configuration. Hence the cost of evaluating the part filters is $1{,}116/36 = 31$ times larger than the cost of finding the optimal part configuration. The next section proposes a new method to reduce this cost.

5.2.2 Fast coarse-to-fine inference

This section proposes a new method base on a coarse-to-fine analysis of the image to speed-up detection by deformable part models. All the best performing part based models incorporate multiple resolutions [85, 148]. Therefore it is natural to ask whether the multi-scale structure can be used not just for better modeling, but also to accelerate inference.

Multiple resolutions have been used in the design of a cascade for deformable part models by [96]. Here we propose an alternative design based on a principle different from global thresholding [85, 86]. Consider the hierarchical part model of Fig. 5.2, similar to the one proposed by [148]. Our method starts by evaluating the root (coarser-resolution) filter at all image locations (Fig. 5.3). It then looks for the best placement of the root filter in, say, all 3×3 neighborhoods and propagates only this hypothesis to the next level. We call this procedure *Coarse-to-Fine* (CF) search.

Justification. The CF algorithm is justified by the fact that locally optimal placements of parts at a coarse resolution are often good predictors of the optimal part placements at the finer resolution levels. Fig. 5.4 shows the empirical probability that the CF procedure finds the same part locations as a globally optimal search procedure based on DP. As it can be seen, for detections with a threshold higher than -0.5 (which approximatively correspond to 80% recall), this probability is more than 70%,



(a) (b)

Figure 5.2: Hierarchical part based model of a person. (a) The model is composed of a collection of HOG filters [24] at different resolutions. (b) The HOG filters form a parent-child hierarchy where connections control the relative displacement of the parts when the model is matched to an image (blue solid lines); additional sibling-to-sibling deformation constraints are enforced as well (red dashed lines).

whereas suboptimal placements for hypotheses that have a small score are not detrimental to performance since those hypotheses would be discarded anyways. Sect. 5.6 gives more evidence of the validity of this assumption.

Lateral connections. The speed-up in our model is due to the fact that the placement of higher resolution parts is guided by the placement of lower resolution ones. This yields high computational savings, but makes inference more sensitive to partial occlusion, blurring, or other sources of noise.

This effect can be compensated by enforcing additional geometric constraints among the parts. In particular, we add constraints among siblings, dubbed *lateral connections*, as shown in Fig. 5.2 (b) (red dashed edges). This makes the motion of the siblings coherent and improves the robustness of the model. Fig. 5.5 demonstrates the importance of the lateral connections in learning a model of a human. Without lateral connections the model captures two separate human instances, but when the connections are added the model is learned properly. In Sect. 5.4 it will be shown that the increase in computational complexity due to the lateral connections is negligible. The next section starts by giving the formal details of the model.

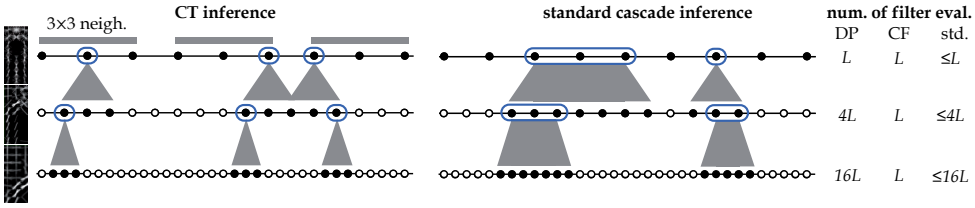


Figure 5.3: Coarse-to-fine cascade designs. *Left.* Our proposed CF cascade starts by matching the coarse resolution part at a set of L discrete locations, here denoted by circles along one image axis. It then propagates to the next resolution level only the best hypotheses (marked by a rounded blue box) for each 3×3 neighborhood. Thus, while there are four times as many locations at the next level, parts are always evaluated at only L locations (filled circles) regardless of the resolution, yielding to a constant saving. *Right.* By contrast, a standard cascade such as [37] propagates all locations whose score is larger than a threshold (rounded blue box). This (i) tends to propagate clusters of neighbor hypothesis at once as these tend to have similar score and (ii) results in a saving that depends on the image content.

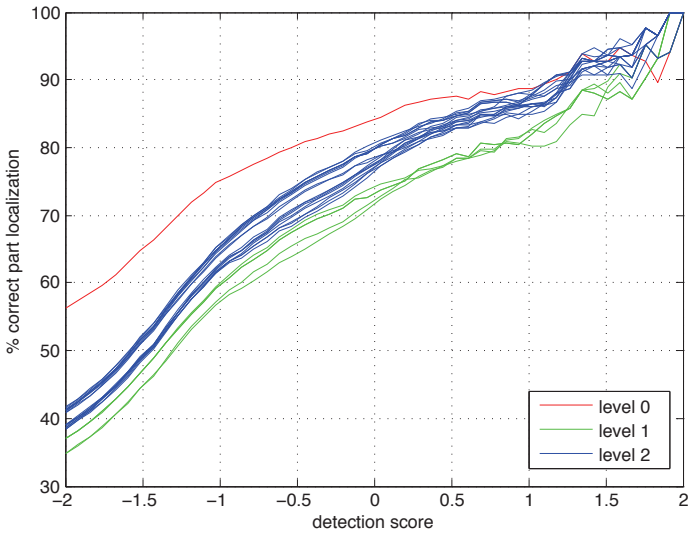


Figure 5.4: Coarse-to-fine predictions. The figure shows the probability that the coarse-to-fine search results in exactly the same part locations as the globally optimal DP algorithm for each part of the hierarchical model of Fig. 5.2. The probability is very high for highly scoring hypotheses (true positive) as desired.

(a) (b)

Figure 5.5: Effect of lateral connections in learning a model. (a) Detail of a human model learned with lateral connections active. (b) The same model without lateral connections.

5.3 Object Model

This section describes formally the model briefly introduced in Sect. 5.2.1. The model is a hierarchical variant of [40] (Fig. 5.2) where parts are obtained by subdividing regularly and recursively parent parts. At the root level, there is only one part represented by a 31-dimensional HOG filter [40, 24] of $w \times h$ cells. This is then subdivided into four subparts and the resolution of the HOG features is doubled, resulting in four $w \times h$ filters for the subparts. This construction is repeated to obtain sixteen parts at the next resolution level and so on. In practice, we use only three resolution levels in order to be able to detect small objects.

Let $\mathbf{y}_i$, $i = 1, \ldots, P$ be the locations of the P object parts. Each $\mathbf{y}_i$ ranges in a discrete set $\mathcal{D}_i$ of locations (HOG cells), whose cardinality increases with the fourth power of the resolution level. Given an image $\mathbf{x}$, the score of the configuration $\mathbf{y}$ is a sum of appearance and deformation terms:

$$S(\mathbf{y}; \mathbf{x}, \mathbf{w}) = \sum_{i=1}^{P} S_{H_i}(\mathbf{y}_i; \mathbf{x}, \mathbf{w}) + \sum_{(i,j)\in\mathcal{F}} S_{F_{ij}}(\mathbf{y}_i, \mathbf{y}_j; \mathbf{w}) + \sum_{(i,j)\in\mathcal{P}} S_{P_{ij}}(\mathbf{y}_i, \mathbf{y}_j; \mathbf{w}) \quad (5.3)$$

where $\mathcal{F}$ are the parent-child edges (solid blue lines in Fig. 5.2), $\mathcal{P}$ are the lateral connections (dashed red lines), and $\mathbf{w}$ is a vector of model parameters, to be estimated during training. The term S_{H_i} measures the compatibility between the image appearance at location $\mathbf{y}_i$ and the i-th part. This is given by the linear filter

$$S_{H_i}(\mathbf{y}_i; \mathbf{x}, \mathbf{w}) = H(\mathbf{y}_i; \mathbf{x}) \cdot M_{H_i}(\mathbf{w}) \quad (5.4)$$

where $H(\mathbf{y}_i; \mathbf{x})$ is the $w \times h$ HOG descriptor extracted from the image $\mathbf{x}$ at location $\mathbf{y}_i$ and M_{H_i} extracts the portion of the parameter vector $\mathbf{w}$ that encodes the filter for the i-th part. The term $S_{F_{ij}}$ penalizes large deviations of the location $\mathbf{y}_j$ with respect to the location of its parent $\mathbf{y}_i$, which is one resolution level above. This is a quadratic cost of the type

$$S_{F_{ij}}(\mathbf{y}_i, \mathbf{y}_j; \mathbf{w}) = D(2\mathbf{y}_i, \mathbf{y}_j) \cdot M_{F_i}(\mathbf{w}), \quad (5.5)$$

where i is the parent of j, $M_{F_i}(\mathbf{w})$ extracts the deformation coefficients from the parameter vector $\mathbf{w}$, and

$$D(2\mathbf{y}_i, \mathbf{y}_j) = \left[(2x_i - x_j)^2, (2y_i - y_j)^2\right] \tag{5.6}$$

where $\mathbf{y}_i = (x_i, y_i)$. The factor 2 maps the low resolution location of the parent $\mathbf{y}_i$ to the higher resolution level of the child. Similarly, S_P penalizes sibling-to-sibling deformations and is given by

$$S_{P_{ij}}(\mathbf{y}_i, \mathbf{y}_j; \mathbf{w}) = D(\mathbf{y}_i, \mathbf{y}_j) \cdot M_{P_{ij}}(\mathbf{w}). \tag{5.7}$$

In this case the factor 2 is not used in D as sibling parts have the same resolution.

In addition to the quadratic deformation costs, the possible configurations are limited by a set of parent-child constraints of the form $\mathbf{y}_j \in \mathcal{C}_j + 2\mathbf{y}_i$. In particular, $\mathcal{C}_j + 2\mathbf{y}_i$ is a set of $m \times m$ small displacements around the parent location $2\mathbf{y}_i$. The parameter m, bounding the deformations, is discussed again in Sect. 5.4 in the analysis of the CF inference procedure, and its impact is evaluated in the experiments (Sect. 5.6).

As in [40, 125] the model is further extended to multiple aspects in order to deal with large viewpoint variations. To this end, we stack N models $\mathbf{w}_1, \ldots, \mathbf{w}_N$, one for each aspect, into a new combined model $\mathbf{w}$. Then the inference selects both one of the n models and its configuration $\mathbf{y}$ by maximizing the score (5.3). Moreover, similarly to [125], the model is extended to encode explicitly the symmetry of the aspects. Namely, each model $\mathbf{w}_k$ is tested twice, by mirroring it along the vertical axis, in order to detect the direction an object is facing.

5.4 DP and CF inference

This section analyses in detail inference with the model introduced in Sect. 5.3. If the hierarchical model does not have lateral connections (i.e. $\mathcal{P}$ is the empty set in (5.3)), the structure is a tree and inference can be performed by using the standard DP technique. In detail, if part j is a leaf of the tree, let $V(\mathbf{y}_j) = S_{H_j}(\mathbf{y}_j)$, where we dropped for compactness the dependency of the score on $\mathbf{w}$ and $\mathbf{x}$. For any other part i define recursively

$$V(\mathbf{y}_i) = S_{H_i}(\mathbf{y}_i) + \sum_{j:\pi(j)=i} \max_{\mathbf{y}_j \in \mathcal{C}_j + 2\mathbf{y}_i} \left(S_{F_{ij}}(\mathbf{y}_i, \mathbf{y}_j) + V(\mathbf{y}_j)\right)$$

where $\mathbf{y}_j \in \mathcal{D}_j$ and $i = \pi(j)$ implies that i is the parent of j. Computing $V(\mathbf{y}_i)$ requires

$$|\mathcal{D}_i| \left(D + \sum_{j:\pi(j)=i} |\mathcal{C}_j| \right) \tag{5.8}$$

operations, where D is the dimension of a part filter and $\mathcal{C}_j$ is the set of allowable deformations given in Sect. 5.3. The terms $|\mathcal{C}_i|$ in the cost can be reduced to one by

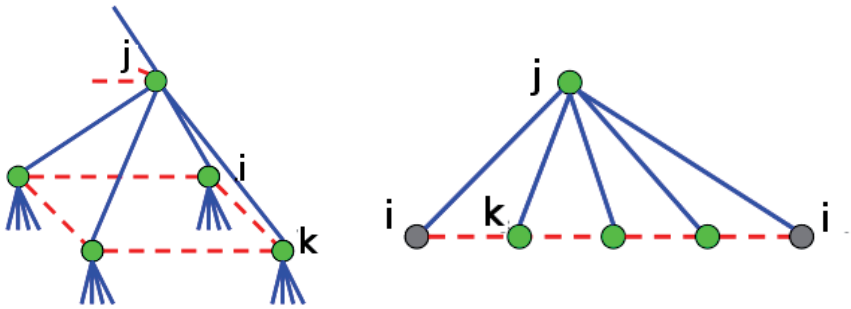


Figure 5.6: Part-to-part constraints. The loopy graph generated by the lateral connections is transformed into a chain by clamping the value $\mathbf{y}_i$ and then solved with DP.

using the distance transform of [38], but the saving is small since $|\mathcal{C}_i|$ is small to start with. Most importantly, the distance transform is applicable only in the case parts are tested at all locations, which precludes the use of a cascade.

DP with lateral connections. The lateral connections in Fig. 5.6 introduce cycles and prevent a direct application of DP. However, these connections form pyramid-like structures (Fig. 5.6(a)) that can be "opened" by clamping the value of one of the base nodes (Fig. 5.6(b)). In particular, denote with i the parent node, j the child being clamped, and k the other children. Then the cost of computing the function $V(\mathbf{y}_i)$ becomes

$$|\mathcal{D}_i| \left(D + |\mathcal{C}_j| \sum_{k:\pi(k)=i, k\neq j} |\mathcal{C}_k| \right), \tag{5.9}$$

which is slightly higher than (5.8) but still quite manageable due to the small size of $\mathcal{C}_i$.

CF inference. Despite the increased complexity of the geometry of a model with lateral connections, the cost of inference is still dominated by the cost of evaluating each part filter to each image location. This cost cannot be reduced by DP; instead, we propose to prune the search top-down, by starting the inference from the root filter and propagating only the solutions which are locally the more promising. Note that, instead of using a fixed threshold to discard partial detections as done by the part based cascade [37], here pruning is performed locally and adaptively. We now describe the process in detail, and estimate its cost.

First, the root part is tested everywhere in the image, with cost $|\mathcal{D}_0|D$. Note that, since the root part resolution is coarse, $|\mathcal{D}_0|$ is relatively small. Then non-maxima suppression is run on neighbors of size $m \times m$, leaving only $|\mathcal{D}_0|/m^2$ possible placements of the root part. For each placement of the root $\mathbf{y}_0$, the parts k at the

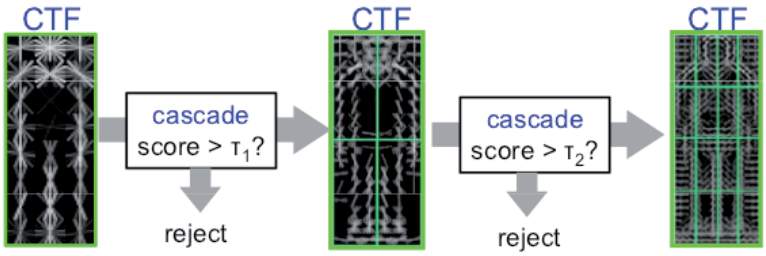


Figure 5.7: Combining CF with a cascade of parts. The score at each resolution level is determined by using the fast CF inference procedure. As soon as the score up to a certain resolution level has been computed, this is compared to a threshold to discard unpromising object locations quickly. The threshold is learned on a validation set as [37].

level below are searched at locations $\mathbf{y}_k \in C_k + 2\mathbf{y}_0$, which costs

$$\frac{|\mathcal{D}_0|}{m^2}\left(\sum_{k:\pi(k)=0}|\mathcal{C}_k|D + |\mathcal{C}_i|\sum_{k:\pi(k)=0,k\neq i}|\mathcal{C}_k|\right)$$

where i is the child clamped, as explained above, in order to account for the sibling connections. The dominant cost is matching the parts at $|\mathcal{D}_0|\ |\mathcal{C}_k|/m^2$ locations (if filters are memoized [37] the actual cost is a little smaller due to the fact that the same part location can be obtained from more than one root hypothesis). The process is repeated recursively, by selecting the optimum placement of each part at resolution r and using it to constrain the placement of the parts at the next resolution level $r+1$. In this way *each part is matched at most* $|\mathcal{D}_0|\ |\mathcal{C}_k|/m^2$ *times*, where $|\mathcal{C}_k|$ can be chosen equal or similar to m^2. This should be compared to the $|\mathcal{D}_k|$ comparisons of the DP approach, which grows with the fourth power of the resolution. Hence the computational saving becomes significant very quickly.

