

**Active Exploration of Training Data  
for Improved Object Detection**

by

Kenji Okuma

Master of Science, The University of British Columbia, 2003

Bachelor of Arts, Hiram College, 2000

A THESIS SUBMITTED IN PARTIAL FULFILLMENT  
OF THE REQUIREMENTS FOR THE DEGREE OF

**Doctor of Philosophy**

in

THE FACULTY OF GRADUATE STUDIES  
(Computer Science)

The University Of British Columbia  
(Vancouver)

February 2012

© Kenji Okuma, 2012

# Abstract

This thesis concerns the problem of object detection, which is defined as finding all instances of an object class of interest and fitting each of them with a tight bounding window. This seemingly easy task for humans is still extremely difficult for machines. However, recent advances in object detection have enabled machines to categorize many classes of objects. Statistical models are often used for representing an object class of interest. These models learn from extensive training sets and generalize with low error rates to unseen data in a highly generic manner. But, these statistical methods have a major drawback in that they require a large amount of training data. We approach this problem by making the process of acquiring labels less tedious and less costly by reducing human labelling effort. Throughout this thesis, we explore means of efficient label acquisition for realizing cheaper training, faster development time, and higher-performance of object detectors.

We use active learning with our novel interface to combine machine intelligence with human interventions, and effectively improve a state-of-the-art classifier by using additional unlabelled images from the Web. As the approach relies on a small amount of label input from a human oracle, there is still room to further reduce the amount of human effort. An ideal solution is, if possible, to have no humans involved in labelling novel data. Given a sparsely labelled video that contains very few labels, our novel self-learning approach achieves automatic acquisition of additional labels from the unlabelled portion of the video. Our approach combines colour segmentation, object detection and tracking in order to discover potential labels from novel data. We empirically show that our self-learning approach improves the performance of models that detect players in broadcast footage of sports games.



# Preface

All the work in this thesis has been conducted under the supervision of David G. Lowe and James J. Little. In addition, I have been fortunate enough to work with several outstanding co-authors on several publications which have become a large part of the thesis.

- Both Chapters 3 and 4 describe our active learning approach that uses a novel interface to combine machine intelligence with human interventions, effectively improving a state-of-the-art classifier by using additional unlabelled data from the Web. This work was presented orally at the International Conference on Computer Vision Theory and Applications (VISAPP) [Okuma et al., 2011]. Here, we worked with a co-author, Eric Brochu, who gave us invaluable guidance on developing the approach and how to design active learning experiments.
- Chapter 5 presents our previous work on the boosted particle filter (BPF) [Okuma et al., 2004] and compares its performance with our more recent collaborative work on another multi-target tracker [Lu et al., 2011] based on a Kalman filter. The work on BPF was presented orally at the 8th European Conference on Computer Vision (ECCV) [Okuma et al., 2004] and received the best paper prize in Cognitive Vision. Wei-Lwun Lu and I worked together on developing the tracking algorithm of [Lu et al., 2011] and evaluated its performance on broadcast footage of sports games.
- Finally, Chapter 6 describes a novel self-learning framework that automates the process of collecting additional training labels and improving models for detecting sports players in sparsely labelled broadcast footage of sports. This is an unpublished work. We are planning to submit the work to an international computer vision conference, aiming for publication sometime early in the year of 2012.

# Table of Contents

|  |             |
|--|-------------|
| <b>Abstract . . . . .</b>                            | <b>ii</b>   |
| <b>Preface . . . . .</b>                             | <b>iii</b>  |
| <b>Table of Contents . . . . .</b>                   | <b>iv</b>   |
| <b>List of Tables . . . . .</b>                      | <b>vii</b>  |
| <b>List of Figures . . . . .</b>                     | <b>viii</b> |
| <b>Acknowledgments . . . . .</b>                     | <b>x</b>    |
| <b>1 Introduction . . . . .</b>                      | <b>1</b>    |
| 1.1 Motivation . . . . .                             | 3           |
| 1.2 Application areas for object detection . . . . . | 7           |
| 1.3 Definitions of relevant terminology . . . . .    | 8           |
| 1.4 Contributions and overview . . . . .             | 9           |
| <b>2 Background . . . . .</b>                        | <b>12</b>   |
| 2.1 Object representation . . . . .                  | 12          |
| 2.1.1 Image descriptors . . . . .                    | 13          |
| 2.2 Object detection . . . . .                       | 21          |
| 2.2.1 Localizing objects . . . . .                   | 21          |
| 2.2.2 Using context . . . . .                        | 23          |
| 2.3 Learning . . . . .                               | 24          |
| 2.3.1 Supervised learning . . . . .                  | 25          |
| 2.3.2 Active learning . . . . .                      | 28          |

|          |   |           |
|----------|---|-----------|
| 2.3.3    | Semi-supervised learning . . . . .  | 29        |
| <b>3</b> | <b>Active Learning for Image Labelling . . . . .</b>                                    | <b>35</b> |
| 3.1      | Active learning for image labelling . . . . .   | 36        |
| 3.2      | Algorithm . . . . .   | 37        |
| 3.3      | Active learning on a latent SVM . . . . .   | 38        |
| 3.4      | The PASCAL visual object classes challenge 2007 . . . . .                               | 40        |
| 3.5      | INRIA person dataset . . . . .  | 44        |
| <b>4</b> | <b>Experiments with Active Learning using an Internet Image Search Engine . . . . .</b> | <b>47</b> |
| 4.1      | An adaptive interface for active localization . . . . .                                 | 47        |
| 4.2      | Experiments . . . . .   | 48        |
| 4.3      | Conclusion . . . . .  | 54        |
| <b>5</b> | <b>Multi-target Tracking in Sports Video . . . . .</b>                                  | <b>59</b> |
| 5.1      | Boosted particle filter . . . . .   | 60        |
| 5.1.1    | Statistical model . . . . .   | 60        |
| 5.1.2    | Observation likelihood . . . . .  | 60        |
| 5.1.3    | Particle filtering . . . . .  | 62        |
| 5.1.4    | Boosted particle filter . . . . .   | 63        |
| 5.1.5    | Multi-target tracking . . . . .   | 65        |
| 5.2      | Comparisons between BPF-Adaboost and KF-DPM . . . . .                                   | 66        |
| 5.2.1    | Implementation differences . . . . .  | 66        |
| 5.2.2    | Performance comparison . . . . .  | 68        |
| 5.3      | Conclusion . . . . .  | 69        |
| <b>6</b> | <b>Self-Learning for Player Localization in Sports Video . . . . .</b>                  | <b>70</b> |
| 6.1      | Weakly-supervised self-learning for player localization . . . . .                       | 72        |
| 6.1.1    | System overview . . . . .   | 74        |
| 6.2      | Semi-supervised learning in videos . . . . .  | 75        |
| 6.3      | Player detection . . . . .  | 77        |
| 6.4      | Colour classification . . . . .   | 78        |
| 6.4.1    | Team classification . . . . .   | 78        |
| 6.4.2    | Figure-ground segmentation . . . . .  | 79        |

|          |   |            |
|----------|---|------------|
| 6.5      | Player tracking . . . . .                             | 83         |
| 6.6      | Data selection . . . . .                              | 86         |
| 6.7      | Experiments . . . . .                                 | 88         |
| 6.7.1    | Data . . . . .  | 88         |
| 6.7.2    | Player detection . . . . .                            | 89         |
| 6.7.3    | Player tracking . . . . .                             | 93         |
| 6.7.4    | Data selection . . . . .                              | 95         |
| 6.7.5    | Computational time . . . . .                          | 95         |
| 6.8      | Application to other sports video . . . . .           | 98         |
| 6.8.1    | Basketball . . . . .                                  | 98         |
| 6.9      | Conclusion . . . . .                                  | 99         |
| <b>7</b> | <b>Conclusions . . . . .</b>                          | <b>105</b> |
| 7.1      | Toward the future of crowdsourcing . . . . .          | 106        |
| 7.2      | Tradeoff between complexity and scalability . . . . . | 108        |
| 7.3      | Future directions . . . . .                           | 108        |
|          | <b>Bibliography . . . . .</b>                         | <b>112</b> |

# List of Tables

|           |   |     |
|-----------|---|-----|
| Table 4.1 | Average precision on VOC2007 and training data statistics . . . . . | 55  |
| Table 4.2 | Query statistics for ALORquery and ALORfull . . . . .               | 56  |
| Table 5.1 | Detection comparison of Adaboost and DPM . . . . .                  | 67  |
| Table 5.2 | Definitions for tracking evaluation metrics . . . . .               | 68  |
| Table 5.3 | Tracking performance of BPF-Adaboost and KF-DPM in our hockey video | 69  |
| Table 6.1 | Average number of labels used for each labelled dataset . . . . .   | 92  |
| Table 6.2 | Tracking evaluation metrics . . . . .                               | 93  |
| Table 6.3 | Hockey tracking results . . . . .                                   | 94  |
| Table 6.4 | Average number of labels used for each labelled dataset . . . . .   | 100 |
| Table 6.5 | Basketball tracking results . . . . .                               | 101 |

# List of Figures

|            |  |    |
|------------|--|----|
| Figure 1.1 | Screen shots of object detection results . . . . .   | 2  |
| Figure 1.2 | Challenges in object detection . . . . .   | 5  |
| Figure 1.3 | Screenshots of object tracking result in sports videos . . . . .                                   | 6  |
| Figure 2.1 | Local and global similarity . . . . .  | 14 |
| Figure 2.2 | Recognition results of SIFT . . . . .  | 17 |
| Figure 2.3 | The training procedure of an appearance codebook . . . . .   | 18 |
| Figure 2.4 | The parts of the pictorial structure model . . . . .   | 18 |
| Figure 2.5 | Deformable part model . . . . .  | 20 |
| Figure 2.6 | Localization process of the deformable part model . . . . .  | 22 |
| Figure 2.7 | Visual phrase . . . . .  | 24 |
| Figure 2.8 | Related machine learning formalisms for detecting koalas and pandas .                              | 26 |
| Figure 2.9 | Semi-supervised learning for detecting faces . . . . .   | 31 |
| Figure 3.1 | PASCAL VOC training images with ground truth annotations . . . . .                                 | 41 |
| Figure 3.2 | These are the first 12 figures of a two-page figure that is continued on<br>the next page. . . . . | 42 |
| Figure 3.2 | VOC2007: Random vs. active sampling of positive data . . . . .                                     | 43 |
| Figure 3.3 | Some example data from the INRIA pedestrian dataset . . . . .                                      | 44 |
| Figure 3.4 | DET curve on INRIA dataset . . . . .   | 46 |
| Figure 4.1 | Screen shots of our interactive labelling system with our actively trained<br>LSVM . . . . .       | 49 |
| Figure 4.2 | Some images from Flickr . . . . .  | 51 |
| Figure 4.3 | Training images from Flickr, sorted for active learning . . . . .                                  | 52 |
| Figure 4.4 | Performance comparison of ALORfull and ALORquery . . . . .   | 53 |

|             |   |     |
|-------------|---|-----|
| Figure 4.5  | Category-wise breakdown of gain in average precisions: VOC baseline<br>vs VOC baseline + ALOR . . . . . | 57  |
| Figure 4.6  | Heterogeneous shapes and sizes . . . . .  | 58  |
| Figure 5.1  | HSV Colour histograms . . . . .   | 61  |
| Figure 5.2  | Training data for the Adaboost detector . . . . .   | 64  |
| Figure 5.3  | Hockey player detection results . . . . .   | 64  |
| Figure 5.4  | Mixture of Gaussians for the proposal distribution . . . . .  | 65  |
| Figure 6.1  | System overview . . . . .   | 73  |
| Figure 6.2  | Challenges in player detection . . . . .  | 74  |
| Figure 6.3  | Colour-based deformable part models . . . . .   | 80  |
| Figure 6.4  | Player detection and team colour classification results . . . . .                                       | 81  |
| Figure 6.5  | Figure-ground segmentation results . . . . .  | 84  |
| Figure 6.6  | Results of “objectness” on hockey data . . . . .  | 85  |
| Figure 6.7  | Ground-truth annotation data for hockey players . . . . .   | 89  |
| Figure 6.8  | Detection result of our weakly self-learning system in hockey videos . .                                | 92  |
| Figure 6.9  | Screenshots of our tracking result in hockey videos . . . . .   | 96  |
| Figure 6.10 | Most confident and least confident candidate bounding windows in hockey<br>videos . . . . .             | 97  |
| Figure 6.11 | Total train and test time in hockey data . . . . .  | 98  |
| Figure 6.12 | Detection result of our weakly self-learning system in basketball videos                                | 100 |
| Figure 6.13 | Screenshots of our tracking result in basketball videos . . . . .                                       | 102 |
| Figure 6.14 | Most confident and least confident candidate bounding windows in bas-<br>ketball videos . . . . .       | 103 |
| Figure 6.15 | Total train and test time in basketball data . . . . .  | 104 |
| Figure 7.1  | Variability of data in broadcast footage of sports games . . . . .                                      | 110 |

# Acknowledgments

I am fortunate to have had remarkable mentors, colleagues, friends, and family who have always given me invaluable support and guidance both prior to and during my time as a graduate student at UBC.

First of all, I would like to thank my supervisors, David Lowe and James Little. They have always been supportive and allowed me to explore my own direction of research. I have been and always will be thankful for their excellent scholarly guidance and support. I would have never been able to complete my PhD without their supervision. I would also like to thank the other members of my supervisory committee: Nando de Freitas and Sidney Fel.

I have been fortunate enough to work with outstanding collaborators and colleagues in LCI. Countless discussions and fruitful conversations with them have also inspired me to continue my research to this end. But, I won't try to name them all for fear of leaving someone out.

I have been lucky enough to be funded throughout my PhD. My research has been funded by the Natural Sciences and Engineering Research Council of Canada and the GEOIDE Network of Centres of Excellence, Terrapoint Canada (A division of Ambercore).

Last but far from least, I would like to thank Mum and Dad for encouraging me in the pursuit of my dreams. I would also like to thank my fiancé, Lana Ohira, for her patient encouragement and support. I am deeply grateful that she was standing beside me when I struggled.



# Chapter 1

## Introduction

A significant body of scientific literature has been devoted to making machines see things as well as humans do. It is usually very easy for humans to localize an object of interest that they see in front of them. For example, whenever you see a flower, you can easily find its location, size and shape. However, it is still extremely difficult for machines to do the same with similar accuracy. This thesis studies the problem of detecting objects — finding any instances of an object class of interest and fitting each of them with a tight bounding window. The problem of object detection is typically formulated as a search problem:

$$R^* = \operatorname{argmax}_{R \subseteq I} f(R|I, \Theta)$$

where  $\Theta$  is a trained object model, and  $R$  ranges over all bounding windows (often rectangular or ellipsoidal shapes) in an image  $I$ . The goal is to find the best bounding window  $R^*$  based on a score function  $f$ , which is usually designed with classification algorithms that depend on training data.

Recent advances in object detection have enabled machines to categorize many classes of objects. Statistical models are often used for representing an object class of interest. These models learn from extensive training sets.

But, these statistical methods have a major drawback in that they require a large amount of training data. In order to achieve performance levels that are sufficient for practical applications, it is common that more than a million labelled instances are used for the

training of a single object class [Viola and Jones, 2004]<sup>1</sup>. One way to resolve this issue is to make the process of acquiring labels less tedious and less costly by reducing human labelling effort.



**Figure 1.1: Screen shots of object detection results.** This shows how our active learning approach improves object localization (Chapter 3). The top left shows the highest scoring detection using only the PASCAL VOC 2007 data, whereas the top right shows the result after retraining with additional data obtained using our active learning interface. The bottom row shows associated confidence maps in the same scale where each point gives the probability of a detection centred at that point. Note how the detection confidence increased and the peak position was shifted. Yellow boxes are the ground truth and red boxes are detections with the associated detection confidence.

Our motivating example of Figure 1.1 illustrates how training data influence results of detecting a bus in an image. The figure shows detection results with red bounding boxes in the top row and corresponding confidence maps in the bottom row. The images in the right column are results of training with a larger amount of data. The detection result

<sup>1</sup>Viola and Jones [2004] trained a face detector with 4,916 bounding windows for the face class and 350 million bounding windows for the non-face background class.

in the left column is clearly worse than the one in the right column where the detection confidence is increased and the object of interest is covered by a tighter bounding window. This performance improvement is due to the amount and quality of training data.

It is possible to improve the performance of an object detector by simply adding more labelled data. However, the cost of providing labelled data — asking a human to examine images and provide labels — becomes impractical as training sets grow. Throughout this thesis, we will explore means of efficient label acquisition for realizing cheaper experiments, faster development time, and higher-performance object detectors.

## 1.1 Motivation

The problem of generic object detection is largely unsolved for machines, but it is an important cognitive task humans do in everyday life. Humans are our best example of a working recognition system, which suggests a clue for solving the problem.

Regardless of fundamental differences between machine vision and biological visual systems, there is one thing that is common — both systems learn from data. From early in life, humans observe huge amounts of visual data from which they learn to recognize objects. Suppose humans capture snapshots of visual information at the rate of 30 frames per second, then humans observe  $30 \text{ (images)} \times 3600 \text{ (seconds)} \times 16 \text{ (hours)} \times 365 \text{ (days)} \approx 6 \times 10^8$  images per year on average. This visual data is largely unlabelled with only a small amount of labels provided by parental teaching or some form of supervision. This informal argument implies that object recognition research will require hundreds of millions of images to build a similar working recognition system. We believe that one way to build a successful computer vision system for recognizing generic objects is to find a way to prepare data on a similar level of scale.

All approaches in object detection must be confronted with two issues: the model and the data. The vast majority of research has been devoted to enriching the model by seeking more sophisticated and complex parametric representations of objects. A much smaller body of work has explored aspects of the training data to improve the performance of existing models.

We believe that the data is as important as the model and that existing models may be sufficient for solving the problem if we have enough training data. In linguistics, speech recognition and machine translation systems based on n-gram language models outper-

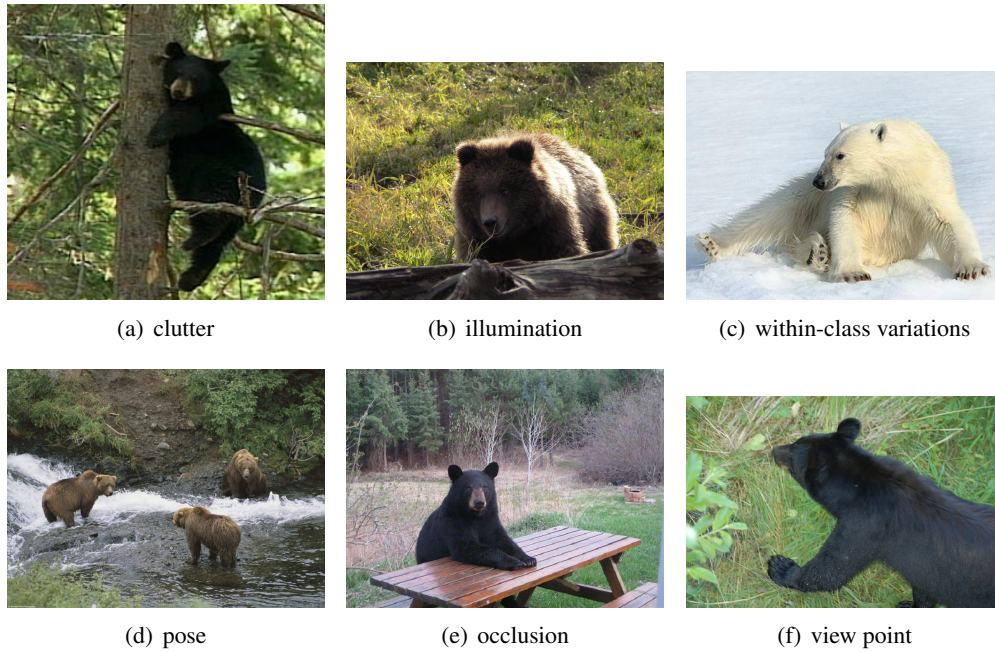
formed other systems that were based on more complex grammars and phrase structure. In computer vision, simple nearest-neighbour algorithms with 79 million tiny ( $32 \times 32$ ) image patches can give reasonable performance on object recognition tasks based on Euclidian distance of intensity as shown in the work of Torralba et al. [2007]. These are successful examples of scalable simple models with large data.

Today, we have access to abundant images and videos on the Web at the scale which resembles the amount of visual information that humans experience for learning how to recognize objects. Large datasets are required for solving the problem of object detection, but that does not mean that any large dataset is helpful. The quality of the data is also important: “good” datasets should at least contain images that include several major challenges of object detection as shown in Figure 1.2. Preparing data means not only obtaining large volumes of such “good” data but also acquiring bounding windows and their labels which, in the case of Figure 1.2, are “bear”. This is a new avenue for further exploration in computer vision [Berg et al., 2010]. Recently, crowdsourcing, which uses an extensive human workforce on a massively parallel platform, has shed light on this problem [Deng et al., 2009]. But there still is no fully automatic way of getting labels from huge amounts of unlabelled data.

The relationships between object detection and tracking are also important issues in this thesis. Our previous work on the boosted particle filter [Okuma et al., 2004] combines the strengths of two successful algorithms: Adaboost for object detection and mixture particle filters for multi-target tracking. Our work has been influential in tracking-by-detection approaches that are the current mainstream of tracking research in the computer vision community.

We believe that solving object detection contributes to solving object tracking. While object detection is a task of identifying a bounding window for all instances from an object class of interest in an image, object tracking in images or a video means associating target objects in consecutive images, and generating a trajectory of each target. The trajectory is defined as a unique sequence of bounding windows. Therefore, detecting bounding windows in consecutive images can form trajectories of target objects.

In the ideal case, the perfect detector specifies a bounding window that fits the target object in every single frame of the video, drawing a smooth trajectory of the target object. However, object detection is rarely perfect. Just as how a set of dots that form a curve may be sparse, the bounding windows provided by object detection form a trajectory. In return,



**Figure 1.2: Challenges in object detection.**

the trajectory can provide information about where it may possibly extend in the near future, suggesting where the bounding windows may be present. In this way, object detection and tracking can cooperate with each other.

Figure 1.3 shows tracking results in sports videos based on our tracking-by-detection approach in Chapter 6. These tracking results are based on different detectors. The left column (a) uses a detector that is trained with only 5 labelled images. The right column (b) uses an improved detector that is trained with the same labelled images as well as additional labels that our self-learning approach collected automatically from unlabelled data. The figure shows how an improved performance of object detection positively influences the performance of object tracking.

Last but not least, object detection has many practical applications that immediately address demanding problems in the real world. The next section covers a wide range of both consumer and industry applications.



(a) Prior to self-learning

(b) After self-learning

**Figure 1.3: Screenshots of object tracking result in sports videos (Chapter 6).** Column (a) uses detection inputs of a detector that is trained with 5 labelled images. Column (b) uses an improved detector that is trained with both labelled and unlabelled data. Note that more players are discovered and tracked successfully with the better detector. Frame numbers are shown in the upper left corner of each image.

## 1.2 Application areas for object detection

There are many applications of generic object detection. In particular, person and face detection have a broad range of applications and have attracted researchers from both academia and industry in recent years. The following list shows some consumer applications that people may be familiar with:

- **Auto-focus:** Modern digital cameras have a built-in face detector to automatically locate faces of people and to automatically adjust its zooming/focus parameters.
- **Digital content management:** Picasa, a free photo editing program from Google, has an intelligent digital content management tool to automatically cluster faces and tag them to facilitate search. Statistics show that the average digital camera owner take around 170 photos in a week, over 8,000 photos in a year, at which point it becomes tedious for humans to manually search, locate, and label these photos.
- **Automotive safety:** Pedestrian detection has been applied to smart cars in order to warn drivers when there are people on the road while driving, and when there are pedestrians on the sidewalks while parking.
- **Driverless driving:** Computer vision recognition system has been applied for detecting near-by obstacles around a vehicle when other sensors are not as effective. Google has tested their autonomous cars while driving over 1,000 miles without human intervention [Markoff, 2010]. In the next several years, there may be autonomous cars that run on public roads.
- **Copyright protection:** For commercial films and video content, object detection plays a crucial role for detecting illegal content by searching for the copyright logo on the Web.
- **Surveillance:** In video surveillance and security, real-time detection systems are employed to analyse and process video sequences for signs of intrusion.
- **Visual search:** Google Goggles is a mobile application on smartphones. It allows users to perform a visual search based on a picture that a user takes, comparing it to images found on the Web. The search works for an object of various kinds such as a book, a wine bottle or places such as a bridge or a statue.

There are also industrial applications:

- **Optical character recognition (OCR):** Automatic recognition systems for optical characters have been used for recognizing handwritten postal codes on letters and number plates.
- **Machine inspection:** Real-time vision systems are also used for high precision inspection in industrial factory automation settings. These systems are designed for detecting rigid objects such as parts of automobiles, and are used to detect failure or defects in the products.

There are many other applications for object recognition systems in computer vision. We recommend a recent book on vision algorithms and applications by [Szeliski, 2010] or a comprehensive list of both consumer and industry applications from David G. Lowe's website <http://www.cs.ubc.ca/~lowe/vision.html>.

### 1.3 Definitions of relevant terminology

Object recognition is a combination of three different tasks that include classifying, localizing, and categorizing instances from classes of objects. For example, we consider a scenario where you are looking for Albert Einstein in a series of pictures. For every picture you see, you may ask the following questions: Is there a person in this picture? If so, where does each person precisely appear in the picture? Finally, is the person you found Albert Einstein? To address these issues from a machine's perspective, there is a vast amount of literature on object recognition in computer vision community. However, the relevant terminology is rather loosely defined. To avoid confusion, important terms are defined here.

- **Image classification**

*Image classification* is to classify an image as a whole and verify the presence of any instances (often only one instance) of an object class in the image, formulated as a one-class problem.

- **Categorization**

*Categorization* is discrimination between  $n$  different classes where  $n > 1$ . *Object categorization* is to find any instances of an object class within an image and assign them with the correct object category (class) labels. Both *class* and *category* mean



the same and they are used interchangeably throughout the thesis. However, *image categorization* is different from *image classification* because the former is a  $n$ -class problem and the latter is a one-class problem.

- **Localization**

Object *localization* is to find any instances of an object class in an image and specify tight bounding windows around them. In contrast to *image classification* where only one bounding window of a whole image needs to be classified, *localization* requires determining  $n$  bounding windows and selecting from them only those ones that fit best around objects.

- **Detection**

Object *detection* has a task of solving both *localization* and *categorization*. In the open literature, object *detection* is particularly loosely defined and is often used interchangeably with object *localization* or object *categorization*. To avoid confusions, this thesis treats object *detection* as a two-class classification problem, where candidate image regions are evaluated on whether they contain (1) an instance of an object class of interest or (2) an instance of an all-encompassing background class. In this respect, object *detection* means the same as object *categorization*.

- **Recognition**

*Recognition* refers to the most generic problem that includes any combination of *localization*, *classification*, and *categorization*.

## 1.4 Contributions and overview

For recognizing object categories, we need to address three main issues: object representation, detection, and learning. This thesis studies these issues from a machine’s perspective and explores aspects of training data for improving object detection.