Note that, while each part location is determined by ignoring the higher resolution levels, the sibling constraints help integrating evidence from a large portion of the image and improve the localization of the parts.

5.4.1 Extensions

This section proposes two extensions of the CF inference procedure. First, our CF cascade can easily integrate global rejection thresholds analogous to the cascade of parts of Felzenszwalb et al. [37] resulting in a multiplication of the speed-up factors of our and their technique. Second, CF can be integrated with the standard DP algorithm by using it as a pre-filtering step to find a short-list of plausible object hypothesis where DP should be computed. This results in nearly exactly the same output as running DP globally but at a fraction of the cost.

CF and cascade of parts. The speed-up of the CF inference leverages on the topology of the part scores: hypotheses are pruned locally maintaining only the most promising ones, analogously to non maximal suppression. This is alternative and independent to pruning based on a global threshold on the classifier score, as in a standard cascade. As a consequence, the two approaches can be combined multiplying the speed-ups. In detail, as proposed by Felzenswalb et al. [37], one can learn thresholds to prune an object hypothesis based on the partial scores obtained by evaluating only a subset of the parts. In the experiments, a simplified version of this idea will be tested where pruning is applied after all parts at a given resolution levels have been evaluated. We call this CF+cascade, summarize it in Fig. 5.7, and report its empirical performance in Sect. 5.6.

CF and DP. The CF procedure recovers almost always the same object locations determined by a globally optimal method such as the standard DP algorithm. However, while the estimated location of the parts is often very similar too (Fig. 5.4, Sect.5.6), the equivalence is not perfect. In particular, the accuracy of the CF detector can be further improved by combining the two techniques at the price of a slightly reduced detection speed. The idea is to first estimate a small set of candidate object locations by using CF, and then computing the exact part placements, and hence the exact detection scores, by using DP only at those locations. Since CF estimates correctly the object locations in the vast majority of the cases and since its computed scores are fairly good by themselves, retaining up to a hundred object hypothesis per image is sufficient to reconstruct the output of the globally optimal DP nearly exactly. This idea is evaluated in Sect. 5.6.

5.5 Learning

This section describes in detail the learning of the model introduced in Sect. 5.3 and how to leverage on the fast inference methods of Sect. 5.4 to do so. Learning is needed to obtain the parameters $\mathbf{w}$ of the scoring function (5.3). This uses a variant of the latent structural SVM formulation of [141, 125], which is also very similar to the latent SVM method of [39].

Training uses a dataset of images and the corresponding bounding box annotations for an object category of interest. Each object bounding box is initially associated to the best matching location and scale $\mathbf{y}$ for the model. This is defined as the location $\mathbf{y}$ in the HOG coordinate space for which the root filter yields maximal intersection-over-union overlap score with the object bounding box. If there are multiple model components, one for each object aspect, the one with best overlap score is selected. This defines a set of positive examples $(\mathbf{x}_i, \mathbf{y}_i)$, $i \in P$, one for each object bounding box, where $\mathbf{x}_i$ denotes the corresponding image. All the other locations that yield an overlap score of less than a threshold T with all the object bounding boxes are used as negative examples $(\mathbf{x}_i, \mathbf{y}_i)$, $i \in N$ (in the case of the CF inference, one negative per root-level neighborhood is generated instead). Note that different $\mathbf{x}_i$ can refer to the same image as detections at different locations are considered independent by

learning. The alternative formulation is to maximize over negative detections [6], but this works better for us as it matches more directly the goal of optimizing AP by treating detections from different images uniformly.

From this construction, one obtains a number of negative samples far larger than the positive ones $|N| \gg |P|$, so that the data is highly unbalanced. Nevertheless, this was not found to be a problem in learning. This is due to the fact that, for the purpose of ranking, only the relative scores are important. While the imbalance may result in scores that are not perfectly calibrated for binary classification, but this does not affect ranking.

Note that the ground-truth locations $\mathbf{y}_i$ are effectively unknown and the procedure just described simply suggests an initial value. During training these are consider latent variables and gradually re-estimated. Training itself optimizes the latent SVM objective function

$$\begin{aligned} E(\mathbf{w}; \{\mathbf{y}_i, i \in P\}) = & \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i \in P} \max\{0, 1 - S(\mathbf{y}_i; \mathbf{x}_i, \mathbf{w})\} \\ & + C \sum_{j \in N} \max\{0, 1 + \max_{\mathbf{y}_j} S(\mathbf{y}_j, \mathbf{x}_i, \mathbf{w})\}. \end{aligned} \tag{5.10}$$

This trades off the usual quadratic regularizer $\|\mathbf{w}\|^2$ with a hinge-loss term for the positive samples, encouraging their score to be above 1 (the margin), and a corresponding term for the negative samples, encouraging their scores to be below -1. Note that the object location and pose $\mathbf{y}_j$ is maxed-out in the negative terms. This is possible without compromising convexity [39, 141]; on the other hand, the pose parameters $\mathbf{y}_i$, $i \in P$ must be kept constant as $\mathbf{w}$ is determined to make the energy convex. Subsequently, $\mathbf{w}$ is fixed and these parameters are re-estimated by maximizing $S(\mathbf{y}_i; \mathbf{x}_i, \mathbf{w})$ and the procedure is repeated. This is known as the Concave-Convex Procedure (CCCP) and is only guarantee to find a locally optimum solution [39].

Updating the latent variables. When the latent variables $\mathbf{y}_i \in P \cup N$ are optimized, the corresponding object locations are searched in a neighborhood of their initial values. In particular, for the negative examples the object location is kept fixed (or within a root-level neighborhood with CF) while the part locations are re-estimated. This is because the goal is to have in the energy function one negative example for each candidate image location. For the positive variables $\mathbf{y}_i$ instead, the object location is adjusted in order to better align the model to the corresponding object instance, including potentially switching the object aspect. This is done by estimating the best pose configuration for all locations and choosing the one which has best score among the ones that predict a bounding box with a sufficiently large overlap with the ground-truth object bounding box. For better accuracy, the bounding box is predicted as the tightest rectangle containing the highest resolution parts rather than the one containing the root filter only (that in our model has fairly low resolution). This also means that in rare cases there might be no location that, after the locations of the parts have been re-estimated, still fits the object bounding box, or that the one that does has lower score than the current setting of $\mathbf{y}_i$. This is handled below, accounting for the approximation due to the CF inference as well.

Constraint generation. The negative samples N are too many to be extracted and stored in memory. Instead, one starts with an empty set of negatives $N = \phi$ and then iteratively re-estimates $\mathbf{w}$ and searches the dataset for a batch of fresh examples N that are in margin violation (i.e. whose score is larger than -1), updates the model, and repeats. This procedure, which is equivalent to constraint generation [5] or mining of hard negatives [39], is guarantee to end in polynomial time provided that the set of support vectors (i.e. the examples violating the margin at the optimum) can fit in memory.

Using CF inference in training. Inference is used during training for two purposes: to estimate the optimal part layout $\mathbf{y}_i$, $i \in P$ for the example object instances (CCCP) and to obtain the most confusing part layout $\mathbf{y}_j$, $j \in N$ for the negative examples. The accelerated CF inference can be used to do this because, contrary to the part based cascade of [37], it does not have parameters to be learned. This fact can be used to substantially accelerate training too (see Sect. 5.6 Table 5.3).

While the CF inference has been found empirically to be quite reliable, it still returns approximated maximizers of the scoring function (5.3). From the viewpoint of the constraint generation procedure, this means that CF might not find all the harder negative constraints that could be determined by a globally optimal algorithm such as DP. However, this is unlikely to be a problem because the distribution of samples found by the CF procedure is the same in test as in training. In particular, if a negative example with a particularly high score was not found by the CF procedure during training, it is also unlikely that CF would find a similar hard negative during testing.

The estimation of the part locations for the positive latent variables $\mathbf{y}_i$ is more delicate. In this case, since the CF optimization is not necessarily optimal, it is possible that re-estimating the latent variables would actually *decrease* the objective (5.10), yielding an inconsistent algorithm. In practice, this can cause the latent variables to gradually drift away from a stable solution, learning a suboptimal model. Furthermore, it becomes difficult to devise a stopping criterion for the CCCP procedure.

This problem is fortunately easy to sidestep. Each time a new part layout for an object instance is re-estimated by means of the CF procedure, it is added to a pool of candidate layouts for that instance rather than assumed directly as the new value of the latent variable. Then the best layout is selected by computing the score (5.3) for all layouts in the pool. In this way, the energy is guaranteed to increase or at least stay constant every time the latent variables are re-estimated since in the worst case the previous assignment is reused, restoring the consistency of the estimation procedure.

5.6 Experiments

This section evaluates our method on three well known benchmarks: the INRIA pedestrians [24] and the 20 PASCAL VOC 2007 and 2009 object categories [35, 34].

method	det. time (s)	AP (%)
cascade [37]	0.23	85.6
CF	0.25	78.8
CF + siblings	0.33	84.0
CF + sib. + casc.	0.12	83.6

Table 5.1
Accuracy and detection speed on the INRIA data. The table reports the average precision and detection time in seconds for images in the INRIA dataset. *Cascade* denotes the part based cascade of [37]. *CF*, *CF + sibling*, and *CF + sib. + casc.* denote our coarse-to-fine inference scheme, respectively without sibling constraints, with sibling constraints, and combined with the cascade of [37]

.

Performance is measured in terms of false positive per window (FPPI) and Average Precision (AP) according to the PASCAL VOC protocol [35, 34].

For the VOC 2007 classes we use an object model with two components (aspects), for the VOC 2009, we use three components, while for the INRIA pedestrians we use a single one as using more did not help. The aspect ratio of each component is initialized by subdividing uniformly the aspects ratio of the training bounding boxes and taking the average in each interval.

5.6.1 INRIA pedestrians

Table 5.1 compares different variants of our coarse-to-fine (CF) detector with the part based cascade of [37] by evaluating the average detection time and precision for the INRIA pedestrian dataset. Our CF search algorithm is slightly slower than the part based cascade (0.33s vs 0.23s per image). However, the two methods are orthogonal and can be combined to further reduce the detection time to 0.12s, with just a marginal decrease in the detection accuracy as suggested in Sect. 5.4.1.

Fig. 5.8 compares the CF detector with other published methods in term of miss rate vs false positives per image (FPPI) rate. The CF detector obtains a detection rate of 88% at 1 FPPI, which is just a few points lower than the current state-of-the-art (91%), but uses only HOG features. In particular, due to the deformable parts and the CF inference, our detection rate is 10% better that the standard HOG detector while being much faster.

Effect of the neighborhood size m. Table 5.2 evaluates the influence of the neighborhood size m, which controls the amount of deformation that the model allows. Compared to Sect. 5.2.2 in which the same m is chosen at all resolution levels, here this parameter is fixed to $m = 3$ for the coarser resolution and changed in the range $m = 3, 5, 7$ for the higher resolutions, to evaluate absorbing larger deformations while

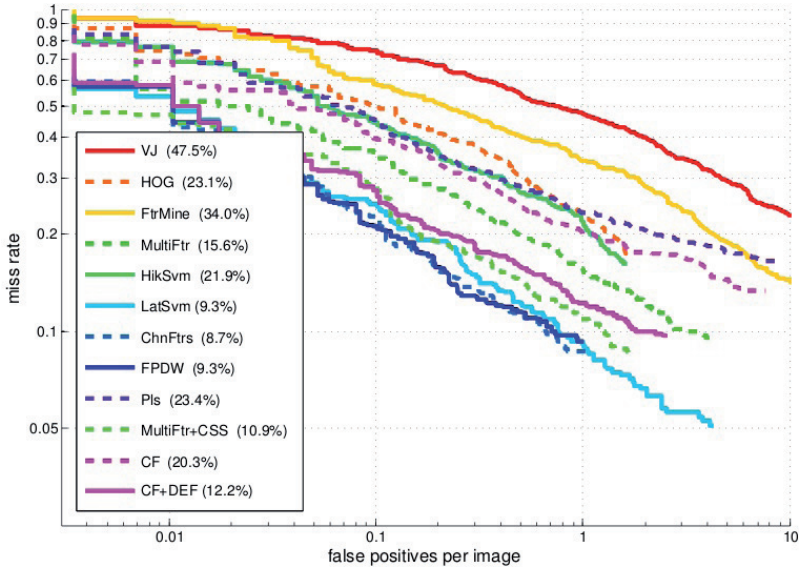


Figure 5.8: Comparison to the state-of-the-art on the INRIA dataset. The miss rate at 1 FPPI is reported in the legend. VJ [132], HOG [24], FtrMine [29], MultiFtr [137], HikSvm [69], LatSvm [40], ChnFtrs [28], FPDW [27], Pls [100], MultiFtr+CSS [134], CF is the coarse-to-fine search, CF+DEF is the method presented in this chapter.

still being able to detect multiple close instances of the objects. While inference slows down by increasing the deformation range m, this is unnecessary as the detection performance saturates at $m = 3$. Larger deformations do not change substantially the detection performance for this model, but greatly affect the inference time, which increases from 0.33s per image for $m = 3$ to almost 10s for $m = 7$.

This is probably due to two reasons. First, pedestrians are relatively rigid compared to humans in general pose. Second, although a deformation of one HOG cell in each direction for with respect to a part rest position ($m = 3$) may seem small, the actual amount of deformation must be assessed in relation of the size of the root filter. If the root filter is three HOG cells wide as in our setting, then a deformation of one HOG cell corresponds to a displacement that is as large as 33% of the object size, which is substantial.

Exact and CF detection scores. Fig. 5.9 shows a scatter plot of the detection scores obtained on the test set of the INRIA database, where the horizontal axis reports the scores obtained by DP (exact inference) and the vertical axis the scores obtained by the CF inference algorithm. The red line represents the ideal case, where the CF inference gives exactly the same results as DP. We distinguish two cases for

m	3	5	7
testing AP (%)	83.5	83.2	83.6
testing time [s]	0.33	2.0	9.3

Table 5.2
Effect of the neighborhood size m. On the INRIA Pedestrian dataset setting m to 3 is sufficient to obtain optimal performance. Increasing the value of m does not change substantially the AP, but has a negative impact on speed.