Generic object detection requires a model to have a score function that can identify bounding windows for each object in the image — there can be zero, one or many such boxes in a single image. Such a model is often developed by machine learning techniques. However, there are many challenges to identifying bounding windows in a large search space, such as background clutter, occlusions, illumination changes, and intra-class variations of the appearance of objects as shown in Figure 1.2.

Broadly speaking, there are two ways to overcome these challenges: One is to enrich the models by making more complex parametrization of objects, and the other is to prepare more training data for improving their performance. While a significant body of work has been devoted to enriching models of generic object detection, there is a much smaller body of work that addresses aspects of training data for improving the performance of existing detection models. The goal of this thesis is to explore and discover additional training data that can enhance performance of existing detection models without much human labelling effort.

In Chapter 2, we provide the necessary background for the problem of object detection and introduce existing work in the area. The chapter covers main issues with regard to designing a vision system for recognizing object categories. Section 2.1 presents representations of objects based on advanced local image descriptors. In Section 2.2, we explain object detection and briefly discuss image context as an additional cue to improve localization of objects. Section 2.3 focuses on the learning aspects of the problem and introduces related machine learning formalisms in the context of object detection.

The following chapters present means of reducing labelling effort on unlabelled data in static images and videos, which comprise the major contributions of this thesis:

- In Chapters 3 and 4, we show that our active learning approach, which uses a novel interface to combine machine intelligence with human interventions, effectively improves a state-of-the-art classifier by using additional unlabelled data from the Web and limited human assistance in labelling. Our approach improved the average precision of a state-of-the-art classifier of [Felzenszwalb et al., 2009] from 26.1% to 30.6%. The improvement is averaged over 20 object classes of the PASCAL Visual Object Classes Challenge 2007 and is based on an average of just 40 minutes of human labelling effort per class.
- Chapter 5 presents our previous work on the boosted particle filter (BPF) [Okuma et al., 2004] and compares its performance with our more recent collaborative work on another multi-target tracker [Lu et al., 2011] based on a Kalman filter. We explain implementation differences of these trackers and analyse why the Lu et al. [2011] tracker outperforms the BPF tracker.

- Chapter 6 introduces a novel self-learning framework that automates the process of collecting additional training labels and improving models for detecting sports players in broadcast footage of sports. Unlike most of the previous self-learning approaches for improving appearance-based object detectors, we allow an unknown, unconstrained number of target objects in a video. Our experimental results show that our approach is particularly effective when there is a very small amount of labelled data, improving the mean performance on localizing sports players by exploiting unlabelled data in sparsely labelled sports videos — the average precision is improved from 38.5% to 66.3% in hockey and from 29.0% to 61.6% in basketball with only 5 labelled images in both cases.

Chapter 7 concludes this thesis with a couple of discussion topics and remarks for future directions of our research.

## Chapter 2

# Background

As object recognition is one of the fundamental challenges in computer vision, a significant body of previous work has been devoted to the problem of generic object detection. In this chapter, we will look at the existing work in this area and introduce the main issues related to designing a computer vision system for detecting generic objects in images and videos. In Sections 2.1 and 2.2, we cover object representation and detection. Section 2.3 summarizes related machine learning formalisms that have been used in object detection scenarios.

### 2.1 Object representation

In designing a computer vision system for detecting generic objects, we first need to define an object class of interest using image features. Image features are used to represent image regions or parts of videos, encoded as feature vectors which are often defined in high dimensions of hundreds or even thousands of real values. Robust object detection systems require a representation that recognizes both inter-class and intra-class variability. For instance, the appearance of pedestrians are quite different from that of cars (i.e., inter-class differences), but pedestrians also have quite different appearances in different poses (i.e., intra-class differences). To encode such visual structure of an object class, feature vectors need to be invariant to changes in illumination and differences in viewpoint, yet be sensitive to the appearance of different object classes. Many forms of advanced local image descriptors have been introduced for this purpose.

### **2.1.1 Image descriptors**

There is a large body of previous work on many different image descriptors for encoding local image regions: of particular relevance to this thesis is the work on those that have been successful in object detection. They can be based on wavelets, image intensities or gradients. For those who are interested, we recommend [Mikolajczyk and Schmid, 2005] for a more comprehensive survey and performance comparisons of many local image descriptors.

#### **Wavelets**

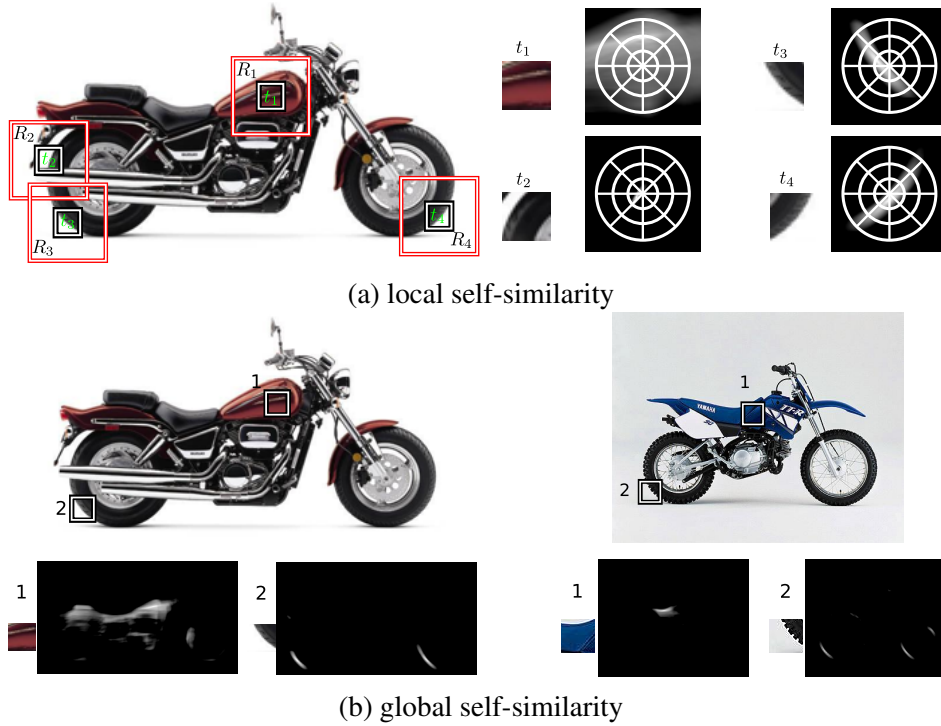
Image descriptors based on Haar wavelets [Mallat, 1989] have been successful in detecting pedestrians and faces [Papageorgiou and Poggio, 2000; Viola and Jones, 2004]. Papageorgiou and Poggio [2000] described object classes in terms of a large set of local oriented intensity differences between adjacent regions. This representation is efficiently encoded by a Haar wavelet transform, creating an over-complete dictionary of Haar wavelets at different orientations and scales. The over-completeness gives a rich multi-resolution representation of an image with wavelet features densely sampled at different scales capturing different levels of detail.

Following the work of Papageorgiou and Poggio [2000], Viola and Jones [2004] built a cascade chain of classification systems that uses generalized Haar wavelets. More complex Haar basis functions give an even richer representation of an object class, resulting in a better generalized performance for detecting faces of the people. The detector runs in real-time: the integral image representation for images enables fast computation of Haar wavelets, and the cascade structure of multiple classifiers reduces redundant computations by guiding attention (i.e., computational resources) on promising regions of the image where the target object instances are most likely present. Viola et al. [2005] extended the real-time face detector to pedestrian detection in videos. Unlike a collection of random static images, the temporal coherence between adjacent frames in a video is used as motion descriptors to improve performance.

#### **Image intensity**

The simplest descriptor is a vector of image intensity, which has been effectively used for computing correlations among image regions. Local self-similarities in [Shechtman and Irani, 2007] represent self-similarities within relatively small (e.g., 40 pixels radius) re-

gions. Self-similarities encode the correlation (e.g., sum of square differences) of a local image patch and its surrounding local region based on image intensities. Deselaer and Ferrari [2010] proposed computationally efficient algorithms of global self-similarity. While local self-similarities are limited in describing local geometric layout of image intensity patterns, global self-similarities extend local self-similarities to the entire image and capture a much wider range of spatial intensity patterns. Figure 2.1 shows both local and global self-similarity descriptors of motorcycles.



**Figure 2.1: Local and global self-similarity.** (a) shows the local self-similarity of four patches  $t_n$  within local regions  $R_n$  specified by the red square. They are quantized with a log-polar grid. (b) shows global self-similarities of two patches over their respective images. Both figures are reproduced from [Deselaer and Ferrari, 2010] ©2010 IEEE

## **Image gradients**

In recent years, image descriptors based on image gradients have attracted much attention in object detection. The most notable ones include shape contexts [Belongie et al., 2001], the Scale Invariant Transform (SIFT) [Lowe, 1999, 2004], and the Histogram of Oriented Gradients (HOG) [Dalal, 2006; Dalal and Triggs, 2005]. SIFT uses weighted orientation histograms based on the local scale and dominant orientation of interest points given by the keypoint detector. The scale information is used to control the smoothing parameter for computing image gradients. SIFT descriptors are therefore scale and rotation invariant feature vectors. While SIFT computes histograms over rectangular grids, shape contexts use log-polar grids and sample edges into a 2-D log-polar histogram. This model was extended by Mori and Malik [2003] to generalized shape contexts that sample edges into a 3-D spatial and orientation histogram.

More recently, Dalal and Triggs [2005] proposed grids of Histograms of Oriented Gradient (HOG). HOG descriptors are also based on image gradients as in SIFT or shape contexts, except that they are computed on a dense grid of uniformly spaced grids and use overlapping local contrast normalizations. HOG descriptors have been widely used in pedestrian detection. Several state-of-the-art approaches [Bourdev et al., 2010; Dalal and Triggs, 2005; Felzenszwalb et al., 2009] used HOG descriptors and showed excellent performance on pedestrian datasets from the PASCAL Visual Object Classes Challenge [Everingham et al., 2010]. The detectors by [Dalal and Triggs, 2005] and Felzenszwalb et al. [2009] won the PASCAL object detection challenge respectively in 2006 and 2009. Following these advances in object detection, Bourdev et al. [2010] proposed a new representation of an object class called poselets. Poselets are a set of HOG descriptors that are clustered based on both geometric configurations (e.g., joints of a person) of object parts and their appearance variations. Bourdev et al. [2010] outperformed the previous methods in pedestrian detection and achieved the best performance of 47.8% average precision on the PASCAL data in 2009, though the detector runs much slower than [Felzenszwalb et al., 2009].

## **Sparse representation**

A set of local descriptors represent an object class by capturing relevant local image regions. These local regions are often detected by point detectors based on saliency and scale invariance. They can be detected based on points [Harris and Stephens, 1988; Miko-

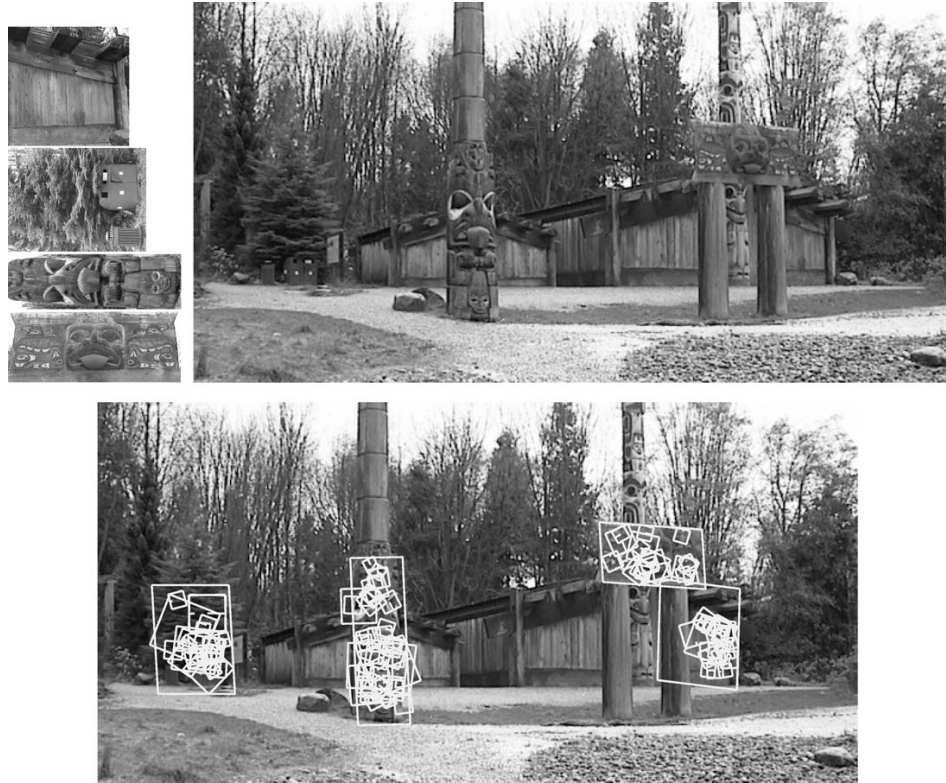
lajczyk and Schmid, 2002], blobs such as Laplacian of Gaussians [Lindeberg, 1998] and Difference of Gaussians (DoG) [Lowe, 2004], intensities [Kadir and Brady, 2001], edges [Jurie and Schmid, 2004], and combinations of colour and texture [Martin et al., 2004]. Sparse representations based on relevant local descriptors have attracted major attention because they are computationally efficient and robust to difficult image variations arising from occlusions and changes in viewpoints, illuminations and scale. Fergus et al. [2003] used the entropy based region detector [Kadir and Brady, 2001] for detecting features and local intensity patches for the appearance representation. Dorkó and Schmid [2003]; Opelt et al. [2006a] used several interest point detectors such as Harris [Harris and Stephens, 1988], Harris-Laplace [Mikolajczyk and Schmid, 2002] and Laplacian of Gaussian [Mikolajczyk and Schmid, 2002]. For detecting and recognizing a specific instance of an object class, SIFT with a scale-invariant DoG detector [Lowe, 2004] has shown the state-of-the-art recognition performance with a fast nearest neighbour matching algorithm called the Best-Bin-First algorithm. Figure 2.2 shows impressive recognition results of SIFT in a complex scene.

These point-based representations are often used in bag-of-features methods where objects are represented by an orderless collection of distinctive local features [Grauman and Darrell, 2007; Zhang et al., 2006b]. However, bag-of-features methods are severely limited in object detection because they disregard most information about the spatial layout of precise feature locations and therefore do not work well for localizing objects. Bag-of-features methods [Grauman and Darrell, 2007; Lazebnik et al., 2006; Zhang et al., 2006b] have therefore been more successful in problems of classifying images rather than detecting objects.

Local image descriptors are more effective in the form of a codebook for object detection. The codebook is a vocabulary of local image regions that are characteristic for the appearance of an object class under different viewpoints. Local image descriptors are the best candidates for the codebook entries because they represent a specific structure of an object class that repeatedly occur in different viewpoints. Figure 2.3 shows how to train an appearance codebook of a person. Combining the codebook with a star topology, Leibe et al. [2007] achieved excellent levels of performance for detecting both rigid and articulated objects with a small amount of training data.

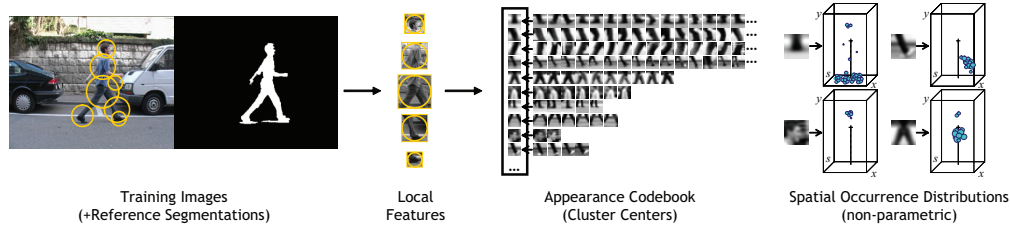
For a more enriched object representation, objects are often encoded by a collection of parts arranged in a deformable configuration. Such representation is called the pictorial



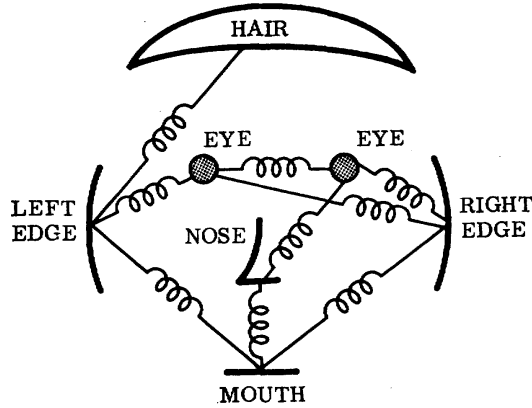


**Figure 2.2: Recognition results of SIFT [Lowe, 2004] ©2004 Springer.** This shows excellent recognition results by SIFT descriptors. Given training images on the left, the results of recognition are shown with parallelograms on separate test images that are taken from a different viewpoint. A set of detected key points are shown with small squares, rotated by the dominant orientation of each descriptor.

structure [Fischler and Elschlager, 1973]. Figure 2.4 shows a structural representation of objects by capturing local appearance configuration of parts and their spatial configuration based on spring-like connections between certain combinations of deformable parts. Many approaches based on the pictorial structure result in outstanding performance [Bourdev et al., 2010; Felzenszwalb et al., 2009; Felzenszwalb and Huttenlocher, 2005; Fergus et al., 2003].



**Figure 2.3: The training procedure of an appearance codebook from [Leibe et al., 2007] ©2007 Springer.** This shows the training procedure of an appearance codebook. First, local image features are extracted from an object. Then they are clustered to form the codebook. For each codebook entry, spatial occurrence distribution is learned with respect to the object centre.



**Figure 2.4: The parts of the pictorial structure model of [Fischler and Elschlager, 1973] ©1973 IEEE.**

### Dense representation

Deformable part models give an elegant representation of an object class, but it has been difficult to establish their value in practice due to their heavy computational cost for localizing objects and difficulties in training with such rich models. On difficult datasets where detectors require a large amount of training data to gain a reasonable performance, deformable part models are often outperformed by rigid templates [Dalal and Triggs, 2005]. Dalal and Triggs [2005] optimized rigid templates of HOG descriptors and considerably advanced levels of performance in human detection. Given enough training images, simple linear classifiers such as SVMs are quite effective in learning rigid templates for detecting

articulated objects such as pedestrians.

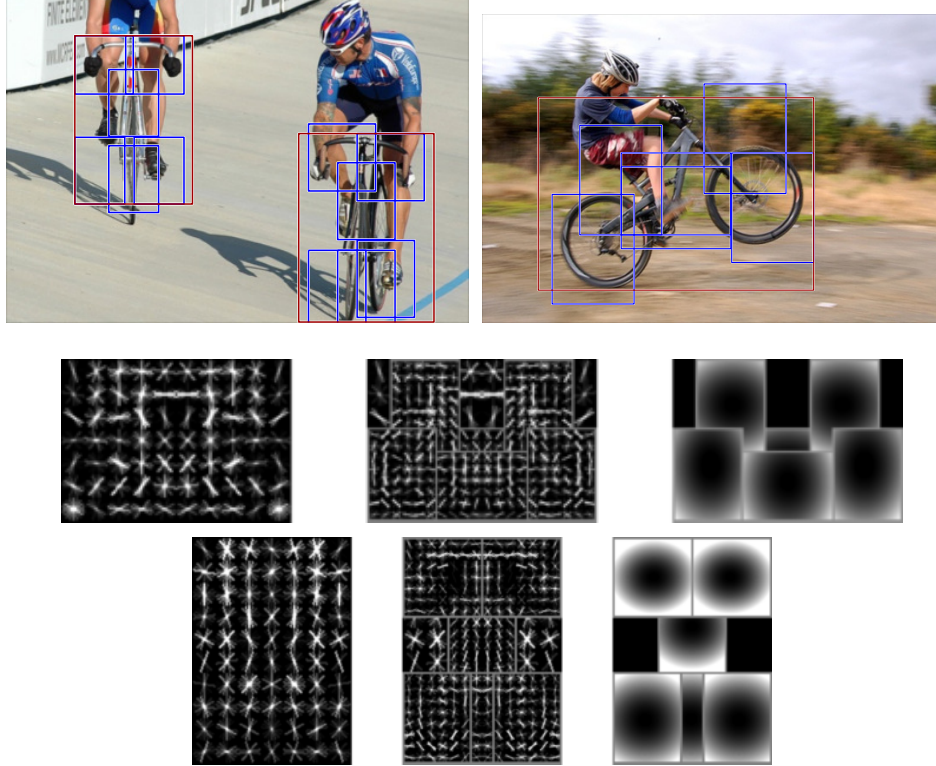
For detecting generic objects, however, a single holistic representation of rigid templates is still not expressive enough to represent rich object classes especially when the size of training data is limited. Recent state-of-the-art approaches therefore combine dense representations of objects with a rich expression of deformable part models to further improve performance for detecting generic objects [Bourdev et al., 2010; Felzenszwalb et al., 2009].

### **State-of-the-art**

Recently, Bourdev et al. [2010]; Felzenszwalb et al. [2009] further advanced the state-of-the-art in human detection by combining both dense and sparse representations. Felzenszwalb et al. [2009] used HOG descriptors for representing each part of objects and used pictorial structures for making parts to be deformable. Combining a set of multiple deformable models with a latent variable formulation of support vector machines, Felzenszwalb et al. [2009] have become a winner of the PASCAL object detection challenge over multiple object categories. Figure 2.5 shows detection results of the bicycle category from the PASCAL data.

Bourdev et al. [2010] proposed more flexible, yet more complex part models called poselets. Poselets are clustered based on geometric configurations (e.g., joints of a person) of object parts and their appearance variations in terms of HOG descriptors. While the Felzenszwalb et al. detector requires a predefined number of parts and mixture models, the poselet framework derives the number of parts and their spatial configurations automatically from training data. Since the Dalal and Triggs detector, dense representations of an object class have shown their value in problems for generic object detection. Deformable part models and poselets are natural consequences of increasing complexity from rigid templates.

The progression of state-of-the-art detectors in recent years has shown that there is a trade-off between increasing complexity in models and difficulties in learning and inference. Careful decisions must be made in order to enrich models without sacrificing performance and losing efficiency. The most complex model among state-of-the-art object detectors is one by Vedaldi et al. [2009]. It learns an optimal combination of non-linear kernels each of which captures a different feature representation of an object class. These features include the distribution of edges, visual words and feature descriptors such as HOG



**Figure 2.5: Deformable part model** From [Felzenszwalb et al., 2009] ©2009 IEEE. The top row shows detection results of the Felzenszwalb et al. [2009] detector. A mixture of its deformable part models are shown below. The first one in the second row represents sideways view of bicycles, while the second one in the third row captures the frontal view. Deformation parameters are shown on the right where lighter regions mean more deformation penalties.

and local self-similarity. The Vedaldi et al. detector is a joint winner of the PASCAL object detection challenge in 2009 with the Felzenszwalb et al. detector. However, the Vedaldi et al. detector sacrificed computational costs for improved performance. As a result, their detector runs an order of magnitude slower than the Felzenszwalb et al. detector and even slower than that of [Bourdev et al., 2010].

## 2.2 Object detection

Object detection is a task of localizing and categorizing any instances of an object class in an image. To achieve this, object detectors need to accurately represent an object model and efficiently perform exhaustive search for determining tight bounding windows around objects. Section 2.1 explains the representation of an object class. This section focuses on object localization and introduces related works in the area.

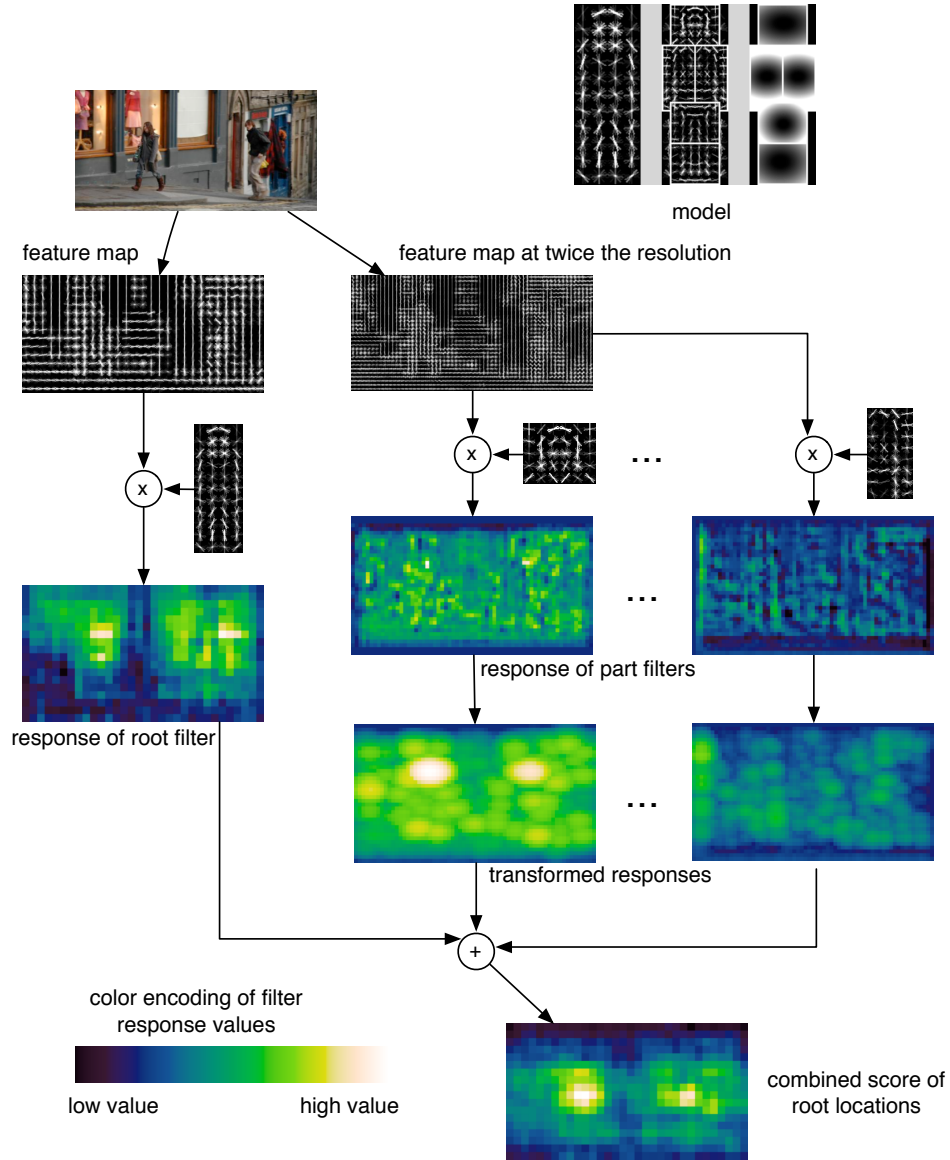
### 2.2.1 Localizing objects

Given a newly observed image  $I$  and a trained object model  $\Theta$ , the task of object localization is formulated as a search problem:

$$R^* = \operatorname{argmax}_{R \subseteq I} f(R|I, \Theta) \quad (2.1)$$

where  $R$  ranges over all bounding windows (often rectangular or ellipsoidal shapes) in the image  $I$ . Then the goal is to find the best bounding window  $R^*$  based on a score function  $f$ . Many approaches have been proposed for solving localization of objects. Broadly speaking, they are divided in two categories: sliding window approaches and the Hough transform approaches. The former has been widely adopted in various approaches [Chum and Zisserman, 2007; Dalal and Triggs, 2005; Deselaer and Ferrari, 2010; Felzenszwalb et al., 2009; Mutch and Lowe, 2006; Viola and Jones, 2004].