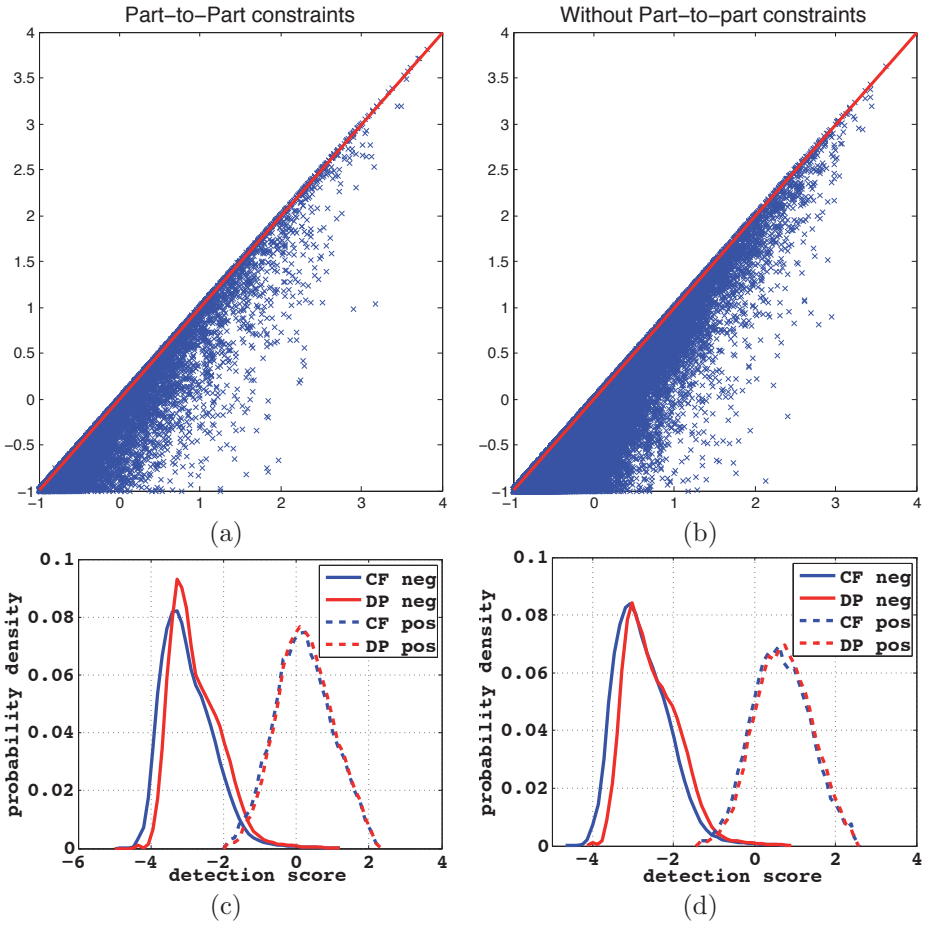


Figure 5.9: Exact vs coarse-to-fine inference scores. Scatter polt of the scores obtained by the exact (DP) and approximated (CF) inference algorithms: (a) with lateral constraints in the model, (b) without. (c,d) the corresponding distribution of the positive and negative hypothesis scores.

model	training method	time	testing AP (%) DP	CF
S_F	DP	20h	83.0	84.0
$S_F + S_P$	DP	22h	83.4	84.0
S_F	CF	1.9h	78.0	80.7
$S_F + S_P$	CF	2.2h	83.5	83.5

Table 5.3
Learning and testing a model with exact and coarse-to-fine inference. The table compares learning the model without lateral connection (S_F) and with lateral connections ($S_F + S_P$) and testing it with the exact (DP) or coarse-to-fine (CF) inference algorithm. For each case, training base on the DP or CF inference is also compared.

the analysis: (a) with lateral constraints and (b) without lateral constraints. We note two facts: First, in both cases the CF approximation improves as the detection score increases. This is reasonable because, if the object is easily recognizable, the local information drives the placement of the parts to optimal locations without much ambiguity. Second, in (a) the scatter plot is tighter than in (b), indicating that the lateral connections are in fact helping the CF inference to stay close to the ideal DP case. The same can be observed from the distribution of the scores (c) and (d).

Training speed and detection accuracy.

Table 5.3 evaluates the effect of using the CF and or the exact (DP) inference methods for training and testing the model. Using the CF inference method instead of the exact DP inference improves the training speed by an order of magnitude, from 20 hours down to just 2. This is because the cost of training is dominated by the iterative re-estimation of the latent variables and retraining, each of which requires running inference multiple times. Note that, differently from [37] which requires tuning *after* the model has been learned, our method can be applied *while* the model is learned.

A notable result from Table 5.3 is the fact that, for each training method (exact DP or CF) and model type (with or without lateral constraints), the accuracy never decreases, and in fact increases slightly, when the exact test procedure (DP) is substituted with the CF inference algorithm. This is probably due to the aggressive hypothesis pruning of the CF search which promotes less ambiguous detections. A second observation is that the lateral constraints are very effective and increase the AP by about 4–5% (depending on the training method). Note also that the improvement due to the lateral constraints is larger when training uses the CF inference algorithm.

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
BOW [124]	37.6	47.8	15.3	15.3	21.9	50.7	50.6	30.0	17.3	33.0	22.5
PS [40]	29.0	54.6	0.6	13.4	26.2	39.4	46.4	16.1	16.3	16.5	24.5
Hierarc. [148]	29.4	55.8	9.40	14.3	28.6	44.0	51.3	21.3	20.0	19.3	10.3
Cascade [37]	22.8	49.4	10.6	12.9	27.1	47.4	50.2	18.8	15.7	23.6	10.3
CF	27.9	54.8	10.2	16.1	16.2	49.7	48.3	17.5	17.2	26.4	21.4

	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mean**	**time**
BOW [124]	21.5	51.2	45.5	23.3	12.4	23.9	28.5	45.3	48.5	32.1	≈ 70
PS [40]	5.0	43.6	37.8	35.0	8.8	17.3	21.6	34.0	39.0	26.8	≈ 10
Hierarc. [148]	12.5	50.4	38.4	36.6	15.1	19.7	25.1	36.8	39.3	29.6	≈ 8
Cascade [37]	12.1	36.4	37.1	37.2	13.2	22.6	22.9	34.7	40.0	27.3	< 1
CF	11.4	55.7	42.2	30.7	11.4	20.9	29.1	41.5	30.0	28.9	< 1

Table 5.4
Detection AP and speed on the VOC 2007 test data. OUR METHOD HAS SIMILAR ACCURACY THAN OTHER STATE-OF-THE-ART METHODS BUT MUCH FASTER, BOTH IN TRAINING AND TEST.

5.6.2 PASCAL VOC

We compare our CF model with state-of-the-arts methods on VOC 2007 using the variant with sibling constraints. Table 5.4 shows that the classification accuracy of the CF detector is similar to the one of state-of-the-art methods which are about an order of magnitude or more slower. The CF detector is also compared to the part-based cascade of [37], which has a similar speed. However, the results reported in [37] are generated from detectors trained on the VOC 2009 data, which contains twice as many training images as found in the VOC 2007 data. Note that, as explained in Sect. 5.5, our results are obtained using the fast CF inference during training too, reducing the training process for each class to few hours.

Rigid vs. deformable model. Fig. 5.10 compares a rigid and a deformable model both using CF inference. The rigid model (*rigid CF*) is a simplified version of our deformable model, where each model resolution is a rigid block without moving parts. This model is very similar to the one presented in [85]. The gain obtained by the deformable model is around 6 AP points. This shows that the increment in the model complexity due to the introduction of local deformations is worth. This is even more relevant if we consider that the number of HOG filters to evaluate in the two cases is the same. The increase of computation is only due to the cost of evaluating the geometrical configuration of the parts as seen in Sect. 5.4. In practice, on a standard laptop computer, excluding computing the HOG features for an 640×480 image which requires around 0.8 seconds, detection with the rigid model requires around 0.25 seconds per model, while the detection with the deformable one requires around 0.3 seconds.

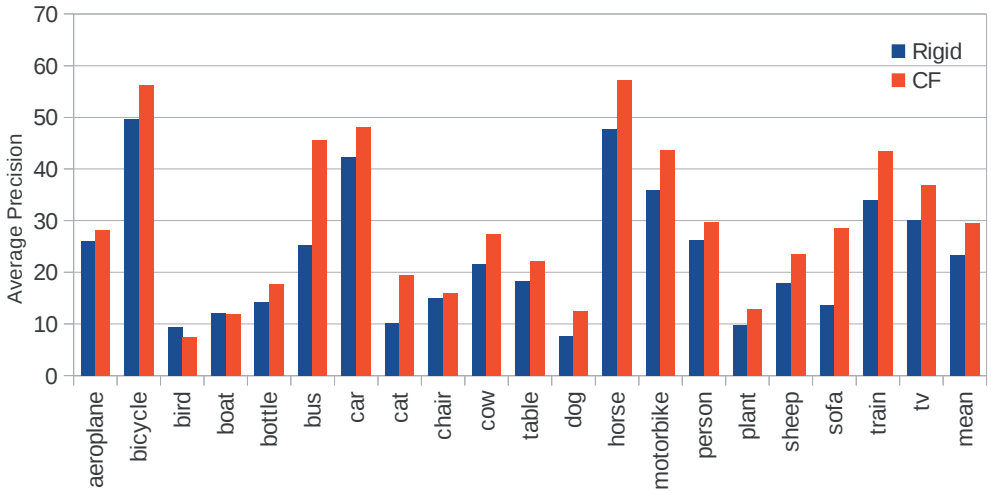


Figure 5.10: Performance of Rigid and Deformable models with CF inference on the PASCAL VOC 2007. The figure reports the average precision obtained for the twenty classes by the two models.

CF, DP, and their combination. Table 5.5 evaluates different inference methods used for both training and testing on the VOC 2007 dataset. Overall results are shown in the last 3 rows in terms of mean AP (*mean*), average number of HOG cells (in milions) that are involved in a filtering operation during inference (*HOG(M)*), and corresponding speed-up (*Speed-Up*). Note that the HOG evaluation dominates the cost of inference and in fact is shown here to correlate very well with the speed of each method. The speed of DP is used as reference and speed-ups are expressed relative to it (so DP has a speed-up of 1.0) Although for different classes results trend can vary, considering the mean of all classes gives a good estimation of the quality of the inference method. In general as expected, using a more expensive inference produces a better AP.

The most accurate detections are obtained by *Exact* (DP) inference for training and test. It obtains a mAP close to 32 points. This is very close to the state-of-the-art, and probably equivalent since it does not use any post-processing such as contextual models or bounding box refinement [39]. By using CF inference in test, the number of HOG cells that enter filtering is reduced from 66 millions to less than 5 millions, with a corresponding speed-up of more than one order of magnitude. However, the mAP decreases slightly. A trade-off between exact and approximate inference is given by the combination of CF and DP (labelled *CF+Ex*), as described in Sect. 5.4.1. Applying DP to the best 100 or 10 best hypotheses selected by CF strategy results in nearly optimal accuracy and a speed-up factor of either 8 or 11 times compared to standard DP.

As shown in Fig. 5.11, substituting Exact inference by CF inference during training produces a loss in AP that varies from 0.5 to 1.5 points. However training using exact

Train	Exact				CF			
Test	Exact	CF+Ex(100)	CF+Ex(10)	CF	Exact	CF+Ex(100)	CF+Ex(10)	CF
plane	32.2	32.2	32.6	28.1	29.8	30.2	30.8	27.9
bicycle	58.4	58.1	54.5	56.2	58.6	58.4	54.2	54.4
bird	10.7	10.7	10.7	7.4	6.4	6.5	6.5	10.2
boat	13.9	14.1	12.5	12	16	15.9	15.5	16.1
bottle	19.0	19.1	17.8	17.8	16.3	16.4	14.2	16.2
bus	49.8	50.0	49.1	45.5	52.6	52.5	51.1	49.7
car	52.0	51.7	49.1	48.2	51.2	50.8	49.4	48.3
cat	23.1	23.1	22.0	19.5	17.1	17.0	18.1	17.5
chair	20.3	19.3	17.7	16.0	19.2	19.2	17.2	17.2
cow	29.4	29.7	28.3	27.5	28.6	28.2	28.2	26.4
table	29.3	29.2	28.3	22.2	23.6	24.3	24.6	21.4
dog	13.5	13.5	13.7	12.4	12.0	12.0	12.7	11.4
horse	59.6	59.3	57.8	57.3	57.7	57.7	56.3	55.7
mbike	44.5	44.3	43.2	43.6	43.1	43.0	42.5	42.2
person	29.7	29.5	26.4	29.7	31.7	31.6	28.3	30.7
plant	12.9	12.2	12.5	12.9	12.4	12.4	12.4	11.4
sheep	26.2	26.2	26.1	23.5	25.2	25.1	23.8	20.9
sofa	29.6	29.8	28.5	28.5	26.2	28.0	27.9	29.1
train	44.0	44.2	45.2	43.5	43.0	43.2	44.0	41.5
tv	39.2	39.5	39.4	36.9	36.8	36.6	35.6	30.0
mean	31.9	31.8	30.8	29.4	30.4	30.5	29.7	28.9
HOG(M)	66.5	8.12	5.75	4.72	66.5	8.12	5.75	4.72
Speed-Up	1.0	8.1	11.6	14.1	1.0	8.1	11.6	14.1

Table 5.5

Performance of different image scan modalities on VOC 07. The table compares different inference methods used for training and testing. As shown in the first row of the table models have been trained using either *Exact* inference or *CF* inference. Test is effectuated using *Exact* inference, *CF+Ex(100,10)* where the best 100 or 10 hypothesis of CF inference are refined using exact inference, *CF* inference. Notice the different level of performance and cost needed from different configurations.

inference is more than 10 times slower than training using CF inference, which can signify a jump from days to hours. Interestingly, the highest loss is found when the model is trained with CF end tested with Exact. This is because training with CF systematically avoids some configurations that are not reachable due to the greedy search. Then, during test, using *Exact* inference these configurations are found again and can produce false positives

CF and cascade of parts. This paragraph evaluates the combination of our CF inference with a threshold-based filtering, as explained in Sect. 5.4.1. In order to simplify the visualization of the results, we set the two thresholds $\tau_1 = \tau_2 = \tau$. Setting independently the optimal value of the two thresholds can further improve the speed-up. Fig. 5.13, 5.13 and 5.14 report for all VOC classes the trade-off between detection speed (taking as reference exact inference computed using DP) and average precision achieved by varying τ.

The shape of the curves is similar in classes with comparable average precision. Fig. 5.12 shows that classes with relatively high average precision have also an excellent behavior in pruning hypotheses, resulting in a speed-up of more than two orders of magnitude with marginal decrease in detection accuracy. Also, all classes reach

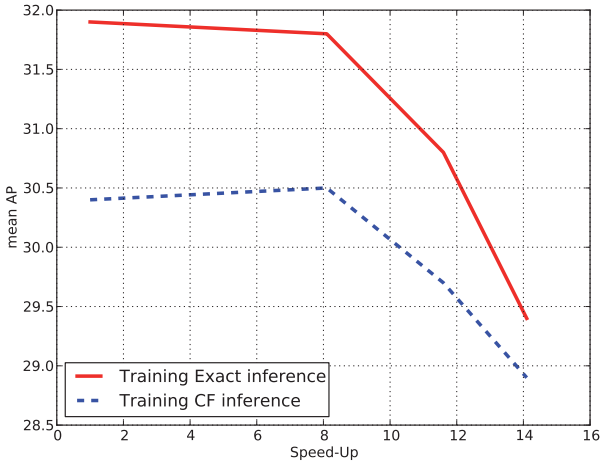


Figure 5.11: AP vs Speed-Up for different inference configurations in training and test. The figure compares the performance of using Exact and CF inference in the training of VOC 2007. The best configuration can be selected depending on the application and the computational resources available.

a maximum speed-up above 200 times the original DP approach. Fig. 5.13 shows classes with an average precision between 0.2 and 0.3. In this case some classes still have a high speed-up with minimum loss in accuracy, while others have a reduced limited speed-up, but still above 100 times.

Finally, for the classes with a low detection rate (Fig. 5.14), the speed-up is limited because the detection accuracy decreases relatively quick when increasing the pruning threshold. Abrupt jumps in the curves are due to the low rate quantization used in the evaluation criteria of the VOC 2007 [35]. This analysis shows that increasing the quality of the detector we can also expect a higher margin of gain in speed.

VOC 2009. Table 5.6 evaluates the CF inference on the PASCAL VOC 2009 [34]. Since the test annotations for this dataset are not publicly available and repeated evaluation on it is discouraged by the challenge organizers to prevent over-fitting we repeated only the most significant experiments. In particular, the table reports only the results of the CF inference alone, which, as shown before, can be further improved by combination with DP or the cascade-of-parts technique. The conclusions are analogous to the 2007 data.

Although in the challenge the detection time is not evaluated, all methods have a computational cost much higher than ours; *UOCTTI* and *MIZZOU* are all based on pictorial structures and at least HOG features, which makes their complexity at least one order of magnitude higher. *OXFORD* bases its method on bag of words and SVM kernel learning, which are computationally more expensive than the linear

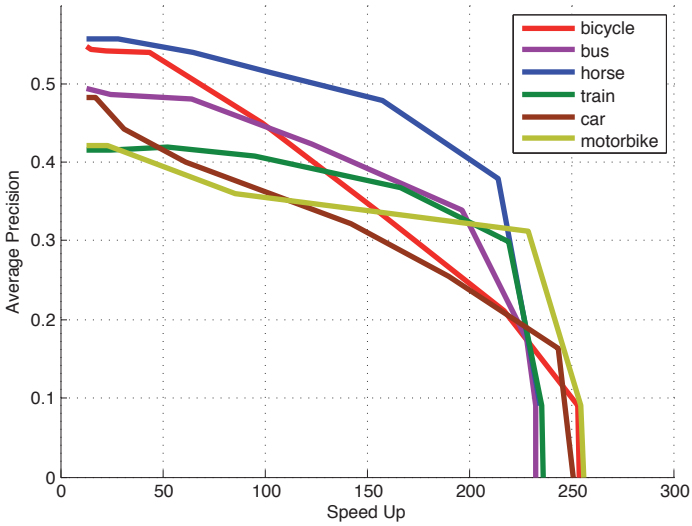


Figure 5.12: Speed-Up vs AP for classes with high AP. The figure reports the average precision vs speed-up (over the exact DP inference algorithm) for the CF detector combined with a pruning cascade on VOC 2007.

models used in pictorial structures. Therefore, even with accelerating techniques like a cascade filtering in one case or branch-and-bound in the other case, they are almost certainly much slower than our approach.

5.7 Conclusions

In this chapter we have presented a method that can substantially speed-up object detectors based on multi-resolution deformable part models. We have shown that, for this type of models, the cost of detection is likely to be dominated by the cost of matching each part to the image, rather than by the cost of finding the optimal configuration of the parts. Based on this observation, we have proposed a new hierarchical model that, combined with the coarse-to-fine search, can dramatically speed-up detection by reducing the number of times parts are matched to the image. While the speedup that can be obtained is similar to the one of the part based cascade [37], this method does not require the learning of thresholds or other parameters which simplify its use during the training of the model; moreover, the speed of detection does not depend on the image content.

Finally, we have proposed two extensions of the CF inference. In the first one, we have showed that our method is orthogonal to the part-based cascade and it can be combined with the latter to obtain speedups of up to a factor 200 in some cases. In the second one, we have added to the CF inference a final stage where the best hypotheses found are further refined using DP. Since DP is applied in this way only to

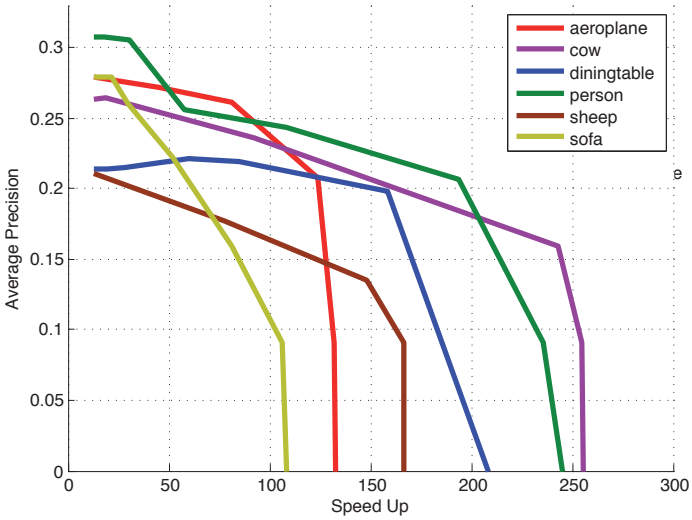


Figure 5.13: Speed-Up vs AP for classes with medium AP. The figure reports the average precision vs speed-up (over the exact DP inference algorithm) for the CF detector combined with a pruning cascade on VOC 2007.

a very small number of locations, we have obtained results almost identical to globally optimal inference, but in a fraction of time.

In the next chapter we will show how to adapt the coarse-to-fine deformable model for a real application.

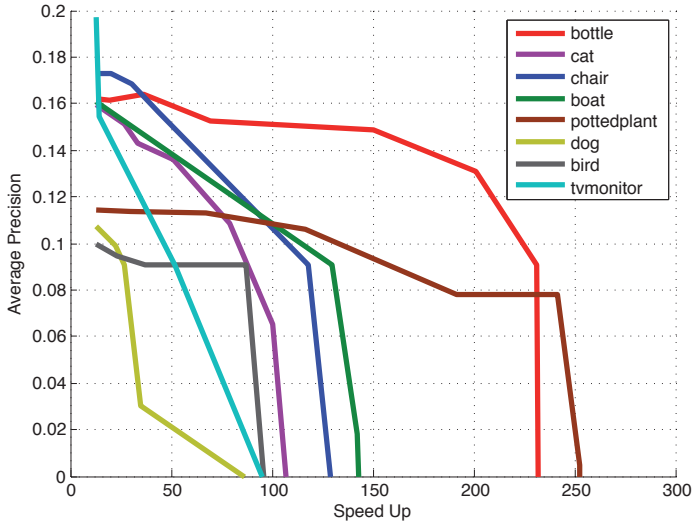


Figure 5.14: Speed-Up vs AP for classes with low AP. The figure reports the average precision vs speed-up (over the exact DP inference algorithm) for the CF detector combined with a pruning cascade on VOC 2007.

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
OXFORD	47.8	39.8	17.4	15.8	21.9	42.9	27.7	30.5	14.6	20.6	22.3
UOCTTI	39.5	46.8	13.5	15.0	28.5	43.8	37.2	20.7	14.9	22.8	8.7
MIZZOU	11.4	27.5	6.0	11.1	27.0	38.8	33.7	25.2	15.0	14.4	16.9
CF	41.3	45.5	10.9	13.6	18.3	44.0	33.3	24.2	11.7	19.1	14.9
CF+Ex(100)	41.5	46.6	11.5	15.3	20.0	44.3	35.9	23.9	13.1	20.7	15.9

	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mean**
OXFORD	17.0	34.6	43.7	21.6	10.2	25.1	16.6	46.3	37.6	27.7
UOCTTI	14.4	38.0	42.0	41.5	12.6	24.2	15.8	43.9	33.5	27.9
MIZZOU	15.1	36.3	40.9	37.0	13.2	22.8	9.6	3.5	32.1	21.9
CF	12.4	37.2	42.5	22.1	10.3	20.6	18.3	39.4	31.8	25.6
CF+Ex(100)	13.4	40.4	44.1	22.4	10.7	23.4	21.9	43.4	34.3	27.1

Table 5.6

Detection AP on the VOC 2009 test data. We compare our method with the official results of the VOC 2009 [34]. Using much less computation in both training and test, CF inference achieves results comparable to the state-of-the-art.

Chapter 6

Pedestrian Detection on a moving vehicle

Reliable Pedestrian detection is still an open issue; Sufficient requirements for applying pure image-based detection to this problem are: (i) an excellent precision-recall trade-off to avoid false alarms but still detecting as much as possible pedestrians; (ii) a very fast computation for quick reactions to dangerous situations. Recent algorithms based on deformable templates have shown reasonable detection performance, but they are computationally too expensive for real-time performance. In this chapter we specialize the deformable coarse-to-fine detector to a real application: pedestrian detection for driving assistance. The complete system is a hierarchical multi resolution part-based model with additional features specialized to the problem domain. As result, the system, due to the local deformations induced by the parts is able to achieve state-of-the-art detection accuracy, but, in contrast to most of the other part-based methods, it uses a fast coarse-to-fine inference that guarantee more than one order of magnitude speed-up. Altogether, using also GPU computation, the proposed system is suitable, in terms of both accuracy and speed, for real-time pedestrian detection using only images.

6.1 Introduction

Driving assistance is a growing area of research that involves many different disciplines, from mechanics to computer science. The fields of application go from very specific and rule based systems, like Antilock Brake System and air-bags, nowadays present in almost every commercial vehicle, to very challenging and complex tasks, like following the correct path and avoiding obstacles and accidents in an uncontrolled scenario.

In this chapter we deal with a very specific, but fundamental task, which is pedestrian detection using a single camera mounted on the vehicle. Being able to detect

pedestrian as well as other objects using only a normal camera sensor would be a great technological advance. In fact cameras, compared with other sensors like laser scanner or ultrasound systems, are very economic and this would permit a vast deploying of this technology especially to low class vehicles.

The computer vision community in the last years has developed excellent methods that achieve high accuracy and can deal with challenging images and complex objects categories [124, 40]. However this level of accuracy has been reached at the cost of renouncing to real-time performance, due to the higher computational cost of complex features [124] or complex object models [40]. Restricting the detection to the specific task of pedestrian detection on a moving vehicle, some additional speed-ups can be achieved. For instance, in normal conditions, the upper part of the image always contains sky and the search for pedestrians can be avoided, producing a save in time as well as in number of false positive detections. Also, more sophisticated techniques to reduce the number of location to scan using specific knowledge of the problem can be used [53]. Still, considering that the time for computing image in a high-level PC is in the order of one minute for [124] and around 10 seconds for [40], even with the previous enhancements, they are too far for real-time performance. Furthermore, these techniques work properly when the object to detect has a relatively high resolution. This condition is not satisfied for pedestrian detection from a moving vehicle, where it is very important for avoiding accidents a quick detection of far pedestrians, which have low resolution.

In this chapter we propose a method that model the pedestrian using a hierarchy of parts at multiple resolutions. We substitute the expensive dynamic programming based search for the object model, with a faster coarse-to-fine search that gives similar detection results in a fraction of time. Also, during the coarse-to-fine search we introduce an additional reasoning about small objects that normal detectors can not detect. In practice, scores of detections from small objects where high resolution features are not available, are made comparable to full-resolution detections, adding an additional bias. In the experimental results (sect. 6.4) we show that this improves the capability of detecting small pedestrians, and for this specific problem it improves quite a lot the overall performance. We adapt and optimize the general detection algorithm for the specific task of human detection from a moving vehicle. In particular, we show that using the coarse-to-fine algorithm the most expensive procedure is the feature computation. By pre-computing the features in GPU and discarding the image regions where the pedestrians are not likely to be, we show that our final system is suitable in terms of both accuracy and speed, for real and fast applications.

The structure of the chapter is the following. In section 6.2 we present a brief overview of our detector system. More in detail, in section 6.2.1 we explain how to make the image scan faster using a CtF representation. Also, as for pedestrian avoidance it is fundamental to discriminate with very high accuracy, in section 6.2.2 we briefly explain the CtF representation to local deformations. Additional "resolution" feature to specifically detect small pedestrians is explained in section 6.2.3, while the computation of the features in GPU is described in section 6.3. In section 6.4 we evaluate the performance of different components and configurations of our system

on a database with images appositely taken from a moving vehicle. Finally, in section 6.5 we conclude this chapter explaining how this system, specifically adapted for detecting pedestrian from a moving vehicle can reach sufficient accuracy as well as speed for a reliable system based only on visual cues.

6.2 Our System

The architecture of our system for pedestrian localization is illustrated in Fig. 6.1. Given an image, we pre-compute the HOG features of the image at different resolutions, obtaining a pyramid of HOGs. Then, given a object template or model, the pyramid is scanned at all resolutions in a coarse-to-fine way, finding the locations that are the most similar to the template, and therefore more likely to contain the sought object, in this case a pedestrian. These locations are further processed applying a non-maximum suppression (NMS) to the overlapping ones. The remaining locations, represent the detected pedestrians. In the following subsections we will explain the most relevant components of our system. For the HOG computation, as in chapter 5 we use the implementation [40] which is an improvement over the standard HOG features [24]. For NMS we rank the detection scores and we select the 1000 best detections for a greedy clustering based on the pascal overlapping criteria [35].

6.2.1 Coarse-to-Fine search

The standard procedure to find the sought object in an image consists of evaluating the similarity between the object model and the image features at every location and scale in the image. Considering that we use a model learned with linear SVM, (see subsection 6.2.4), the similarity measure is the scalar product of the object model M and the corresponding pyramid feature H at location $\mathbf{x} = (x, y, s)$, where x, y are the coordinate of the window center and s is its scale. Therefore, the standard search is the correlation between M and each level of the HOG pyramid:

$$D(\mathbf{x}) = \langle M, H(\mathbf{x}) \rangle \,. \tag{6.1}$$

Generally, to obtain more discriminative detectors, a finer feature resolution is needed, which produces as well an object model with higher resolution. This implies that the vectors of the scalar product in Eq. (6.1) can be of the order of thousand dimensions. Therefore, the complete scan over positions and scales is very expensive and often it is the computational bottle-neck of the entire system. We use the coarse-to-fine search (CtF) proposed in chapter 4 to save computation but still obtain results very similar to the complete search. In the rest of this section we reformulate the CtF search in a recursive formulation that is more expressive and synthetic for our specific problem.