Sliding window approaches consider all possible sub-windows of an image and use a classifier to evaluate them on whether they contain an instance of an object class of interest. Depending on the resolution of an image, a large set of candidate sub-windows require millions, if not billions, of evaluations. Therefore, efficiency becomes an important issue for building an object detector that is even feasible to run in practice. Most sliding window approaches to date have used either linear classifiers [Dalal and Triggs, 2005; Felzenszwalb et al., 2009; Mutch and Lowe, 2006], a highly efficient non-linear approach [Maji et al., 2008] or an efficient branch-and-bound search [Lampert et al., 2008]. Figure 2.6 shows the process of searching the location of parts (“head” and “right shoulder” in this case) of a person in an image at one scale and aggregating all responses to show two good hypotheses. To detect two people in the image, the localization process is performed at all possible scales from which millions of sub-windows are evaluated.



**Figure 2.6: Localization process of the deformable part model from [Felzenszwalb et al., 2009] ©2009 IEEE.** This shows the localization process at one scale by a person model.

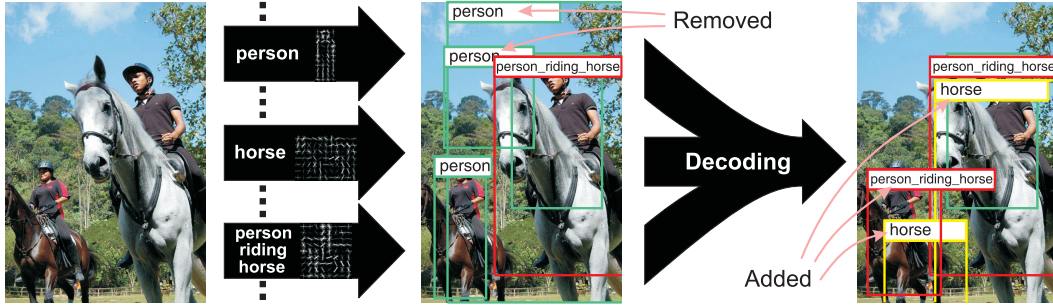
Other methods are based on the Hough transform [Ballard, 1981; Leibe et al., 2007; Maji et al., 2008; Opelt et al., 2006b]. Ballard [1981] extended the Hough transform for detecting lines to the generalized Hough algorithm for detecting generic shapes. Leibe et al. [2007] extended the idea of the generalized Hough transform in object categorization and proposed the implicit shape model that specifies distinctive characteristics of the object in terms of the codebook entries. They used a bottom-up localization approach that extracts relevant local features each of which casts probabilistic votes for possible object locations. Such feature-based localization idea has been adopted in several approaches [Maji et al., 2008; Opelt et al., 2006b].

Most of the existing work on object localization falls into either a sliding window or the Hough transform approach except for Lehmann et al. [2009] that recently combined the ideas of the efficient sub-window search [Lampert et al., 2008] and the implicit shape model in their localization framework.

### **2.2.2 Using context**

The task of object localization is formulated as a search problem in Equation 2.1 where the success largely depends on evaluations of candidate hypotheses based on a score function  $f$ . Many successful approaches used classification algorithms for scoring each hypothesis. Therefore, learning a good score function is critical for designing an optimal object detector. In localizing objects from a single object class in videos, motion information of objects such as optical flow or directional motion filters are used as additional cues to be combined with appearance information of objects for a better performance [Dalal, 2006; Viola et al., 2005]

In more general cases of object localization from multiple different classes, several approaches explored the use of additional contextual information of the image, which is modelled by the relationships of different objects or different object classes [Desai et al., 2009; Murphy et al., 2003; Sadeghi and Farhadi, 2011; Sudderth et al., 2005]. For example, Sadeghi and Farhadi [2011] recently proposed visual phrases that encapsulate visual relations of objects and showed that appearance information of objects in some interaction drastically improve detections of individual objects. Figure 2.7 shows how a visual phrase, “person riding horse,” helps improving performance of detecting objects from two different classes.



**Figure 2.7: Visual phrase from [Sadeghi and Farhadi, 2011] ©2011 IEEE.**

Combining a visual phrase (“person riding horse”) with object models, the resulting predictions remove false positives and even add true positive detections that have never been discovered before.

## 2.3 Learning

In designing a visual system for recognizing object categories, we need to address three main issues: object representation, detection, and learning. Previous sections cover the first two issues. This section explains the issue of learning, which is perhaps the least well understood among them. Object localization requires a model to have a score function that can identify bounding windows for the object in each image — there can be zero, one or many such boxes in a single image. Such a model is often learned by machine learning techniques. However, there are many challenges that need to be dealt with when a large search space of bounding windows includes background clutter, occlusions, illumination changes, and intra-class variations of the appearance of objects.

The ultimate goal of this learning task is to identify an invariant structure of object representation that can handle these difficulties. To achieve this, a model often requires a large amount of training data and needs to be not too complex for reasons of efficiency. A significant body of work on learning a model for generic object detection usually focuses on adding more complexity to models. Aspects of training data with relatively simple models have not been explored as much. We summarize the existing work in the area and focus on issues of labelled data and its requirement for human labelling effort rather than learning techniques.

Figure 2.8 shows related machine learning formalisms in the context of detecting koalas and pandas. Images on the black background are labelled with localization windows. Ones

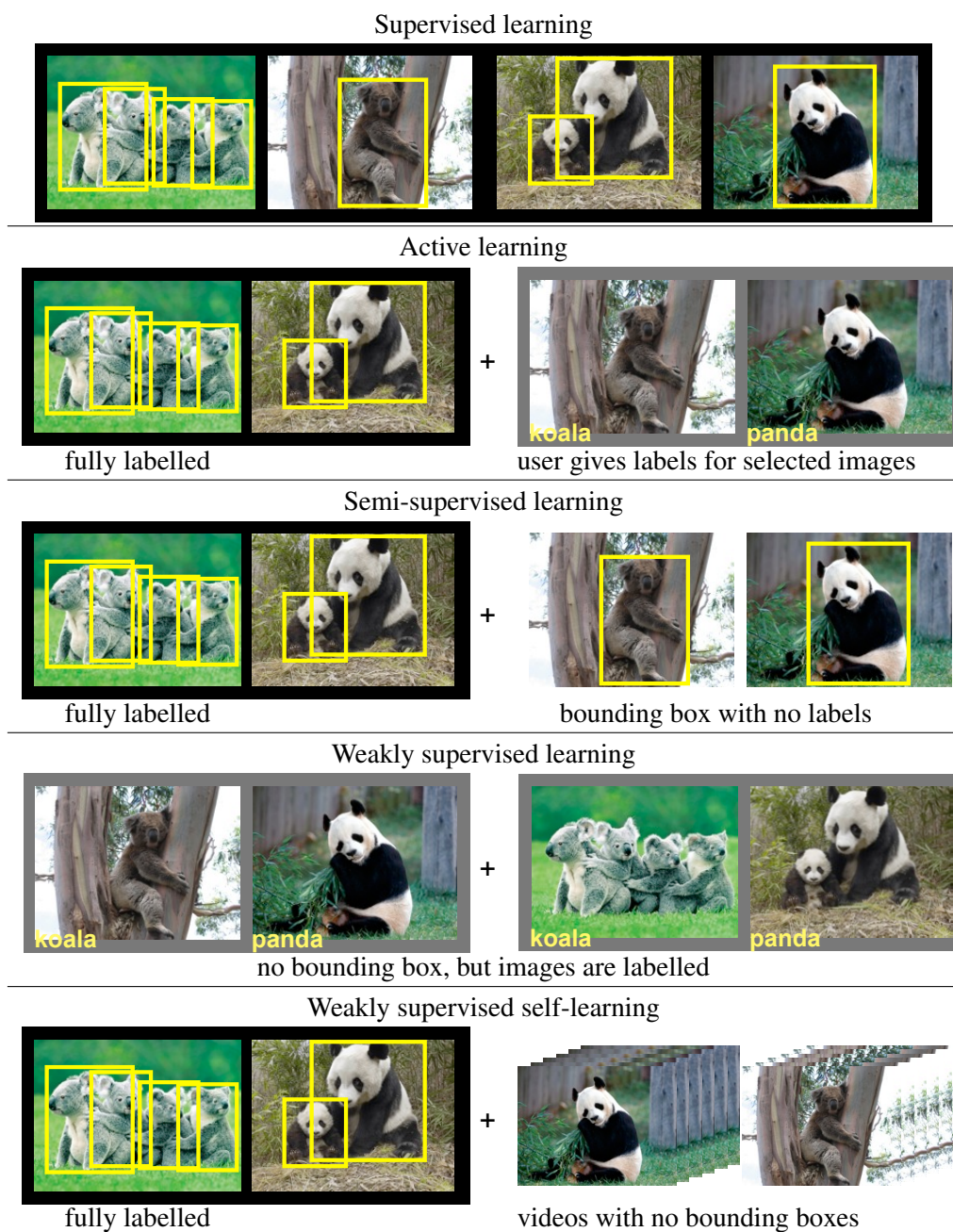


on the grey background are weakly labelled without localization windows. Others are unlabelled. The data on the left column is the initial set of data that is used for training an initial model. The data on the right column is additional data that is easier to obtain and can be used for additional training. Supervised learning uses labelled instances (i.e., bounding boxes in yellow) of koalas and pandas. There is no additional data for this scenario since only labelled data are given. Active learning uses a small set of labelled instances to train an initial model. Labels for an additional unlabelled instances are given by an oracle which is often a human observer or rarely an intelligent machine. Semi-supervised learning uses additional unlabelled bounding windows of koalas and pandas. Weakly supervised learning does not have additional bounding windows, but images that contain instances of koalas and pandas. One can view this as semi-supervised learning with additional constraints such as the category label for an image. Weakly supervised self-learning requires additional frames from videos. In this case, the category label of a video is given where any number of target objects are allowed to be present throughout the video.

### 2.3.1 Supervised learning

Supervised learning deals with a labelled dataset  $\mathcal{L} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l)\}$  where  $\mathbf{x}_i \in \mathcal{X} = \mathbb{R}^M$  is an  $M$ -dimensional feature descriptor and  $y \in \mathcal{Y}$  is a label. The goal is to learn a classifier  $H : \mathcal{X} \mapsto \mathcal{Y}$ . Many supervised learning models have been tried including SVMs, neural network, randomized trees or boosting [Chum and Zisserman, 2007; Dalal and Triggs, 2005; Deselaer and Ferrari, 2010; Felzenszwalb et al., 2009; Lepetit and Fua, 2006; Mutch and Lowe, 2006; Papageorgiou and Poggio, 2000; Rowley et al., 1998; Vedaldi et al., 2009; Viola and Jones, 2004]. A predominant number of approaches formulate object detection as the binary classification problem with  $\mathcal{Y} = \{+1, -1\}$ . A label  $y_i$  is  $+1$  when a feature vector  $\mathbf{x}_i$  represents an instance of the target object class or  $-1$  for an instance of all encompassing background class.

The classification performance of machine learning methods for object recognition can be improved by simply providing more training instances. An image may have one or more instances of an object class. Viola and Jones [2004] showed that object detectors require millions of training instances from hundreds of thousands of images to get reasonable performance for detecting a single object class such as faces and pedestrians in practice. The annotations of the training data for building an object detector are usually provided in a set



**Figure 2.8: Related machine learning formalisms for detecting koalas and pandas.** Images on the black background are labelled. Ones on the grey background are weakly labelled. Others are unlabelled. Refer to the text for more explanation.

of bounding windows  $\{R_1, \dots, R_l\}$  that determine the size and location of all instances of the target object class in a set of  $n$  images  $\{I_1, \dots, I_n\}$ . For each bounding window  $R_i$ , a feature vector  $x_i$  is computed in the form of an image descriptor.

The two largest public datasets<sup>1</sup> are the MIT LabelMe data [Russell et al., 2008] and the ImageNet [Deng et al., 2009]. The MIT LabelMe data is an open database of images where public users can submit their annotations through a web-based annotation interface. The LabelMe provides a large set of bounding polygons over 3,000 object categories, which is definitely one of the largest publicly available datasets today for general object categorization. The ImageNet is a more recent publicly available dataset that is even larger than the MIT LabelMe data. It is an image dataset that is organized based on the WordNet hierarchy. In the hierarchy, there are more than 100,000 synonym sets or “synsets”, each of which is described by multiple words or word phrases. The ImageNet provides 1000 images on average for over 17,000 synsets, which sums to over 12 million images in total. The ImageNet has become the largest dataset today for image classification problems and has been used for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) annually since the year of 2010. The 12 million images, an unprecedented scale of annotation data, which includes more than 657,000 images with bounding box annotations, were collected by crowdsourcing through Amazon Mechanical Turk. This subset comprises over 3,000 synsets, each of which has on average 150 images with bounding boxes. However, the object localization challenge, based on these bounding box annotations, is still preliminary because it is held as a “taster” competition according to the organizers of the ILSVRC competitions.