The key idea is to decompose the search over multiple resolutions: from coarse and then fine. The coarse resolution has less locations where to scan and the scalar product is faster to compute because the vectors have less features. However, few coarse features are not enough for good discrimination of the model. For this reason

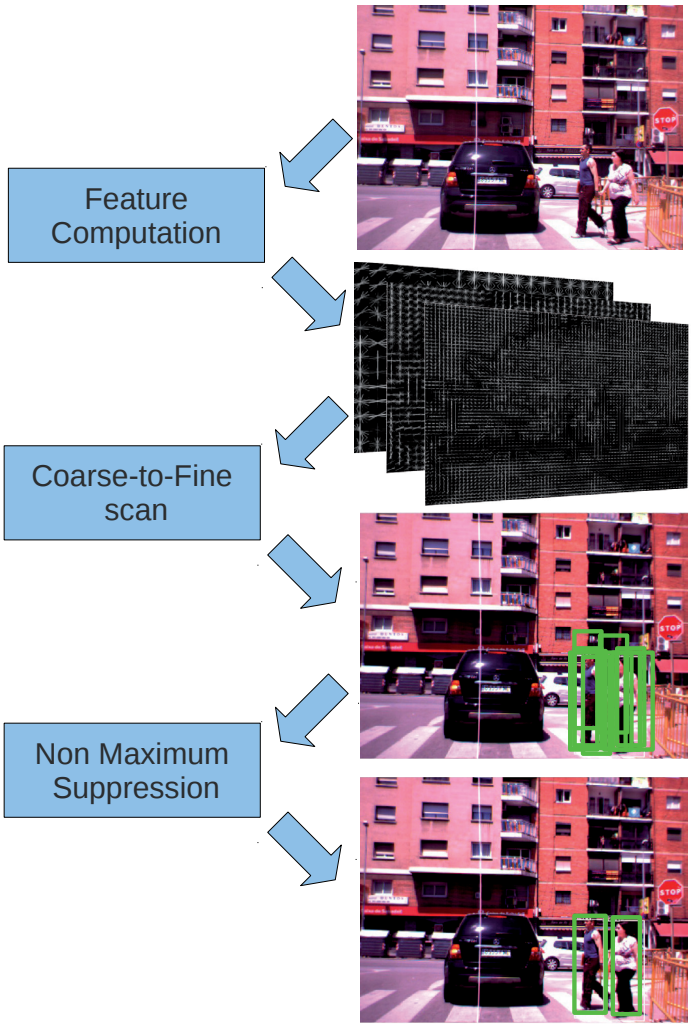


Figure 6.1: Overview of our detection system: From the image a pyramid of HOG features is computed and given as input for the CtF procedure, which finds the best detections. These detection are subsequently filtered using a non maximum suppression procedure.

adding finer resolutions improves performance but increases the computational cost. In practice we can think the CtF procedure like a progressive refinement search: the coarse object representation is used to roughly and quickly find the object locations and then successive local refinement are applied with the next model resolutions.

The score of the multiple resolution detector is computed as sum over resolutions r of the object model M_r with the corresponding features H:

$$D(\mathbf{x}) = \sum_{r=1:R} \langle M_r, H(\mathbf{x}_r) \rangle \,. \tag{6.2}$$

Considering a model resolutions with scale ratio equal to 2 (i.e. each model M_{r+1} doubles the previous model resolution M_r), we set :

$$\mathbf{x}_r \rightarrow \mathbf{x}_{r+1} = 2\mathbf{x}_r \tag{6.3}$$

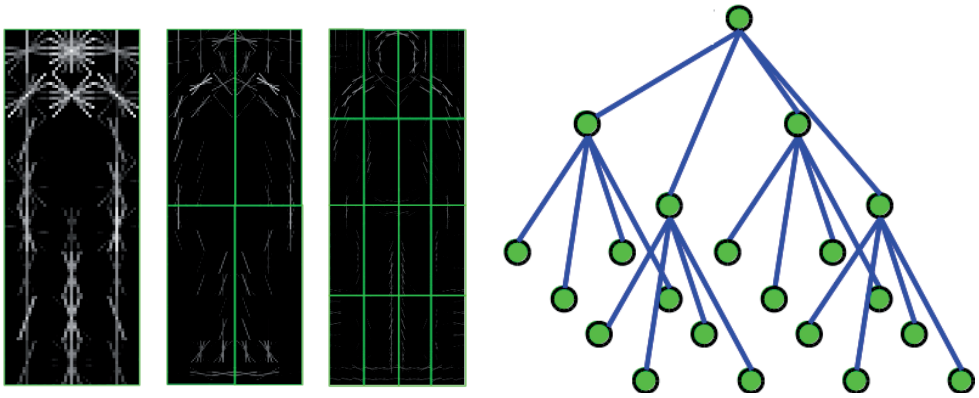
to impose that the locations x_r for all resolutions r represent the same image position. An example of a 3-resolutions object model for pedestrians is shown in Fig. 6.3(a).

In the coarse-to-fine procedure, the search starts computing the score of the coarse model everywhere in feature space $\mathbf{x}_1$. After that, the score locations are clustered into local neighborhoods, and like in NMS only the highest score for each neighborhood is selected. The selected hypotheses are propagated to the following resolution model M_2 using Eq. (6.3). Now again, a local neighborhood is build around every hypothesis. Notice, that after the first resolution level, the local neighborhoods do not cover anymore all the possible locations of the image, but only a small fraction around the hypotheses. This produces a high computational saving because the scalar product has a computational cost that increases 4 times when doubling resolution, but with the CtF procedure only a small fraction of locations is actually computed: those that are close to the hypotheses. The procedure is then recursively repeated for all model resolutions.

6.2.2 Coarse-to-Fine search with deformations

In this section we extend the previous recursive formulation of the CtF search to deformable parts models. Adding moving parts allows the detector to better adapt to local object deformations that are produced by view point changes or articulated movements, like limbs movements in the case of pedestrians. Unfortunately adding deformation to the object model supposes a huge increment of computation because for each location the best object parts configuration should be found. Previous methods reduce the computational cost of finding the best object parts configuration using distance transform, assuming squared deformation cost [40]. This procedure reduces the cost of matching parts, but still, all locations have to be evaluated computing the costly scalar product between features and object model.

To reduce this cost, we use the CtF procedure extended to deformable models presented in chapter 5. Now, in the object model, each resolution level is further divided into parts as shown in Fig. 6.2(a) (green boxes). Specifically, the coarse



(a) (b)

Figure 6.2: Multiresolution deformable model. (a) Example of a multiple resolution deformable part model: each part is a collection of HOG filter at different resolution. (b) The HOG filters form a father-child hierarchy where connections control the relative displacement of parts.

(a) (b)

Figure 6.3: Detail of a pedestrian head and torso model. (a) Model learned *without local deformation* (b) Model learned *with local deformation*. The second model has clearer edges due to the local deformations.

representation of the model has only one part. The middle resolution is divided into P local parts, and moving to the finer resolution, each of these parts is again decomposed into P sub-parts in a recursive way, creating a tree-like structure as shown in Fig. 6.2(b). The object score at a certain location $\mathbf{x}$ is now computed as:

$$D(\mathbf{x}) = \sum_{r=1:R} \sum_{p=1:P^{r-1}} \left\langle M_{r,p}, \left[H(\mathbf{x}_{r,p}), d_x^2, d_y^2\right]\right\rangle \tag{6.4}$$

where the feature vector is now extended with the deformation features d_x, d_y that represent the displacement of a part p with respect to its father. In the CtF procedure with the new object model, the initial hypothesis produced by the model $M_{1,1}$ is propagated to the next resolution level generating P new hypotheses for the sub-parts as shown in Eq. 6.5.

$$\mathbf{x}_{r,p} \rightarrow \mathbf{x}_{r+1,1}, \mathbf{x}_{r+1,2}, ..., \mathbf{x}_{r+1,P-1} \tag{6.5}$$

The procedure is recursively repeated until covering all parts of the model. Each new hypothesis $\mathbf{x}_{r+1,i}$ is found at double resolution and with a certain offset $\mathbf{o}_i$ due to the relative sub-part location:

$$\mathbf{x}_{r+1,i} = \mathbf{o}_i + 2\mathbf{x}_{r,p}. \tag{6.6}$$

As before, from each hypotheses a local neighborhood is used to find the maximum local score, which is the hypothesis for the next resolution level. However, while in the rigid CtF algorithm, the local search was used to align the entire object model with the image features, now this procedure is done locally for each part, simulating local deformations.

In Fig. 6.3 a comparison of an object model learned with and without local deformations is shown. The model learned without local deformations is quite fuzzy, while the model learned with local deformations has stronger edges that make the model more discriminative.

6.2.3 Small Objects

An important requirement in pedestrian detection for driving assistance is to detect low resolution pedestrian instances. Due to perspective distortion, low resolution pedestrians correspond to pedestrians far from the vehicle. Detecting far pedestrian gives enough time for a proper action to avoid collision, which is the first aim of a driving assistance system. However, when the number of pixels representing an object is low, the ability to recognize an object is highly reduced (see section 6.4).

Concretely, for HOG computation, the number of pixels needed to build the features should always be the same to avoid under-sampling effects. This means that, when an object instance in an image has very low resolution, the corresponding HOG features can not be properly extracted and the object would be missed.

When using multiple resolutions, like in our case, a small instance object do not have fine resolution features, but it still has the coarse representation. So, in the CtF

procedure, the search for the object can be extended to those scales that contains very small objects. In this case the high resolution features are filled with zeros. This allows the method to detect small objects. Unfortunately, the missing features will produce detections with a score that is unbalanced (lower) with respect to full resolution detections because a part of the descriptor is artificially filled with zeros.

To overcome this problem, we extend [84] to our multiresolution object model. We add to the feature descriptor a further binary feature for each resolution level, which represents whether, in the considered example, the corresponding resolution is available. Now the score is computed as:

$$D(\mathbf{x}) = \sum_{r=1:R} \sum_{p=1:P^{r-1}} \left\langle M_{r,p}, \left[H(\mathbf{x}_{r,p}), d_x^2, d_y^2, h_r\right]\right\rangle \tag{6.7}$$

where h_r is a binary variable that is enabled, when the corresponding HOG features $H(x_{r,p})$ are missing and therefore set to 0. In this way, h_r acts similarly to a bias term that makes scores of detections generated without high resolution features comparable to full resolution detections. We evaluate the advantage of this solution in the experimental results section. For easy understanding in the rest of the chapter we will refer to these additional features as "resolution" features.

From the computational point of view the increment of computation due to the use of the resolution feature is limited to the scan of the coarser resolutions of the model at the finer resolutions of the feature pyramid, which is actually much smaller than applying the detector to a bigger image size.

6.2.4 Learning

The learning procedure of our system is based on latent SVM [141, 125, 40]. Given a set of input data $\{x_1, \ldots, x_n\}$ and the associated labels $\{y_1, \ldots, y_n\}$, we find a parameter vector w of a function y that minimizes the regularized empirical risk:

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i y(x, w)). \tag{6.8}$$

In our problem the input data x_i is the set of features extracted from the HOG pyramid H defined in the previous section and associated to an image region, while the output data y_i is a binary label indicating whether the object is present in the region. We introduce a latent variable $\mathbf{k}$ that represents the relative position of each child part with respect to its father. Considering the local position of each part allows the detector to learn a more discriminative model during learning and also to obtain a better alignment of the object model with the image data. The estimated output y is computed as:

$$y(x, w) = \max_{\mathbf{k}} D_w(\mathbf{x} + \mathbf{k}) \sum_{r=1:R} \sum_{p=1:P^{r-1}} \max_{\mathbf{k}_{r,p}} \left\langle M_{r,p}, \left[H(\mathbf{x}_{r,p} + \mathbf{k}_{r,p}), d_x^2, d_y^2, h_r\right]\right\rangle \tag{6.9}$$

From Eq. (6.7) we see that w corresponds to the flattened and concatenated version of all the parts $M_{r,p}$ of our object model.

In contrast to normal SVM optimization, y is no longer linear in w due to the maximization on $\mathbf{k}$, therefore the empirical risk is no longer convex, and standard optimization techniques can not be used. Instead, we use the iterative procedure proposed in [40] and already used in chapter 4 and 5, where learning is divided into two iterative steps: the optimization of w with $\mathbf{k}$ fixed for the positive examples and the estimation of the best $\mathbf{k}$ using the computed w.

The optimization of w given $\mathbf{k}$ is convex and is computed using parallelized stochastic gradient descend [150]. The estimation of $\mathbf{k}$ with the current object model w is computed from Eq. 6.7.Instead of computing the exact maximization of Eq. 6.7, we apply the CtF procedure. Although there is not guarantee of the final performance of the approximate learning, we have empirically seen in chapter 5 that it produces close to optimal results with a reduced computation.

6.3 Adapting the system for real-time computation

So far we have described a general detection system that has a faster image scan due to the coarse-to-fine procedure and it is able to detect small pedestrians due to the introduction of "resolution" features. However, as we will show in the experiments (section 6.4), although the system can reach excellent performance, it is not fast enough for real-time application. In this sense, adapting the general framework to our specific task of pedestrian detection from a moving vehicle, we can obtain some additional speed that can lead to almost real-time performance. In contrast to normal sliding window methods, where the main cost of a detection is produced by the image scan, in the coarse-to-fine procedure, the image scan is reduced by more than 10 times and therefore the dominant cost is the feature computation (see table 6.1). In this regard, we propose to take advantage of the high computational power provided by the parallel computation available from the Graphics Processing Unit (GPU). Also, as we want to detect pedestrian from a moving vehicle, although there is no steady background ,we can still take advantage from the fact that the camera is mounted in a fixed location on the vehicle. Knowing that, we can discard from detection the regions of the image where the pedestrian is not likely to be.

6.3.1 Accelerating HOG computation by GPU

Graphic Processing Units (GPU) beside their general use in computer graphics, they can be also used for improving the speed of general algorithms using their high capability of parallel computing. By using high-level programming interfaces, such as CUDA, OpenCL and DirectCompute, an algorithm designed in parallel manner can take advantages of the SIMT (Single-Instruction, Multiple-Threads) architecture, in which a block of threads can be executed concurrently on a streaming multiprocessor. On the other hand, the scalable programming model enables high performance thread

scheduling. Generally, an optimized program can achieve more than 10 times speedup comparing with any CPU implementation.

Inspired by similar implementations [89, 109], we propose a fast HOG feature computation in CUDA. Our implementation follows the design of[40]. In contrast to the HOG [24], our implementation use both contrast insensitive and sensitive orientation channels, but substitutes the multiple normalizations of the HOG cells with additional normalization features. In this way, the final descriptor is smaller than the original HOG (31 dimensions instead of 36), but more discriminative.

The pipeline of HOG computation is divided into five steps: gradient computation, spatial aggregation, normalization, feature assembling and image rescaling. We see that the main limitation of GPU computing is data transfer. Exchanging data with host memory frequently might lose the time saved by parallel computing. Thus, an efficient way is to keep all steps executed in GPU. We transfer an image into the global memory (off-chip memory) on GPU. After rescaling, the gradient map is computed together with the spatial aggregation. We found that some parallel nature in the improved HOG can be utilized. By caching some variables in the shared memory (on-chip high-speed cache), it can reduce global memory access. In each thread block, we use 4×18 float variables in shared memory to cache the cell aggregation. Each pixel can contribute to 4 histograms surrounding the cell where the pixel is located in. The caching trick is also used in the feature normalization and assembling, where $(b+1) \times (b+1)$ float variables are stored in the shared memory, where b is the size of thread block.

Note that the number of threads in a block is constrained by the resource allocated to the block. In our case, we can assign each computation unit to one thread within every step. In addition, we exploit the texture memory to compute a fast bilinear interpolation directly in hardware, such that the image rescaling can be performed in an easy and very fast way. The final output from the GPU is the feature pyramid. It can take the place of original CPU version seamlessly in our detection system. In figure 6.4 show a comparison of the HOG computation of a 40-level feature pyramid in GPU and in CPU. We can see that GPU works much faster in large scales.

6.3.2 Region of interest

In contrast to general object detection, pedestrian detection from a moving vehicle has some prior about the camera location and this can be used to further speed-up the final detector. In [97] for instance, the 3D location of the road (assuming it a plane) is estimated from stereo images to reduce the pedestrian search only on this plane, therefore reducing the search cost. In contrast, we do not make any assumption about the road structure and do not try to estimate its 3D location. We take a simpler and conservative approach. Considering that the camera orientation in the vehicle is fixed and the maximum level of steep variation a road can present is limited, we can avoid to search for pedestrian in the upper part of the image. More in detail, we can discard the superior one third of the image without any loss in recall, but with a corresponding saving of 30% the total amount of computation.

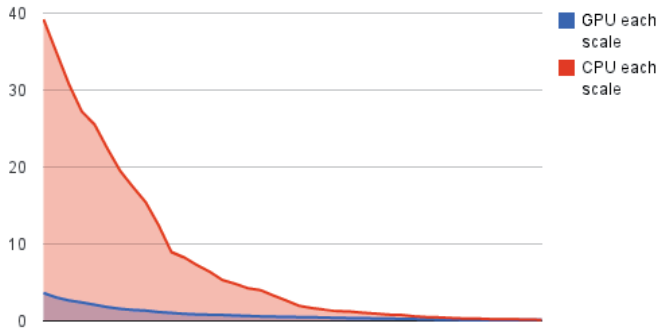


Figure 6.4: CPU and GPU HOG computation. The two algorithms were tested in the same computer with Intel Xeon CPU and NVidia GTX560 graphics card. The test image is 640×480. The computation starts from scale 0 to scale 40 and take into account all the steps mentioned in section 6.3.1. The largest speedup is in scale 0, where CPU HOG needs 39.18 milliseconds while GPU HOG only takes 3.62 milliseconds. Note that, for fair comparison, we consider also the image transfer time between host and device to GPU HOG computation.

6.4 Experiments

We evaluate our method on the CVC02 dataset [47], which is a dataset specific for pedestrian detection in the context of driving assistance. It consists of pedestrians taken in the range from 0 to 50 m, which correspond approximatively to 70×140 to 12×24 pixels bounding boxes. The training set consists of 1016 cropped humans with corresponding vertical mirror, for a total of 2032 images. The testing set consists of 250 urban images containing pedestrians.

6.4.1 Comparison with [47]

In Fig. 6.5 we compare our detector in terms of detection-rate (DR) versus false-positive-per-image (FPPI) with different configuration of the simplified HOG based on SVM learning detector (SHOG+SVM) proposed in [47]. For this experiment we use our CtF configuration with deformations and "resolution" features activated. Our detector has a quite relevant higher DR than the other when the working point is set to high precision (< 1 FPPI). Interestingly, thanks to the CtF procedure the methods is also faster than the ones from [47]. Following the explanations from the original manuscript, the fastest configuration from [47] takes more than 10 s for detecting pedestrian in an image of the database (size 640×480 pixels). This is in line with our results using a complete image scan (see table 6.1 rows 1 and 2). In contrast, our method in our machine (Intel Pentium Xeon 2.67Mhz using only one core) takes less than 1 s to compute the HOG pyramid in CPU and less than 0.1 s in GPU, while the image scan takes less than 0.5 s depending on the specific configuration.

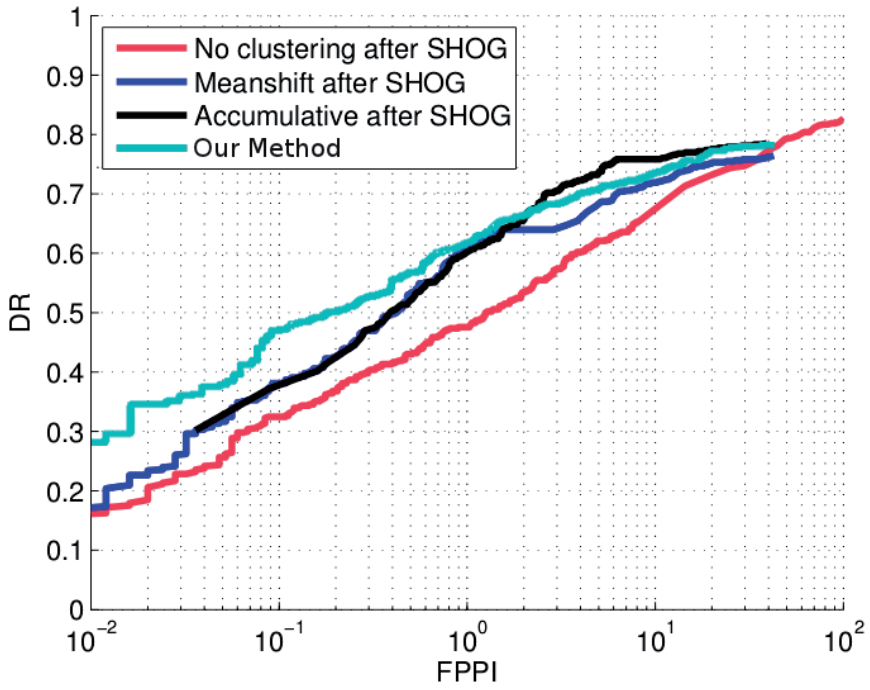


Figure 6.5: Detection Rate versus False Positive Per Image on the CVC02 database. Our method corresponds to the CtF detector with deformations and "resolution" features activated (corresponding to the 7^{th} row of Table 6.1). The other curves are obtained from [47]: they are based on SHOG, a variant of HOG features, linear SVM and different kinds of non-maximal suppression.

image scan		windows		im. size		model			Time	Time CPU		Time GPU	
Ex	CtF	Sm.	Feat.	×1	×2	Rig.	Def	AP(%)	Scan	feat	Total	feat	Total
×				×		×		29.8	7.6	0.8	8.4	0.077	7.7
×				×			×	34.8	9.9	0.8	10.7	0.077	10.0
	×			×		×		29.8	0.14	0.8	0.94	0.077	0.22
	×			×			×	33.7	0.25	0.8	1.05	0.077	0.33
	×	×		×		×		59.4	0.24	0.8	1.04	0.077	0.32
	×	×		×			×	63.1	0.4	0.8	1.2	0.077	0.48
	×	×	×	×			×	65.0	0.4	0.8	1.2	0.077	0.48
	×	×			×	×		66.2	0.9	3.8	4.7	0.35	1.25
	×	×			×		×	73.1	1.3	3.8	5.1	0.35	1.65
	×	×	×		×		×	73.4	1.3	3.8	5.1	0.35	1.65

Table 6.1

Average precision (AP) and detection time with different configurations of our system. Image scan can be *Ex* if searching at all possible locations and correspond to standard sliding windows in case of using a rigid template, or *CtF* if using the faster procedure explained in section 6.2.1 for rigid templates and in section 6.2.2 for deformable templates. *windows* correspond to evaluating also the windows where the high resolution model is not present without adding the corresponding feature *Small* or adding the corresponding "resolution" feature *Feat.* *image size* is the size of the image used for detection that can be the original ×1 of 640 × 480 or the double ×2. Finally the object *model* can be *rigid* or *deformable*.

6.4.2 Rigid versus Deformable models

In table 6.1 we evaluate the quality of a detector configuration in terms of average precision (AP) and Time. AP is the averaged values of the precision obtained by the detector in a precision-recall curve drawn applying the detector the complete test set. *Time* is the average time needed for: (i) the image scan, i.e. searching the object in the image, (ii) the feature pre-compuation in both CPU and GPU and, (iii) the overall computation of a frame.

The first tow row of table 6.1 compare a model using a rigid template (as explained in sect. 6.2.1) and a model using deformable parts (as explained in sect. 6.2.2). In both cases we use a complete search to avoid any possible problem due to the CtF scan. Learning local deformations through the object parts is useful to better align the object model with the image. This translate into an improved detector recall because also misaligned object can be correctly detected. In practice, the overall detector performance using the deformable model is increased of almost 5 points with respect to the rigid model with a relatively small increment of computational cost (from 8.4 to 10.7 seconds).

6.4.3 CtF versus Complete search

We want to evaluate the performance of the CtF search compared with the complete search (i.e sliding window), both in terms of speed and accuracy. The first 2 rows of table 6.1 have exactly the same configuration as row 3 and 4. The only difference is in the scan procedure. The first rows use complete search while the second CtF.

While the Average Precision (AP) of the two methods is comparable, CtF procedure scans an image in much less than 1 s while standard sliding windows takes up to 10 s. Note that for the rigid model (row 1 and 3) complete and CtF search give exactly the same AP. In case of deformable model, using the CtF approximation produces a loss in AP of 1 point. Still, the improvement compared to the rigid model is quite high, as well as the gain in time.

6.4.4 Use of "resolution" features

For pedestrian detection from a moving vehicle, the detection of small examples (far from the vehicle and therefore at low resolution) is fundamental. This is proved in the last part of table 6.1. Here it is possible to notice the high AP improvement when adding the searching for objects at smaller size and the corresponding "resolution" features as explained in section 6.2.3. The AP for rigid model rise from 29.8 to 59.4. For deformable models the gap is almost similar: from 33.7 to 63.1 adding only the search of small objects and 65.0 adding also the "resolution" feature. In term of time, the search at a smaller resolution of the object adds some overhead in the image scan (from 0.14 to 0.24 for the rigid model and from 0.25 to 0.4 for the deformable). However, this is minor than the one introduced by doubling the image resolution. Doubling the image resolution would generates a slow-down of around 4× in the image scan as well as in the feature computation. In contrast, the search of the object a smaller sizes is done only at low resolution, which is not very expensive but good enough to find some additional detections.

Even though the average precision of the method is highly increased by using "resolution" features, the overall performance is still affected by small objects that in certain cases are missed. We evaluate this testing the method on resized images at double resolution. This configuration obtain the best AP gaining around 10% over the normal image size AP. However, the detection time per frame also highly increased, up to 5 s.

6.4.5 GPU for feature computation

From table 6.1 and Fig. reffig:GCC we can see that the use of GPU has a stable speed-up in the feature computation of around 10 times. Compared with other implementations like [89, 109] this is not very high. However, we should remind that in contrast to other methods, in our current implementation, only the feature computation is computed in GPU, while the image scan is still done in CPU. This is mainly

due to a design choice drawn from different reasons. First, implementing everything in GPU, and especially the recursive part of the CfF algorithm would be quite complex, long and prone to errors. Second, leaving the scan of the image in the CPU can be useful in case of further developments where multiple classes (i.e. cars, bicycles, etc...) should be detected at the same time. In this case, the algorithm can be easily extended to use multiple cores (nowadays quite common in standard PCs), each one for each class. Thus, the final detection speed would remain the same. This can not be exploited computing everything in GPU. Finally, due to the CtF procedure the time spent in the feature computation is much more relevant than in the complete scan. For instance, using the complete search with a rigid model, the CPU and GPU overall time for detection are respectively 8.4 and 7.4 s. The relative difference is not very relevant. In contrast, using CtF search, the total time in CPU for computing an image is around 1 second while for GPU is 0.22 s, which means an overall speed-up of 5 times. Also, as introduced in section 6.3.1, we can a further improvement in speed computing only the region where pedestrian should appear. In the CVC02 dataset, using a selected region of the lower two thirds of the image produces an additional 33% speed-up while maintaining the same degree of accuracy. Using these speed-ups we can obtain a final system that is able to run at 5 frames per seconds and have a considerable AP of 65. Note that all the reported times are computed in our PC (Xeon 2.7 GHz CPU) using a single CPU. Taking advantage of multiple CPUs nowadays available in almost every PC could give a further boost of the overall speed form 2 to 6 times.

6.4.6 Comparison with other GPU-based detectors

A general comparison with other methods based on different implementations and different machines is quite difficult. However, one advantage of our work compared with previous pedestrian detectors is the use of deformations in conjunction with the CtF search, that give a high boost in performance and a relatively small increment of computation. In this sense, we can compare our detector with those proposed in [89, 109], which are based on a fast GPU implementation of the Dalal and Triggs detector [24]. In the INRIA dataset [24], our detector with deformable model at 1 FPPI has a recall of 0.8 while the rigid model of [109] has a recall of 0.6 and the fastHOG of [89] has a recall of 0.5.

6.5 Conclusions

In this chapter we have presented a new framework for fast pedestrian detection in the context of driving assistance. The framework is based on the combination of recent state-of-the-art techniques for fast and accurate object detection in still images.

We have evaluated our system in a dataset specific for pedestrian detection from a moving vehicle and we have shown that it. is competitive with the state-of-the-art in both speed and accuracy. This is due to the use of (i) a CtF procedure for fast image

scan, (ii) the use of model parts to simulate local deformations (iii) the evaluation of detections with missing resolutions (iv) the introduction of an additional feature that balances out scores with missing resolutions (v) the feature computation which is the bottleneck of the system is quickly computed in GPU.

Finally, we propose a real system that can run at 5 fps and has a detection accuracy superior to other real-time systems. The speed-up of the application can be further increased estimating the ground-plane for filtering hypotheses. Also combining the CtF with a cascade, has done in chapter 5 can produce a further increase of speed.

Chapter 7

Conclusions and Perspectives

In this final chapter we summarize the main contributions of this thesis. Also we give a perspective of the multiple lines of research that can start from the work developed in this thesis.

7.1 Summary and Contributions

In this thesis we have proposed the use of hierarchical multiresolution models for object detection. In chapter 1 we have introduced the topic of object detection, underlining the possible applications, the main challenges and the differences from other similar topics. A broad overview of the most important techniques used for each stage of object detection, as well as a complete list of the most interesting methods presented in the last years, are presented in chapter 2.

In chapter 3 we have proposed our first algorithm, which is a cascade of classifiers at multiple resolutions. The key contribution of the approach lies in the evidence that using features at different resolution, from coarse-to-fine is very convenient for object detection. In particular, using a cascade of HOG features at multiple resolutions, we have developed a pedestrian detector that has two different reasons of speed-up: the reduction of the classifier cost due to the initial use of coarse feature resolution and a reduced set of detection hypotheses due to the high stride used for the image scan. Also using model resolutions we can use the same pyramid of features for both the multiple scale search and the levels of the cascade.

In chapter 4, using the same multiresolution model, we have proposed a new algorithm for the image scan, where the commonly used cascade of classifiers is substituted by a coarse-to-fine search. In this way, in addition to the advantages previously introduced, we use an alternative approach to filter hypotheses. This approach is based on the progressive refinement of the object search that is driven by the different resolution models from coarse-to-fine. The new algorithm does not need to learn pruning

thresholds because the reduction of hypotheses is effectuated performing a form of local non maximal suppression that maintains, for each location, only the most relevant hypothesis. It is empirically shown that, even without tuning any classifier-specific threshold, the method has performance similar to the commonly used cascade in both speed-up and accuracy. Also, the method is tested on the 20 classes of the VOC 07 database; the results are comparable with the complete search in terms of accuracy, but more than 10 times faster. Finally, the speed-up introduced by the coarse-to-fine search is constant and independent of the image content, which can be an advantage for real-time requirements.

Further and relevant extensions of the method have been presented in chapter 5. There, we have shown that the coarse-to-fine search can be extended to deformable models, which can be better aligned to the image and thus improve the detector accuracy. In the chapter we have proved that, for deformable models, assuming the object deformation to be bounded, the dominant cost for detection is the classifier evaluation and not the cost due to finding the best parts configuration. In this sense, the coarse-to-fine procedure avoids to evaluate the object model everywhere in the image and it permits the deformable model to gain a speed-up similar to the rigid-template, as the computation of the part configuration is negligible, but with a better accuracy due to the moving parts. With small changes, the coarse-to-fine procedure can be applied also in the estimation of the latent variables and the search of hard negatives during training, which is the most costly part of a latent SVM training. Finally, we have shown that the coarse-to-fine procedure can be easily used in conjunction with other methods like a cascade of classifiers for a further multiplication of the speed up to 2 orders of magnitude, or with a final refinement using exact dynamic programming for improved accuracy. All of these experiments are validated on the INRIA pedestrian dataset as well as on the 20 classes of the VOC 2007 and 2009 datasets.