These two large datasets are still not ideal for solving object detection problems. Web annotations in the MIT LabelMe are still *incomplete* because many images are not fully labelled. That is, these *incomplete* images have both labelled and unlabelled objects: for example, an image contains two cars and a person, where a car is labelled with a bounding box, but no annotated bounding box is given to a person or another car in the same image. The ImageNet classification challenge with localization is still *preliminary* in a “taster” stage. With these datasets, it is difficult to evaluate performance of object detectors in the level of precision of the PASCAL data which is a smaller, yet more complete dataset for object detection. The PASCAL data has provided, annually since the year of 2005, a

---

<sup>1</sup>For pedestrian detection, [Dollár et al., 2009] introduced the Caltech Pedestrian Dataset. It contains very large data of approximately 10 hours of 30Hz video (  $10^6$  frames, a total of 350,000 bounding boxes). They provide benchmark results on performance of the state-of-the-art detectors.

complete dataset of about 10,000 images with approximately 25,000 bounding boxes of objects. The PASCAL data has 20 different object categories where each category contains a varying number of instances of objects, ranging from a few hundred to several thousands.

Realistically, the PASCAL dataset is not large enough to attain a proficient level of performance with supervised learning models. With the abundance of images and videos readily available on the Web and through digital devices such as digital cameras and smart-phones, adding extra training data is a straightforward means of overcoming this lack of data. But acquiring extra labels is not so easy because labelled data often involves time-consuming human labour. Therefore, we will consider several other related learning scenarios that help to reduce the overall labelling effort.

### 2.3.2 Active learning

Active learning refers to any situation in which the learning algorithm has control over the inputs on which it trains, querying an oracle to provide labels for selected instances. Active learning is quite different from other learning scenarios because it often employs a human oracle for label acquisition. To implement active learning, the algorithm is usually provided with a small set of labelled data  $\mathcal{L} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l)\}$  to learn an initial model  $H$ . The model  $H : \mathcal{X} \mapsto \mathcal{Y}$  is then iteratively updated with additional label information  $\{y_{l+1}, \dots, y_{l+m}\}$  from a human oracle who answers queries that are selected from a much larger set of unlabelled data  $\mathcal{U} = \{\mathbf{x}_{l+1}, \dots, \mathbf{x}_{l+m}\}$  where a feature vector  $\mathbf{x}_i \in \mathcal{X}$ , a label  $y_i \in \mathcal{Y}$  and  $l \ll m$ . In this learning process, the total labelling effort is determined by the amount of queries a human oracle has to answer and the size of the initial set of fully labelled data.

In recent years, active learning in computer vision has gained much attention. It has been used in tracking [Hewitt and Belongie, 2006], handwritten digit identification [Zhu et al., 2003], and visually-guided robots [Mahdavian et al., 2005]. Outside of the vision community, active learning has found success in domains such as customer support classification [Tur et al., 2005], text classification [Schohn and Cohn, 2000; Tong and Koller, 2001; Zhu et al., 2003], robot kinematics [Cohn et al., 1996], and sensor placement [Guestrin et al., 2005]. While various active learning extensions have also been proposed in image classification [Collins et al., 2008; Kapoor et al., 2007; Qi et al., 2008; Vijayanarasimhan et al., 2010; Zhang et al., 2008], relatively little attention has been paid to the more chal-

lenging problem of object localization [Abramson and Freundndom, 2005; Siddiquie and Gupta, 2010; Vijayanarasimhan and Grauman, 2011; Vijayanarasimhan and Kapoor, 2010]. Object localization requires a model that can identify bounding windows for the object in each image — there can be zero, one or many such boxes in a single image — whereas image classification needs only a single global classification per image. Applying active learning to localization requires a substantial change in approach, as a single label cannot be applied to an image — it must be applied to potentially numerous overlapping windows of an image.

Therefore most active learning approaches have not been tested for object localization problems even in a relatively small scale dataset such as the PASCAL 2007, which has around 10,000 images. However, active learning can be very powerful when there are abundant unlabelled data. [Okuma et al., 2011] became the first work to test an active learning approach on the PASCAL 2007 and improve object detection models of [Felzenszwalb et al., 2009] by actively selecting a small portion (10%) of image data that consist of over 60,000 images downloaded from Flickr image search. This work is introduced in Chapter 4. Later in the same year, Vijayanarasimhan and Grauman [2011] combined active learning and crowdsourcing to efficiently scale up training sets for building an object detector. The framework they proposed enables an interactive training of an object detector online by actively requesting annotations from Amazon Mechanical Turk. The process is repeated multiple times to refine their part-based object models. Their approach is the first large scale study that attempts to train an object detector with crowdsourced annotations from the Web.

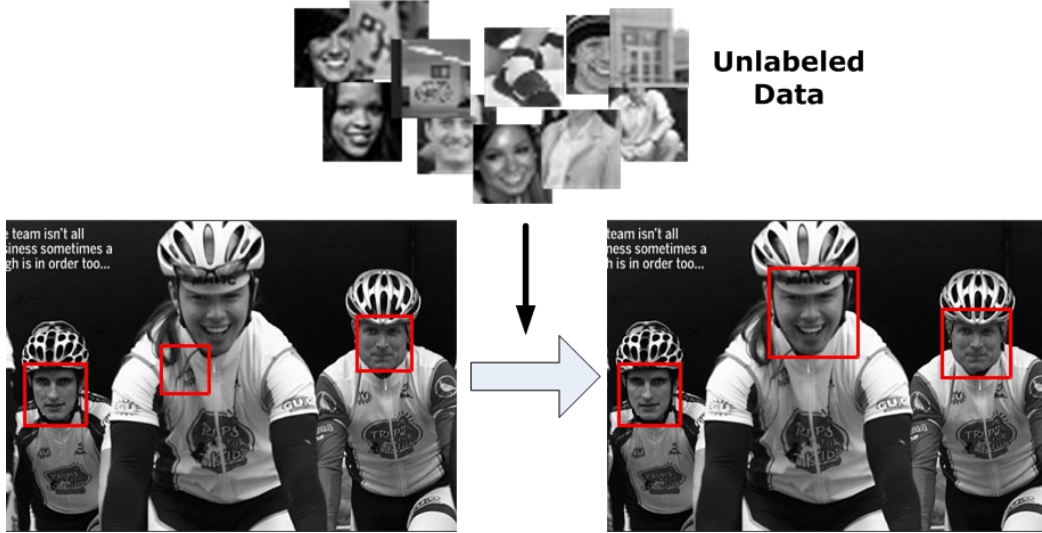
### 2.3.3 Semi-supervised learning

Semi-supervised learning is a means for reducing the labelling effort, which exploits both labelled and unlabelled data for efficiently training a classifier. The algorithm is provided with a small set of fully labelled data  $\mathcal{L} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l)\}$  and an additional set of unlabelled data  $\mathcal{U} = \{\mathbf{x}_{l+1}, \dots, \mathbf{x}_{l+m}\}$  where a feature vector  $\mathbf{x}_i \in \mathcal{X}$ , a label  $y_i \in \mathcal{Y}$  and  $l \ll m$ . Once an initial classifier  $H$  is trained with the labelled dataset  $\mathcal{L}$ , the classifier is then used to prepare additional labels  $\{y_{l+1}, \dots, y_{l+m}\}$  from a much larger dataset  $\mathcal{U}$  and iteratively updated with these additional data in order to learn a mapping  $H$  from  $\mathcal{X}$  to  $\mathcal{Y}$ . There is a vast amount of literature on methods of semi-supervised learning, which

originally dates back to the work of [Scudder, 1965]. In recent years, semi-supervised learning approaches have succeeded in various domains of problems such as text classification [Nigam et al., 2000], handwritten digits recognition [Lawrence and Jordan, 2005] and object detection [Leistner et al., 2007; Rosenberg et al., 2005]. The literature review of semi-supervised learning approaches in various problem domains is out of scope of this thesis. We recommend a book by [Chapelle et al., 2006] and a comprehensive literature survey by [Zhu, 2008].

In the context of object detection, semi-supervised learning is provided with both labelled and unlabelled localization windows that contain an instance of either an object class or all encompassing background class. Rosenberg et al. [2005] implemented a semi-supervised learning approach with a boosting algorithm in order to reduce the manual labelling effort for training a model for human eye detection. Their approach uses self-learning which is perhaps the most traditional approach for semi-supervised learning [Chapelle et al., 2006]. It is a wrapper algorithm that starts by a small set of labels to train the initial model. The model is then used to evaluate unlabelled data, being retrained with a selected portion of its own predictions as additional labels. Their selection metric uses the Mahalanobis distance between the wavelet transform of a candidate bounding window and that of all labelled instances. Such a similarity measure for selecting additional training data from a pool of unlabelled data is less biased than the detection confidence and is shown effective in self-learning. To the best of our knowledge, their work is the first comprehensive scale study of semi-supervised learning in object detection.

As designing a similarity function for measuring the distance between labelled and unlabelled samples is a fundamental requirement for semi-supervised learning approaches, Leistner et al. [2007] used boosting to learn a similarity function between labelled and unlabelled samples. Their approach is based on SemiBoost [Mallapragada et al., 2007] which uses boosting in semi-supervised learning setting and guides the learning process by the pairwise similarities of training data. Their SemiBoost framework has been used to effectively improve a state-of-the-art face detector with unlabelled data. Figure 2.9 demonstrates how an additional unlabelled localization windows can improve a trained detector by increasing the detection rate, reducing the false positive rate or a better alignment of localization windows.



**Figure 2.9: Semi-supervised learning for detecting faces from [Leistner et al., 2007] ©2007 IEEE.** This shows how a semi-supervised learning system of [Leistner et al., 2007] improves a face detector with an additional set of unlabelled localization windows.

### Weakly-supervised learning

Weakly-supervised learning in object categorization refers to a scenario where training images are provided without the location of object instances. Broadly speaking, one can view this learning scenario as semi-supervised learning with additional constraints such as the presence of object instances in an image. Let's denote a weakly labelled dataset  $D = \{(I_1, y_1), \dots, (I_n, y_n)\}$  where a label  $y_i \in \mathcal{Y} = \{+1, -1\}$  is  $+1$  when an image  $I_i$  contains one or more instances of the target object class or otherwise  $-1$ . In contrast to semi-supervised learning that exploits localized bounding windows with no category labels, weakly-supervised learning requires accurate localization of bounding windows for assigning the category label to each bounding window. For each image  $I_i$ , there can be zero, one or more feature descriptors  $x_j$  with a label  $y_i$  where  $x_j \in \mathcal{X}$  and  $y_i \in \mathcal{Y}$ . Once localization windows are computed, the goal of weakly-supervised learning is the same as in supervised learning: to learn a classifier  $H : \mathcal{X} \mapsto \mathcal{Y}$ .

The amount of labelling effort without localization of bounding windows could still be (much) cheaper than preparing unlabelled bounding windows. For further reducing the

overall human labelling effort of large data, weakly-supervised learning has become a major topic of research in recent years [Arora et al., 2007; Deselaer et al., 2010; Fergus et al., 2007; Galleguillos et al., 2008; Nguyen et al., 2009]. In general, weakly labelled data is referred as a set of images where each image has one or more instances of an object class. But most of work has the very stringent assumption of only one instance in an image mainly for avoiding difficult localization issues. For achieving localization with only one instance of an object of interest in an image, some approaches use the sparse representation of parts of an object [Crandall and Huttenlocher, 2006; Fergus et al., 2007], segmentations [Arora et al., 2007; Galleguillos et al., 2008; Winn et al., 2005] or bag-of-words with spatial pyramid matching [Chum and Zisserman, 2007].

For more than one instance of an object of interest in an image, [Deselaer et al., 2010] used a conditional random field (CRF) to simultaneously localize object instances and learn an appearance model for an unknown object class. The CRF exploits localization results from other object classes and circumvents localization of the target object class by guiding the selection of localization windows. The model has been shown effective in a challenging dataset, the PASCAL 2007 Visual Object Classes [Everingham et al., 2010].

### Weakly-supervised self-learning

The previous learning scenarios assume that the unlabelled instances are independent. However, the data in video sequences are strongly related due to spatio-temporal dependencies in videos. In object detection, the task is to find one or more bounding windows, each of which precisely determines the location and size of an instance of an object class in an image. The location of the bounding windows in a video defines a trajectory of an object instance. The trajectory represents a spatio-temporal structure of subsequent labels in a video sequence.

Here, we consider the following scenario: Given sparsely labelled video data that consists of  $n$  different video sequences  $\{V_i\}_{i=1}^n$ , the task is to train an initial model  $H : \mathcal{X} \mapsto \mathcal{Y}$  from a small set of labels  $\mathcal{L} = \{(x_1, y_1), \dots, (x_l, y_l)\}$  and exploit the rest of unlabelled data  $\mathcal{U} = \{x_1, \dots, x_m\}$  for improving the model, assuming that  $x \in \mathcal{X}$ ,  $y \in \mathcal{Y}$ , and  $l \ll m$ . Assuming that there are an unconstrained number of instances of the target object class throughout these videos, ones can view this problem as a semi-supervised learning scenario with weakly labelled videos and their category labels. Therefore, we can consider



this as a scenario of weakly-supervised self-learning.

For exploiting the structure of the video data, several tracking-by-detection approaches [Babenko et al., 2009; Kalal et al., 2010; Leistner et al., 2011] have been proposed to learn an appearance model of an object instance from videos. Tracking-by-detection approaches take detections as input and produce tracking results as output. However, these approaches have the stringent assumption of having only one instance of the target object class in each frame of a video sequence. Such an assumption strictly limits real-world applications to detection of only a single instance of the target object class where the detection with the maximal confidence is identified as a positive label and all remaining instances are negative. In order to detect multiple instances of an object class such as pedestrians or faces, videos that contain multiple people and thus a dense collection of potential labels are crucial to the training dataset. But localization of multiple target objects remains too difficult for most of self-learning tracking-by-detection approaches. There are a few approaches that have addressed the problem of exploiting the unlabelled video data with multiple target objects.