Chapter 6 is an example of practical application of the algorithms previously presented. In particular, we specialize our algorithm to the task of pedestrian detection for a camera mounted in a moving vehicle. In this case, the application requires to be real-time and with a high detection accuracy. These requirements are accomplished assuming that the upper part of the camera cannot contain any pedestrian because it always pointing at the sky, and by speeding up the HOG computation using GPU parallel processing.

7.2 Future Work

The main contribution of this thesis lies on the introduction and the evaluation of new methods to reduce the cost of searching for an object in an image, using hierarchies of multiresolution features. In this regards the natural next step is to use the gained speed for the further improvement of detectors precision and recall. Hereunder we list some of the possibilities to enhance the object detection accuracy.

Fast BOW Recent works have shown that the state-of-the-art for object detection

is currently obtained by the combination of deformable part-based template models together with a bag of words representation [18]. Unfortunately, as explained in chapter 2, the bag of words representation is, in general, orders of magnitude slower than (deformable) template matching. Hence, naively combining our fast approach with a heavy bag of words would not follow the spirit of this work regarding a parsimonious use of computation. A better solution is to find some approximation of the bag of word approach that can still lead to fast performance. A possibility is to use some fast quantization of groups of HOG cells to create the words. In this way, no additional computation for the features is necessary and the following histogram computation, once the words are computed, can be quite fast.

Increase parts invariance. During this work we realized that, for excellent detection results, are not necessary very complex and computationally expensive features. Instead, even quite simple HOG features, if correctly aligned, can achieve very impressive performance citefelzenszwalb10object,everingham09voc. In current methods, object parts can only move spatially. If we remove this limitation, allowing the search of the parts for a given location over scales and rotations, we believe we can contribute to an even better alignment between object model and image, thus less false positives and better discriminative capability. For instance, when a car is seen in a 3/4 view, due to perspective, the rear wheel is smaller than the frontal. Considering that the wheel is a part of the object model, if the wheel can move over scales, it can perfectly fit the smaller wheel. Otherwise, the part appearance should be able to detect small and big wheels, which produces a less discriminative model. The same reasoning can be effectuated for rotations.

Better deformation model Still considering the alignment problem, object parts can be considered as a rough quantization of a continuous deformation model, where each pixel of the object model can be deformed to better adapt to the image. Without changing our model representation that is based on HOG features, we can reduce the parts size and increase their number to get a better approximation to the continuous deformation. However, using the father-child part relationships presented in chapter 5, this representation would not work. This is because as the part becomes smaller and smaller, its ability to recognize the same object location is reduced and the global capability of the model to discriminate examples is irreparably corrupted. This is because each part of displacement is independent from the other. Therefore, if instead of using father-child relationships we could use siblings relationships, as in chapter 5, each part displacement is conditioned by the close parts position and therefore a good modulation of the deformation should be obtained.

Better appearance model. An object model is generally composed of a grid (or parts) of weights that are associated to HOG features. Using a linear model, the final score is the dot product between the HOG features and the model. Thus, each weight and, therefore, the complete model can represent only a unimodal distribution. However, real object classes are very complex and can have multimodal appearances. Think for instance about the cloths of a pedestrian: skirt and trousers. They need multiple representations because local deformations are not enough for representing so different appearances. In this case, the object representation should be enlarged to

multiple appearances, that can be considered an additional level of latent variables.

3D models. Real objects live in a 3D environment, thus a 3D model should be better situated for their representation. Unfortunately, normal images only give the 2D projection of the real object. Still, if we can in some way learn the real 3D shape of an object, then its 3D projective distortions can be better approximated than by using local parts as we do now, and consequently better results should be expected. Initial works on 3D detection can be found in [88, 87, 113, 108], although, generally, they require a higher level of supervision, for instance the 3D pose annotation for each example.

We think that also the 3D pose can be introduced in our model as latent variable. In this case the main problem is to find a good way to initialize the 3D pose. In fact, as the latent SVM learning algorithm is not convex, a proper initialization of the latent variables is fundamental for good results. In contrast to object parts, where their location is in general close to their rest location, for 3D pose the latent variable has approximately the same probability to take any value. This means that, probably, a trivial initialization would be enough for learning a good 3D model.

Weak Supervision. The level of supervision generally used for object detection is fixed at bounding box level. Some methods use more supervision, for instance, using the object parts location [10] or the 3D object pose [108]. However, the real challenge is in obtaining similar or better results reducing the level of supervision. For instance, using a robust selection of the examples to use for training, we believe that it is possible to learn good detectors knowing only their presence in the image, but not their real location (the bounding box). Initial works on this field have already obtained discrete results [19, 26].

Appendix A

Datasets for Object Detection

INRIA

In object detection, human detection has aquired special importance due to the number of possible applications that can take advanage of it. In this sense, many mehtods for object detection are specific and specialized to human detection. The INRIA dataset has been presented in [24] and it is one of the standard datasets for human detection. It has been proposed to sobstitute the MIT dataset [83] that was the previous standard benchmark. In the MIT dataset humans have all a very similar appearance: standing and looking in front at the camera or backwards, but never lateral or spurious pose that will introduce much more appearance variation. In this way the MIT dataset is easier to learn, but at the same time a detector trained with this dataset will fail to detect humans not in same pose of the training images. Dalal and Triggs in their work [24] showed that HOG features perform almost perfectly in the MIT dataset and for this reason the intorduction of a more realistic and challenging dataset was necessary. In the INRIA dataset humans are taken from many different poses, perspectives, clothing, so that a much richer variety of examples is given. It contains 1218 pedestrains taken form 614 images as positive training examples and other 1218 images not containing any human as negatives examples. The test set is composed of 288 images containing a total of 740 humans and another 453 images not containing any human.

CVC02

In contrast to the INRIA dataset, CVC02 is a dataset specific for pedestrian detection [47]. This means that all images are taken from the same camera that is mounted on a car moving around a hurban environment. The dataset consists of pedestrian observed from 0 to 50 m, which corresponds approximatively to a bounding box of

70×140 to 12×24 pixels respectively. The training set consists of 1016 cropped humans with corresponding vertical mirror, for a total of 2032 images. The test set consists of 250 urban images containing pedestrians.

PASCAL VOC

The PASCAL VOC [35] is one of the most complete datasets for object detection. It contains 20 different classes of objects: vehicles (aeroplane, car, bus, bike, motorbike, train), person, animals (bird, cat, cow, dog, horse, sheep) and indoor objects (bottle, chair, dining table, potted plant, sofa, tv-monitor). The dataset is very challenging because it contains objects in realistic conditions which implies significant changes in illumination, point of view and scale as well as occlusions, objects interaction and background clutter.

Appendix B

Publications and Other scientific activities

Refereed Journals

- Marco Pedersoli, Jordi Gonzàlez, Andrew D. Bagdanov, Xavier Roca, **"Efficient Discriminative Multiresolution Cascade for Real-Time Human Detection Applications"**, Pattern Recognition Letters, Volume 32, Issue 13, pages 1581-1587 , October, 2011.
- Daniel Rowe, Jordi Gonzàlez, Marco Pedersoli, J. J. Villanueva, **"On Tracking Inside Groups"**, Machine Vision and Applications, Volume 21, Issue 2, Pages 113-127, February, 2010.
- Marco Pedersoli, Andrea Vedaldi, Jordi Gonzàlez and Xavier Roca, **"A Coarse-to-fine approach for fast deformable object detection"**, in IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) (under submission).
- Marco Pedersoli, Jordi Gonzàlez, Xu Hu, Xavier Roca, **"Towards a Real-Time Pedestrian Detector using only vision"**, in IEEE Transactions on Intelligent Transportation Systems (ITTS) (under submission).

Refereed Interneational Conferences

- Marco Pedersoli, Andrea Vedaldi, Jordi Gonzàlez, **"A Coarse-to-fine approach for fast deformable object detection"**, in 24th IEEE Computer Vision and Pattern Recognition (CVPR2011), Colorado Springs, CO, June, 2011, (**oral presentaion**).
- Marco Pedersoli, Jordi Gonzàlez, Andrew D. Bagdanov, Juan J. Villanueva, **"Recursive Coarse-to-Fine Localization for fast Object Detection"**,

in 11th European Conference in Computer Vision (ECCV2010), Crete, Greece, September, 2010.

- Marco Pedersoli, Jordi Gonzàlez, J.J. Villanueva, **"High-Speed Human Detection Using a Multiresolution Cascade of Histograms of Oriented Gradients"**, in 4th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA2009), Póvoa do Varzim, Portugal, June, 2009, (oral presentation).

- Marco Pedersoli, Jordi Gonzàlez, Bhaskar Chakraborty, Juan Jose Villanueva, **"Enhancing Real-time Human Detection based on Histograms of Oriented Gradients"**, In 5th International Conference on Computer Recognition Systems (CORES'2007), Wroclaw, Poland, October, 2007

- Nataliya Shapovalova, Wenjuan Gong, Marco Pedersoli, F. Xavier Roca and Jordi Gonzàlez, **"On Importance of Interactions and Context in Human Action Recognition"**, in 5th Iberian Conference on Pattern Recognition and Image Analysis (ibPRIA2011), Las Palmas de Gran Canaria, Canary Islands, Spain, June, 2011.

Projects

- **HERMES IST 027110**: Human-Expressive Representations of Motion and their Evaluation in Sequences.

- **VIDI-Video IST 045547**: Interactive Semantic Video Search with a Large Thesaurus of Machine-Learned Audio-Visual Concepts.

- **CICYT ERINYES TIN2009-14501-C02**: Epistemological Reasoning for the Interpretation of coNtext and securitY Events for Surveillance.

- **ViCoMo ITEA2 TSI-020400-2009-133**: Visual Context Modelling to improve security and logistics monitoring.

- **CONSOLIDER-INGENIO 2010 MIPRCV**: Multimodal Interaction in Pattern Recognition and Computer Vision.

- **CICYT SISYPHUS TIN2006-14606**: Security Indoor SYstem for Places with HUman in Scenes.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *Proc. CVPR*, 2010. [Page **27**]

[2] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *CVPR*, pages 1–8, 2008. [Page **2**]

[3] F. Bach. Sparse methods for machine learning. Tutorial, 2009. [Page **22**]

[4] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 2008. [Page **21**]

[5] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In *Proc. ECCV*, 2008. [Pages **26**, **32** and **75**]

[6] M. B. Blaschko, A. Vedaldi, and A. Zisserman. Simultaneous object detection and ranking with weak supervision. In *Proc. NIPS*, 2010. [Page **74**]

[7] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008. [Page **25**]

[8] A. Bosch, A. Zisserman, and X. Muñoz. Representing shape with a spatial pyramid kernel. In *Proc. CIVR*, 2007. [Page **23**]

[9] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *CVPR*, 2006. [Page **29**]

[10] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *Proc. ICCV*, 2009. [Page **106**]

[11] S. Boyd and L. Vanderberghe. *Convex Optimization*. Cambridge University Press, 2004. [Page **26**]

[12] Leo Breiman. Random forests. *machine learning*, pages 5–32, 2001. [Page **26**]

[13] Michael D. Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier, and Luc Van Gool. Robust tracking-by-detection using a detector confidence particle filter. In *IEEE International Conference on Computer Vision*, October 2009. [Page **2**]

[14] B. Karaçali and H. Krim. Fast minimization of structural risk by nearest neighbor rule. *IEEE Trans. on Neural Networks*, 14(1), 2002. [Page **25**]

[15] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *ECCV*, pages 778–792, 2010. [Page **21**]

[16] John Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986. [Page **20**]

[17] C. C. Chang and C. J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm. [Page **52**]

[18] Yuanhao Chen, Long Zhu, and Alan Yuille. Active mask hierarchies for object detection. In *Proceedings of the 11th European conference on Computer vision: Part V*, ECCV'10, pages 43–56, Berlin, Heidelberg, 2010. Springer-Verlag. [Pages **15**, **30**, **32** and **105**]

[19] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *Proc. CVPR*, 2007. [Page **106**]

[20] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *PAMI*, 25(5), 2003. [Page **2**]

[21] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967. [Pages **22** and **25**]

[22] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 1 edition, 2000. [Pages **23**, **25** and **26**]

[23] G. Csurka, C. R. Dance, L. Dan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Proc. ECCV Workshop on Stat. Learn. in Comp. Vision*, 2004. [Pages **30** and **63**]

[24] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, 2005. [Pages **13**, **21**, **23**, **27**, **30**, **32**, **34**, **36**, **39**, **41**, **44**, **51**, **54**, **59**, **60**, **61**, **64**, **65**, **67**, **69**, **75**, **77**, **89**, **96**, **101** and **107**]

[25] C. Desai, D. Ramanan, and C. Fowlkes. Discriminative models for multi-class layout. In *Proc. ICCV*, 2009. [Page **29**]

[26] Thomas Deselaers, Bogdan Alexe, and Vittorio Ferrari. Localizing objects while learning their appearance. In *Proceedings of the 11th European conference on Computer vision: Part IV*, ECCV'10, pages 452–466, Berlin, Heidelberg, 2010. Springer-Verlag. [Page **106**]

[27] P. Dollár, S. Belongie, and P. Perona. The fastest pedestrian detector in the west. In *Proc. BMVC*, 2010. [Pages **32**, **52** and **77**]

[28] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *BMVC*, 2009. [Page **77**]

[29] P. Dollar, Zhuowen Tu, Hai Tao, and S. Belongie. Feature mining for image classification. In *CVPR*, June 2007. [Pages **60** and **77**]

[30] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *Proc. CVPR*, June 2009. [Pages **13**, **43** and **59**]

[31] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Mach. Learn.*, 29(2-3):103–130, nov 1997. [Page **24**]

[32] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Comm. ACM*, 15(1), 1972. [Page **27**]

[33] M. Elad, Y. Hel-Or, and R. Keshet. . Pattern detection using a maximal rejection classifier. *PRL*, 23(12):1459–1471, 2002. [Page **29**]

[34] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge 2009 (voc2009) results, 2009. [Pages **75**, **76**, **83** and **86**]

[35] M. Everingham, A. Zisserman, C. Williams, and L. Van Gool. The pascal visual obiect classes challenge 2007 (voc20067) results. Technical report, ., 2007. [Pages **3**, **13**, **15**, **30**, **51**, **54**, **75**, **76**, **83**, **89** and **108**]

[36] G. Fanelli, J. Gall, and L. Van Gool. Real time head pose estimation with random regression forests. In *Computer Vision and Pattern Recognition (CVPR)*, pages 617–624, June 2011. [Page **27**]

[37] P. F. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Proc. CVPR*, 2010. [Pages **29**, **32**, **57**, **68**, **71**, **72**, **73**, **75**, **76**, **79**, **80** and **84**]

[38] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1), 2005. [Pages **63**, **64**, **65** and **71**]

[39] P. F. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proc. CVPR*, 2008. [Pages **11**, **13**, **15**, **26**, **27**, **30**, **32**, **36**, **37**, **43**, **50**, **52**, **60**, **61**, **66**, **73**, **74**, **75** and **81**]

[40] Pedro F. Felzenszwalb, Ross B. Girshick, David A. McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1627–1645, 2010. [Pages **21**, **30**, **32**, **51**, **63**, **64**, **65**, **69**, **70**, **77**, **80**, **88**, **89**, **91**, **94**, **95** and **96**]

[41] Vittorio Ferrari, Tinne Tuytelaars, and Luc J. Van Gool. Object detection by contour segment networks. In *ECCV*, pages 14–28, 2006. [Pages **21** and **32**]

[42] M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computer*, 22:67–92, 1973. [Pages **30**, **63** and **65**]

[43] Francois Fleuret and Donald Geman. Coarse-to-fine face detection. *Int. J. Comput. Vision*, 41(1-2):85–107, jan 2001. [Page 64]

[44] B. Fulkerson, A. Vedaldi, and S. Soatto. Localizing objects with smart dictionaries. In *Proc. ECCV*, 2008. [Page 24]

[45] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. In *Proc. CVPR*, 2010. [Page 28]

[46] D. M. Gavrila and S. Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *Int. J. Comput. Vision*, 73(1):41–59, jun 2007. [Page 29]

[47] D. Gerónimo. *A global approach to vision-based pedestrian detetcion for advanced driver assistance systems*. PhD thesis, Computer Vision Center, Universitat Autònoma de Barcelona, 2009. [Pages 3, 29, 32, 97, 98 and 107]

[48] M. Grabner, H. Grabner, and H. Bischof. Fast approximated sift. In *Proc. ACCV*, 2006. [Page 21]

[49] K. Grauman and T. Darrel. The pyramid match kernel: Discriminative classification with sets of image features. In *Proc. ICCV*, 2005. [Page 23]

[50] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. of The Fourth Alvey Vision Conference*, pages 147–151, 1988. [Page 20]

[51] Hedi Harzallah, Frederic Jurie, and Cordelia Schmid. Combining efficient object localization and image classification. In *International Conference on Computer Vision*, page 8, Kyoto, Japan, Sep 2009. [Pages 29, 30, 32, 43 and 51]

[52] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, jul 2006. [Page 6]

[53] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Putting objects in perspective. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2137–2144, Washington, DC, USA, 2006. IEEE Computer Society. [Pages 29 and 88]

[54] T. Joachims, T. Finley, and C. N. J. Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1), 2009. [Pages 26 and 51]

[55] T. Kadir and M. Brady. Saliency, scale and image description. *Int. J. Computer Vision*, 45:83–105, 2001. [Page 20]

[56] T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. In *Proc. ECCV*, 2004. [Page 20]

[57] C. H. Lampert. An efficient divide-and-conquer cascade for nonlinear object detection. In *Proc. CVPR*, 2010. [Page 32]

[58] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localizationby efficient subwindow search. In *Proc. CVPR*, 2008. [Pages 28, 30 and 32]

[59] S. Lazebnik and M. Raginsky. Learning nearest-neighbor quantizers from labeled data by information loss minimization. In *Proc. Conf. on Artificial Intelligence and Statistics*, 2007. [Pages **30** and **63**]

[60] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bag of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*, 2006. [Page **23**]

[61] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. J. Huang. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press, 2006. [Page **25**]

[62] Alain Lehmann, Bastian Leibe, and Luc Gool. Fast prism: Branch and bound hough transform for object class detection. *Int. J. Comput. Vision*, 94(2):175–197, sep 2011. [Page **53**]

[63] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *IJCV*, 2008. [Pages **28**, **32** and **53**]

[64] T. Lindeberg. Feature detection with automatic scale selection. *IJCV*, 30(2):77–116, 1998. [Page **20**]

[65] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. ICCV*, 1999. [Pages **19**, **20** and **21**]

[66] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2(60):91–110, 2004. [Pages **10**, **21** and **51**]

[67] Noha M. Elfiky, Fahad Shahbaz Khan, Joost van de Weijer, and Jordi Gonzílez. Discriminative compact pyramids for object and scene recognition. *Pattern Recogn.*, 45(4):1627–1636, apr 2012. [Page **24**]

[68] James MacQueen. Some methods for classification and analysis of multivariate observations. In *In Berkeley Symposium on Mathematical Statistics and Probability, 1967*, 1967. [Page **22**]

[69] S. Maji, A. C. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *Proc. CVPR*, 2008. [Pages **23**, **60** and **77**]

[70] D. R. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 1, 2004. [Page **20**]

[71] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. In *Proc. CVPR*, 2006. [Pages **32** and **53**]

[72] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proc. ICCV*, 2001. [Page **20**]

[73] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proc. ECCV*, pages 128–142. Springer-Verlag, 2002. [Page **20**]

[74] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, 2005. [Page **21**]

[75] F. Moreno-Noguer. Deformation and illumination invariant feature point descriptor. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1593–1600, 2011. [Page **21**]

[76] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithmic configuration. In *Proc. VISAPP*, 2009. [Pages **22** and **25**]

[77] J. Mutch and D. G. Lowe. Multiclass object recognition with sparse, localized features. In *Proc. CVPR*, 2006. [Page **10**]

[78] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Proc. NIPS*, 2001. [Page **24**]

[79] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *Proc. ECCV*, 2006. [Page **21**]

[80] R. Okada. Discriminative generalized hough transform for object detection. In *Proc. ICCV*, 2009. [Page **28**]

[81] Aude Oliva and Antonio Torralba. Building the gist of a scene: the role of global image features in recognition. In *Progress in Brain Research*, page 2006, 2006. [Pages **21** and **23**]

[82] A. Opelt, A. Pinz, and A. Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *Proc. CVPR*, 2006. [Pages **21** and **32**]

[83] M. Oren, C. P. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *cvpr*, pages 193–99, 1997. [Page **107**]

[84] D. Park, R. Ramanan, and C. Fowlkes. Multiresolution models for object detection. In *Proc. ECCV*, 2010. [Pages **29** and **94**]

[85] M. Pedersoli, J. Gonzàlez, A. D. Bagdanov, and J. J. Villanueva. Recursive coarse-to-fine localization for fast object detection. In *ECCV*, 2010. [Pages **66** and **80**]

[86] M. Pedersoli, A. Vedaldi, and J. Gonzàlez. A coarse-to-fine approach for fast deformable object detection. In *CVPR*, 2011. [Page **66**]

[87] J. Ponce, S. Lazebnik, F. Rothganger, and C. Schmid. Toward true 3d object recognition. In *Reconnaisance de Formes et Intelligence Artificelle*, 2004. [Page **106**]

[88] A. R. Pope and D. G. Lowe. Learning 3d object recognition models from 2d images. In *Proc. ICCV*, 1993. [Page **106**]

[89] Victor Adrian Prisacariu and Ian Reid. fasthog - a real-time gpu implementation of hog. *Science*, 2310(2310):1–13, 2009. [Pages **96**, **100** and **101**]

[90] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the Twenty-fourth International Conference on Machine Learning*, 2007. [Pages **7** and **8**]

[91] Rajesh P. N. Rao, Gregory J. Zelinsky, Mary M. Hayhoe, and Dana H. Ballard. Eye movements in iconic visual search. *Vision Res*, 42(11):1447–63, may 2002. [Page **52**]

[92] A. P. Dempster; N. M. Laird; D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977. [Page **22**]

[93] Mohammad Amin Sadeghi and Ali Farhadi. Recognition using visual phrases. In *CVPR*, 2011. [Page **9**]

[94] Ruslan Salakhutdinov, Antonio Torralba, and Josh Tenenbaum. Learning to share visual appearance for multiclass object detection. In *CVPR*, 2011. [Page **9**]

[95] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *PAMI*, 32(9), 2010. [Page **21**]

[96] B. Sapp, A. Toshev, and B. Taskar. Cascaded models for articulated pose estimation. In *ECCV*, 2010. [Pages **64** and **66**]

[97] A. D. Sappa, F. Dornaika, D. Ponsa, D. Geronimo, and A. Lopez. An efficient approach to onboard stereo vision system pose estimation, 2008. [Page **96**]

[98] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5), 1998. [Page **25**]

[99] P. Schnitzspan, M. Fritz, S. Roth, and B. Schiele. Discriminative structure learning of hierarchical representations for object detection. In *Proc. CVPR*, 2009. [Page **32**]

[100] W. R. Schwartz, A. Kembhavi, D. Harwood, and L. S. Davis. Human detection using partial least squares analysis. In *Proc. ICCV*, 2009. [Pages **30**, **32**, **43** and **77**]

[101] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated subgradient solver for svm. In *Proc. ICML*, 2007. [Page **26**]

[102] J. Shawe-Taylor and N. Cristianini. *Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000. [Page **23**]

[103] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *Proc. CVPR*, 2007. [Page **21**]

[104] Michael Shindler, Alex Wong, and Adam W. Meyerson. Fast and accurate k-means for large datasets. In J. Shawe-Taylor, R. S. Zemel, P. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2375–2383. 2011. [Pages **22** and **25**]

[105] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, 2003. [Pages **30** and **63**]

[106] J. Sochman and J. Matas. Waldboost-learning for time constrained sequential detection. In *CVPR*, 2005. [Page **29**]

[107] C. Stauffer and W. E. L. Grimson. Adaptive background misture models for real-time tracking. In *Proc. CVPR*, 1998. [Page **2**]

[108] H. Su, M. Sun, L. Fei-Fei, and S. Savarese. Learning a dense multi-vew representation for detection, viewpoint classification and synthesis of object categories. In *Proc. ICCV*, 2009. [Pages **32** and **106**]

[109] P. Sudowe and B. Leibe. Efficient use of geometric constraints for sliding-window object detection in video. In *ICVS*, 2011. [Pages **96**, **100** and **101**]

[110] Min Sun, Sid Yingze Bao, and Silvio Savarese. Geometrical context feedback loop. In *BMVC*, 2010. [Page **29**]

[111] C. Schmid T. Tuytelaars. Vector quantizing feature space with a regular lattice. In *Procs. ICCV*, 2007. [Page **23**]

[112] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Proc. NIPS*, 2003. [Page **26**]

[113] A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiele, and L. Van Gool. Towards multi-view object class detection. In *Proc. CVPR*, 2006. [Page **106**]

[114] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *PAMI*, 2010. [Page **21**]

[115] Antonio Torralba. How many pixels make an image? *Visual Neuroscience*, 26(1):123–131, 2009. [Page **52**]

[116] Antonio Torralba and Alexei A. Efros. Unbiased look at dataset bias. In *CVPR*, 2011. [Page **8**]

[117] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6, 2005. [Page **26**]

[118] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Conference on Computer Vision and Pattern Recognition*, pages 586–591. IEEE Comput. Sco. Press, 1991. [Page **30**]

[119] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Foundational Trends in Computer Graphics and Vision*, 3(3), 2007. [Page **20**]

[120] T. Tuytelaars and L. Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *Proc. BMVC*, pages 412–425, 2000. [Page **20**]

[121] O. Tuzel and F. Porikli alnd P. Meer. Pedestrian detection via classification on riemannian manifolds. *PAMI*, 30(10), 2008. [Pages **27** and **32**]

[122] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders. Segmentation as selective search for object recognition. In *ICCV*, 2011. [Pages **27** and **32**]

[123] Vladimir N. Vapnik. *Statistical learning theory*. Wiley, 1 edition, sep 1998. [Page **25**]

[124] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Proc. ICCV*, 2009. [Pages **27**, **28**, **29**, **30**, **32**, **34**, **43**, **80** and **88**]

[125] A. Vedaldi and A. Zisserman. Structured output regression for detection with partial occulsion. In *Proc. NIPS*, 2009. [Pages **26**, **32**, **70**, **73** and **94**]

[126] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *Proc. CVPR*, 2010. [Page **23**]

[127] S. Vijayanarasimhan and K. Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. In *CVPR*. [Pages **11**, **28** and **32**]

[128] S. Vijayanarasimhan and K. Grauman. Efficient region search for object detection. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1401–1408, 2011. [Page **28**]

[129] Michael Villamizar, Francesc Moreno-noguer, Juan Andrade-cetto, Alberto Sanfeliu, Institut De Rob, and Llorens Artigas. Efficient rotation invariant object detection using boosted random ferns. In *CVPR*, pages 1038–1045. IEEE, 2010. [Page **32**]

[130] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, june 2001. [Pages **15**, **51**, **52** and **64**]

[131] P. Viola and M. Jones. Robust real-time object detection. In *Proc. of IEEE Workshop on Statistical and Computational Theories of Vision*, Vancouver, Canada, 2001. [Page **29**]

[132] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 2004. [Pages **4**, **30**, **32**, **60** and **77**]

[133] Paul Viola, John C. Platt, and Cha Zhang. Multiple instance boosting for object detection. In *In NIPS 18*, pages 1419–1426. MIT Press, 2006. [Page **26**]

[134] S. Walk, N. Majer, K. Schindler, and B. Schiele. New features and insights for pedestrian detection. In *Proc. CVPR*, 2010. [Pages **32** and **77**]

[135] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. *Proc. CVPR*, 2010. [Page **22**]

[136] Xiaoyu Wang, Tony X. Han, and Shuicheng Yan. An hog-lbp human detector with partial occlusion handling. In *ICCV*, pages 32–39. IEEE, 2009. [Pages **30**, **32**, **43** and **51**]

[137] Christian Wojek and Bernt Schiele. A performance evaluation of single and multi-feature people detection. In *Proceedings of the 30th DAGM symposium on Pattern Recognition*, pages 82–91, Berlin, Heidelberg, 2008. Springer-Verlag. [Pages **43**, **60** and **77**]

[138] Rui Xu and Ii. Survey of clustering algorithms. 16(3):645–678, may 2005. [Page **22**]

[139] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *in IEEE Conference on Computer Vision and Pattern Recognition(CVPR*, 2009. [Pages **8**, **22** and **23**]

[140] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. *CVPR*, 2011. [Page **12**]

[141] C. N. J. Yu and T. Joachims. Learning structural svms with latent variables. In *Proc. ICML*, 2009. [Pages **26**, **73**, **74** and **94**]

[142] T. H. Yu, T. K. Kim, and R. Cipolla. Real-time action recognition by spatiotemporal semantic and structural forest. In *BMVC10*, 2010. [Page **27**]

[143] Alan Yuille, Anand Rangarajan, and A. L. Yuille. The concave-convex procedure (cccp. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002. [Page **26**]

[144] Luca Bombini Zani, Pietro Cerri, Paolo Grisleri, Simone Scaffardi, and Paolo. An evaluation of monocular image stabilization algorithms for automotive applications. In *Procs. IEEE Intl. Conf. on Intelligent Transportation Systems 2006*, pages 1562–1567, Toronto, Canada, SEP 2006. [Page **29**]

[145] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*, 2007. [Pages **21**, **30** and **63**]

[146] W. Zhang, G. Zelinsky, and D. Samaras. Real-time accurate object detection using multiple resolutions. In *Proc. ICCV*, 2007. [Pages **15**, **29**, **30**, **32**, **36**, **40**, **52** and **57**]

[147] Yu Zhong, Anil K. Jain, and M. P. Dubuisson-Jolly. Object tracking using deformable templates. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(5):544–549, 2000. [Page **2**]

[148] L. Zhu, Y. Chen, A. Yuille, and W. Freeman. Latent hierarchical structural learning for object detection. In *Proc. CVPR*, 2010. [Pages **64**, **66** and **80**]

[149] Qiang Zhu, Shai Avidan, Mei-chen Yeh, and Kwang-ting Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *In CVPR06*, pages 1491–1498, 2006. [Pages **29**, **30**, **32**, **37**, **52** and **57**]

[150] M. Zinkevich, M. Weimer, A. Smola, and L. Li. Parallelized stochastic gradient descent. In *Proc. NIPS*. 2010. [Page **95**]