Ramanan et al. [2007] proposed a semi-supervised method for building a large collection of labelled faces from archival video of the television show “Friends.” Their final collection contains 611,770 faces, which is the largest existing collection of faces to date in academia. Their approach used the Viola and Jones face detector to detect faces, grouping them with colour histograms of body appearance (i.e, hair, face, and torso) and tracking them using part-based colour tracking for multiple people in videos. Although their approach is effective in a large scale data, the approach performed only one iteration of exploring the unlabelled data for building a large collection of faces and never used the acquired collection for improving classifiers they used.

Very recently, [Ali et al., 2011] implemented self-learning on sparsely labelled videos, which allows any number of instances of the target object class. Their approach exploits spatio-temporal information of objects for improving an appearance-based object detector. Given a sparse labelling of the video sequence, an initial model is trained by a boosting algorithm with a small set of labelled instances. The rest of unlabelled portion of the video is used for collecting additional labels that are consistent with the current classification model and a constraint of continuous motion of target objects. After a few iterations of this process, they showed noticeable performance improvement on pedestrian detection in videos and cell detection in microscopy image sequences. To the best of our knowledge, the work of [Ali et al., 2011] has been the first work that addresses localization of multiple

target objects in videos for improved object detection.

Their work is the most related to our work which is explained in Chapter 6. However their approach differs significantly from ours. They used a boosting algorithm to exploit the temporal coherence of videos, whereas we adopt the latent SVM formulation for learning the appearance of objects in our self-learning framework and use figure-ground segmentation as an additional information to validate the unlabelled data. Furthermore, we use broadcast footage of sports which is much more challenging than their surveillance data of a stationary camera view where pedestrians are walking along with very simple, predictable motions.

## Chapter 3

# Active Learning for Image Labelling

It is possible to improve the classification performance of machine learning methods for object recognition by simply providing more training instances. However, the cost of providing labelled data — asking a human to examine images and provide labels — becomes impractical as training sets grow.

In addition, object localization with bounding windows is important for many applications and plays a significant role in improved performance [Lazebnik et al., 2006; Mutch and Lowe, 2006; Viola and Jones, 2004], but this also places an increased burden on human labelling by requiring accurately aligned bounding windows around each training instance.

We present a method for using active learning [Settles, 2010; Tong and Koller, 2001] to reduce the required number of labelled training instances to achieve a given level of performance. Our method also proposes optimal bounding windows aligned with the current classification function. Active learning refers to any situation in which the learning algorithm has control over the inputs on which it trains, querying an oracle to provide labels for selected images.

In particular, we are concerned with the problem where a person is required to provide labels for bounding windows. Unlike image classification where the image is classified as a whole, we apply this approach to *object localization* by iteratively selecting the most uncertain unlabelled training windows from a large pool of windows, and present them to the user for labelling. If the object of interest is correctly bounded by the localization window, the human provides a positive label, otherwise they provide a negative label or indicate their uncertainty. In a case of multiple overlapping candidate windows, we apply

non-maximum suppression to select the best candidate by suppressing ones with a lower confidence. We have designed a user interface that is efficient for the user and provides feedback on performance while also enabling the user to retain some initiative in selecting image windows.

Our experimental results apply the approach to the PASCAL Visual Object Classes 2007 (VOC2007) [Everingham et al., 2007] on twenty different categories learned with a mixture of multi-scale deformable part models from Felzenszwalb et al. [2009], one of two joint winners in the VOC2009 challenge for object detection problems. In the first experiment, we show that we can match their results on VOC2007 data while using many fewer labelled windows for training. Then we demonstrate our full user interface on more diverse images, such as those obtained from web search engines and show how it enables a user to efficiently improve performance on VOC2007. While these experiments show that our actively trained deformable part model works well with active learning, the system does not depend on a specific classifier or feature set. If other classifiers or features are found to be more suitable to a domain, we can also incorporate them into our framework.

### 3.1 Active learning for image labelling

Image labelling requires human effort to supply the labels for each positive window in the training set. This typically involves thousands of instances, and is thus extremely labour-intensive. To make object recognition practical over multiple domains, we need to recognize that human labelling is a costly resource, and must be used as efficiently as possible. Active learning is the machine learning method typically used in this case. A full discussion of active learning is beyond the scope of this thesis, so we will focus only on the most relevant areas for our work. We direct the interested reader to, *e.g.*, the recent review by Settles [Settles, 2010] for a more extensive survey.

In *pool-based active learning*, which we use here, candidate data are selectively sampled for labelling from a pool of unlabelled data  $\mathcal{U}$ . The model is then updated to take the new labels into account, and new candidates selected. Candidate selection is typically performed by greedily selecting data  $\mathbf{x}^*$  from  $\mathcal{U}$  according to some utility function,  $\mathbf{x}^* = \operatorname{argmax}_{\mathbf{x} \in \mathcal{U}} U(\mathbf{x})$ . The process repeats as long as the human labeller is willing, or until some other termination condition is reached.

The candidate selection objective function is designed to maximize the utility of the

labels gathered, typically by some measure of the expected informativeness of the labels. This is what makes active learning efficient in situations where data is plentiful but labelling it is expensive. By identifying the training instances that actually impact the model, fewer labels are required, and the bottleneck can be reduced. In Section 3.3, we use an *uncertainty sampling* utility function [Lewis and Gale, 1994], which is popular as it performs well on a variety of problems and has been extensively studied. Other methods include maximizing entropy, combining a committee of models, or maximizing the expected model change.

In recent years, active learning in computer vision has gained much attention. While various active learning extensions have been proposed in image classification [Collins et al., 2008; Kapoor et al., 2007; Qi et al., 2008; Vijayanarasimhan et al., 2010; Zhang et al., 2008], relatively little attention has been paid to the more challenging problem of object localization [Abramson and Freundndom, 2005; Siddiquie and Gupta, 2010; Vijayanarasimhan and Kapoor, 2010]. Object localization requires a model that can identify bounding windows for the object in each image—there can be zero, one or many such boxes in a single image. Image classification needs to consider only a single global classification per image. Applying active learning to localization requires a substantial change in approach, as a single label cannot be applied to an image—it must be applied to potentially numerous and overlapping windows of an image. Most active learning approaches are therefore infeasible for object localization problems in even a relatively large scale dataset such as the VOC2007 dataset of around 10,000 images. To the best of our knowledge, we are the first to apply and test active learning performance on the PASCAL or similar datasets.

## 3.2 Algorithm

Our system for active learning uses the Support Vector Machine (SVM) [Schölkopf and Smola, 2002] classifier, which has proven its success in many state-of-the-art recognition systems [Dalal and Triggs, 2005; Moosmann et al., 2006; Mutch and Lowe, 2006; Zhang et al., 2006a]. We also incorporate the recent latent SVM (LSVM) approach of Felzenszwalb *et al.* [Felzenszwalb et al., 2009]. We will briefly review these models and describe how we incorporate active learning. The goal of a supervised learning algorithm is to take  $n$  training samples and design a classifier that is capable of distinguishing  $M$  different classes. For a given training set  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$  with  $\mathbf{x}_i \in \mathbb{R}^N$  and  $y_i \in -1, +1$  in their simplest form with two classes, LSVM is a classifier that scores a sample  $\mathbf{x}$  with the following

function [Felzenszwalb et al., 2009],

$$f_{\beta}(\mathbf{x}) = \max_{\mathbf{z} \in \mathbf{Z}(\mathbf{x})} \beta \cdot \Phi(\mathbf{x}, \mathbf{z}) \quad (3.1)$$

Here  $\beta$  is a vector of model parameters and  $\mathbf{z}$  are latent values. The set  $\mathbf{Z}(\mathbf{x})$  defines possible latent values for a sample  $\mathbf{x}$ . Training  $\beta$  then becomes the following optimization problem.

$$\min_{\beta, \xi_i \geq 0} \quad \frac{1}{2} \|\beta\|^2 + c \sum_{i=1}^n \xi_i \quad (3.2)$$

$$s.t. \quad \forall i \in \{1, \dots, n\} : y_i f_{\beta}(\mathbf{x}_i) + b \geq 1 - \xi_i \quad (3.3)$$

$$\text{and } \xi_i \geq 0 \quad (3.4)$$

where  $c$  controls the tradeoff between regularization and constraint violation. For obtaining a binary label for  $\mathbf{x}$ , we have the decision function,  $sign(h(\mathbf{x}))$ , where

$$h(\mathbf{x}) = f_{\beta}(\mathbf{x}) + b \quad (3.5)$$

In general, a latent SVM is a non-convex optimization problem. However, by considering a single possible latent value for each positive sample that can be specified in training, it becomes a convex optimization problem for classical SVMs. For multi-scale deformable part models, the set of latent values represents the location of parts for the object.

### 3.3 Active learning on a latent SVM

We incorporate active learning into an LSVM by at each iteration selecting the candidate that has the minimum distance to the decision boundary. That is, for a set of data  $\mathcal{C} \subseteq \mathcal{U}$ , where  $\mathcal{U}$  is the set of unlabelled image windows, we select:

$$\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x} \in \mathcal{C}} \left| \frac{h(\mathbf{x}_i)}{\|\beta\|} \right| \quad (3.6)$$

where  $\mathbf{x}^*$  is the candidate datum we ask the human expert to label. Ideally, we then move the new labelled datum from the candidate to the training set, update the LSVM model, and repeat, selecting a new candidate based on the updated model. However, due to the heavy

computational load of LSVM, we choose to update the model in a batch after collecting a set of new labelled data. In our experiments, we obtain a set of unlabelled image windows by applying a sliding window at multiple scales on the image and computing a Histogram of Oriented Gradients (HOG) descriptor [Dalal and Triggs, 2005; Felzenszwalb et al., 2009] from each window as our datum.

This *uncertainty sampling* utility seeks to label the most uncertain unlabelled datum to maximize the margin. There are known to be other successful criteria for SVM active learning, particularly version space methods [Tong and Koller, 2001]. We use uncertainty sampling as we wish to remain agnostic as to the underlying model. It has also been shown to perform well and has the advantage of being fast to compute and easy to implement. The learning task then becomes an iterative training process as is shown in Algorithm 3.1.

---

**Algorithm 3.1 : Pool-based active learning algorithm for image window labelling**

---

$\mathbf{x}$  is a window descriptor,  $\mathcal{I}$  is a set of  $m$  images,  $\mathcal{L}$  is a set of labelled image windows and  $\mathcal{U}$  is a set of unlabelled windows.

---

```

1: {Collect a set of unlabelled image windows,  $\mathcal{U}$ }
2:  $\mathcal{U} = \emptyset$ 
3: for  $i \in \mathcal{I}$  do
4:    $\mathcal{U} = \mathcal{U} \cup \mathbf{x}$  where  $\mathbf{x}$  is the highest scoring detection window in image  $i$ 
5: end for
6: while the user continues the active learning session do
7:   {Obtain a set of new labelled data,  $\mathcal{C}, \mathcal{Y}$ }
8:    $\mathcal{C} = \emptyset$ 
9:    $\mathcal{Y} = \emptyset$ 
10:  for  $t = 1$  to  $n$  do
11:     $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x} \in \mathcal{U}} \left| \frac{h(\mathbf{x}_i)}{\|\mathbf{w}\|} \right|$ 
12:    Get the label  $y^*$  of the datum  $\mathbf{x}^*$  from the human expert
13:    The expert can add a set of  $k$  samples  $\mathcal{X} = \{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_k\}$  in case of miss
    detections or false positives with a set of associated labels  $\mathcal{L} = \{y_0, y_1, \dots, y_k\}$ 
    of  $\mathcal{X}$ 
14:     $\mathcal{C} = \mathcal{C} \cup \mathbf{x}^* \cup \mathcal{X}$ 
15:     $\mathcal{Y} = \mathcal{Y} \cup y^* \cup \mathcal{L}$ 
16:     $\mathcal{U} = \mathcal{U} - \mathbf{x}^*$ 
17:  end for
18:  Add  $\mathcal{C}$  and  $\mathcal{Y}$  to the training set and update the LSVM model,  $\hat{\beta}, \{\hat{\mathbf{x}}_l, \hat{\mathbf{z}}_l\}_{l=1}^m$  and  $\hat{b}$ 
  where  $m$  is the number of support vectors
19: end while

```

---

### 3.4 The PASCAL visual object classes challenge 2007

In this section, We will test our active learning approach against a state-of-the-art object detector on VOC2007 [Everingham et al., 2007] to show the ability to reduce labelling requirements without sacrificing performance. The PASCAL Visual Challenge has been known as one of the most challenging testbeds in object recognition. The VOC2007 dataset contains 9,963 images: 50% are used for training/validation and the other 50% are used for testing. Figure 3.1 shows a set of randomly selected training images with ground truth annotations. We used VOC2007 to test our active learning approach to see if it would reduce labelling requirements for positive data. We use our own Python implementation of Felzenszwalb *et al.*'s detector [Felzenszwalb et al., 2009] with multi-scale deformable part models. The original source code is available online and written in Matlab and C<sup>1</sup>.

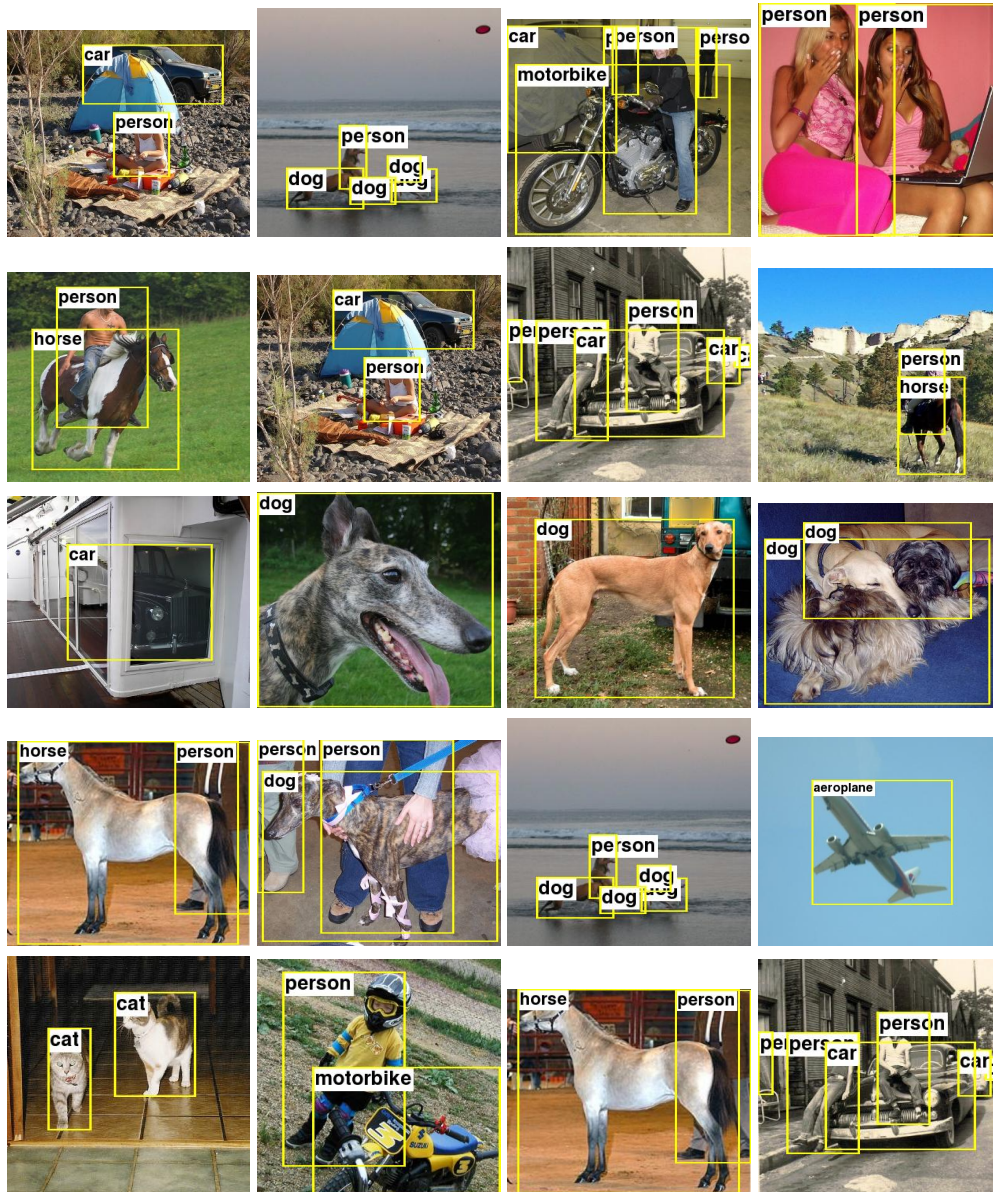
In this experiment, we used the data provided by the VOC2007 dataset to show that active learning can achieve the same classification performance by using only an actively selected subset of the data. To show the reduced data requirements for active learning, we trained our model in each category on four different positive datasets, including 25%, 50%, 75% and 100% of all positive training images, comparing the effect of adding training data *randomly* and *actively*. We used the same negative dataset in these experiments, as negative data is cheap to collect. For both the random and active data sets, we start by randomly selecting the first 25% of positive images and training. In the random training set, we then repeatedly add a further 25% of the positive images and train again, until all the available data have been added. We performed active sampling by first training our base model with a randomly selected 25% of the positive data. We used the base model to choose the most uncertain positive images from the remaining pool of positive data, repeatedly adding 25% of the data. Due to randomness in the training procedure, we repeated these experiments five times and present average results with error bars. Figure 3.2 shows the results of all twenty categories.

This simulation shows that active sampling is effective at selecting the most useful data for training. In most cases, a total of only half of all positive data is required to get the performance of the full training set.

---

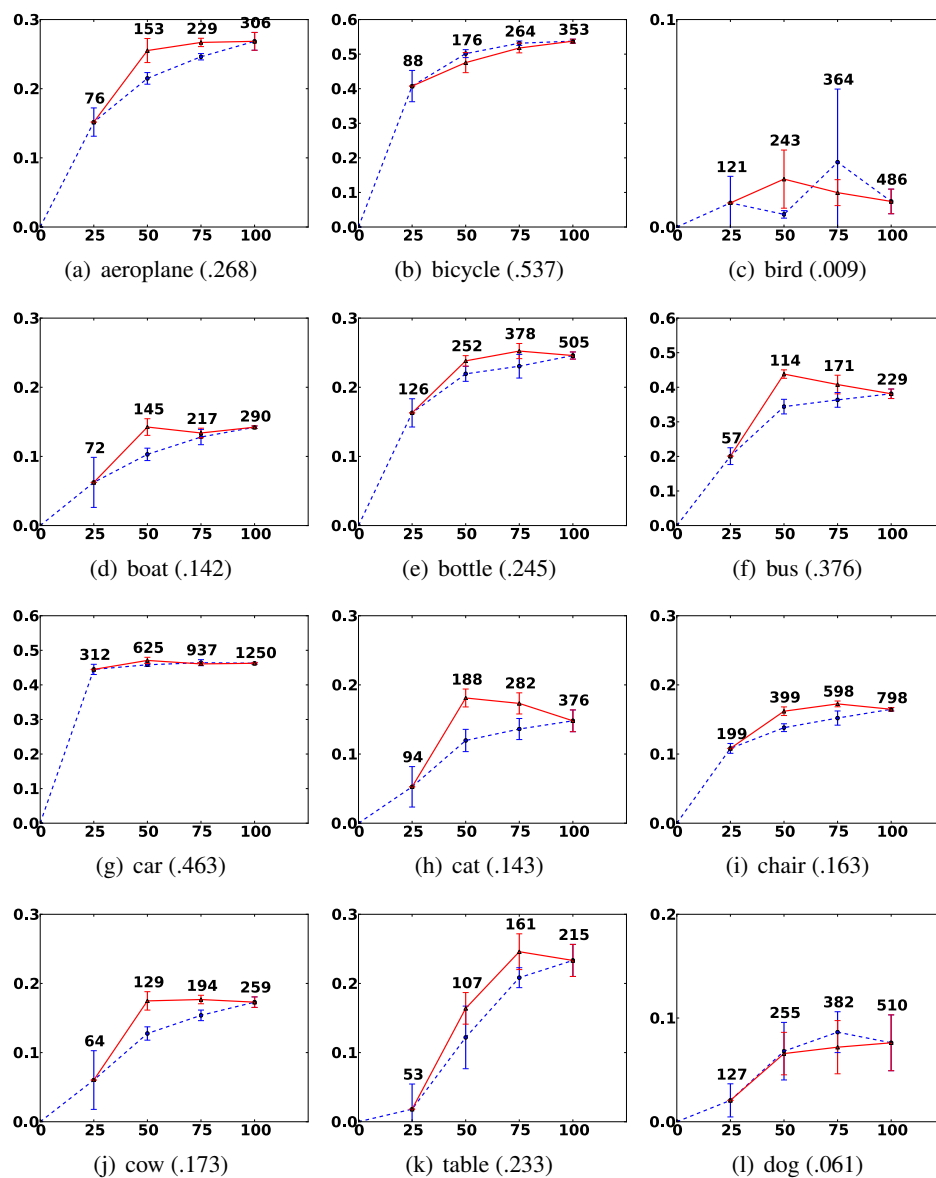
<sup>1</sup>The source code is publicly available at <http://people.cs.uchicago.edu/~pff/latent/>



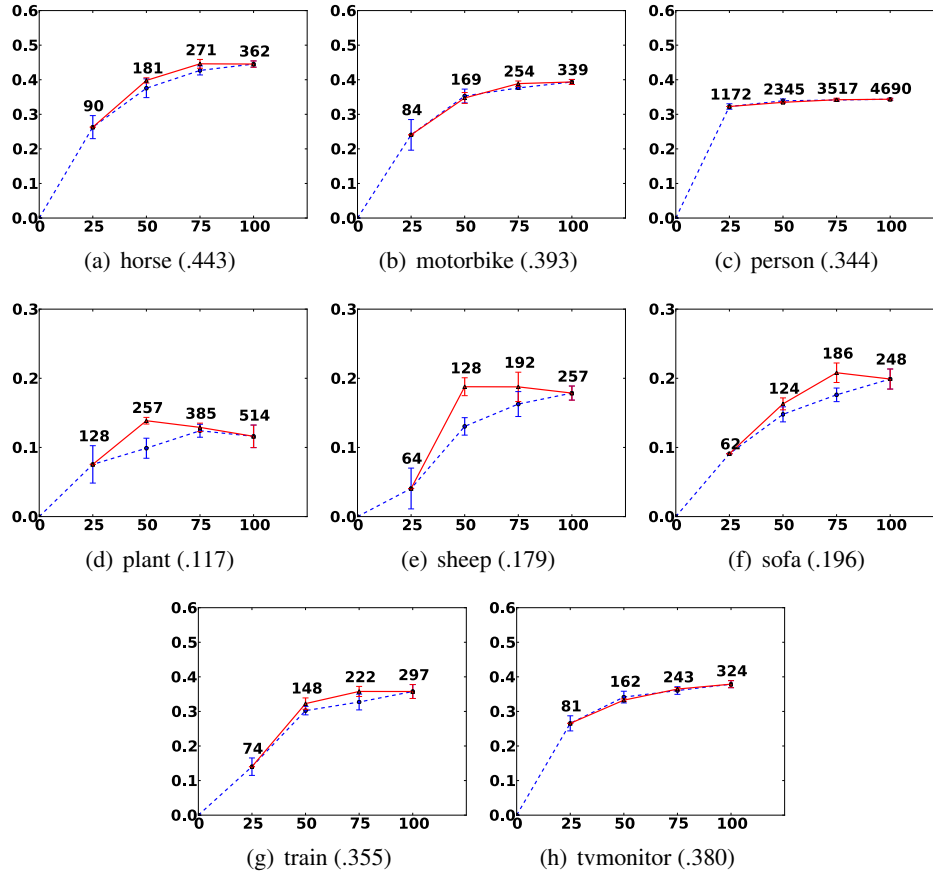


**Figure 3.1: PASCAL VOC training images with ground truth annotations.**

This shows some example images from the PASCAL VOC 2007 dataset. Yellow boxes are annotations with the category label on the top-left corner. These images represent complex scenes with severe occlusions and a wide range of the scale of objects.



**Figure 3.2:** These are the first 12 figures of a two-page figure that is continued on the next page.



**Figure 3.2: VOC2007: Random vs. active sampling of positive data.** There are 20 different visual categories. The horizontal axis of each figure represents the portion of positive training samples in percent (%) and the vertical axis gives the average precision. The solid red line shows the result of active sampling and the dotted blue line is for random sampling. We performed both random and active sampling of positive data five times with a different random seed. The errorbars are based on one standard deviation. For both random and active data sets, we start by training on a randomly selected 25% of positive samples. In the random training set, we then repeatedly add a further 25% of the positive samples and train again. In the active set, we added an additional 25% of actively selected samples iteratively until we use the full positive data. The number of positive samples in each subset is shown above each errorbar. The final average precision with the full positive data is shown next to the name of a category. Note that the result of active sampling often performs better with 50% or 75% of the images.

### 3.5 INRIA person dataset

To test active learning on another source of a realistic object class recognition problem, we also use data from the INRIA pedestrian dataset<sup>2</sup>. We use the INRIA Object Detection and Localization Toolkit<sup>3</sup> to implement the SVM with the person dataset and compare it with our actively trained SVM. The INRIA dataset and software were originally developed and used by Dalal and Triggs Dalal and Triggs [2005]. Figure 3.3 shows some training instances from the INRIA pedestrian dataset. We use their HOG descriptor, and compare generalization performance between a randomly trained SVM and an actively trained SVM.



**Figure 3.3: Some example data of pedestrians and non-pedestrians from the INRIA pedestrian dataset.**

In this experiment, we use the same HOG descriptor as Dalal and Triggs [2005], which is a vector of 3780 dimensions. For clarity, we call their method a semi-actively trained SVM, because their training consists of two phases. In the first phase, they train a naive person detector by using 2416 pedestrian patches and 12180 randomly sampled non-pedestrian patches. Then the person detector is re-trained by adding hard instances (i.e., false positives that are produced by the initial detector by scanning 1218 negative images). To provide comparisons, we train both a randomly trained SVM and an actively trained SVM. We use the same training dataset, which consists of 2416 pedestrian patches and many non-pedestrian patches from a set of 1218 images that do not contain any pedestrians, and the same test dataset, which consists of a separate set of 1132 pedestrian patches and over two million non-pedestrian patches that are sampled from a separate set of 453 non-pedestrian

<sup>2</sup>The data is publicly available at <http://pascal.inrialpes.fr/data/human/>

<sup>3</sup>The software is publicly available at <http://pascal.inrialpes.fr/soft/olt/>

images.

For our actively trained SVM, the learning task is an iterative training process: Based

---

**Algorithm 3.2 : Simulation of pool-based active learning algorithm for image window labelling**

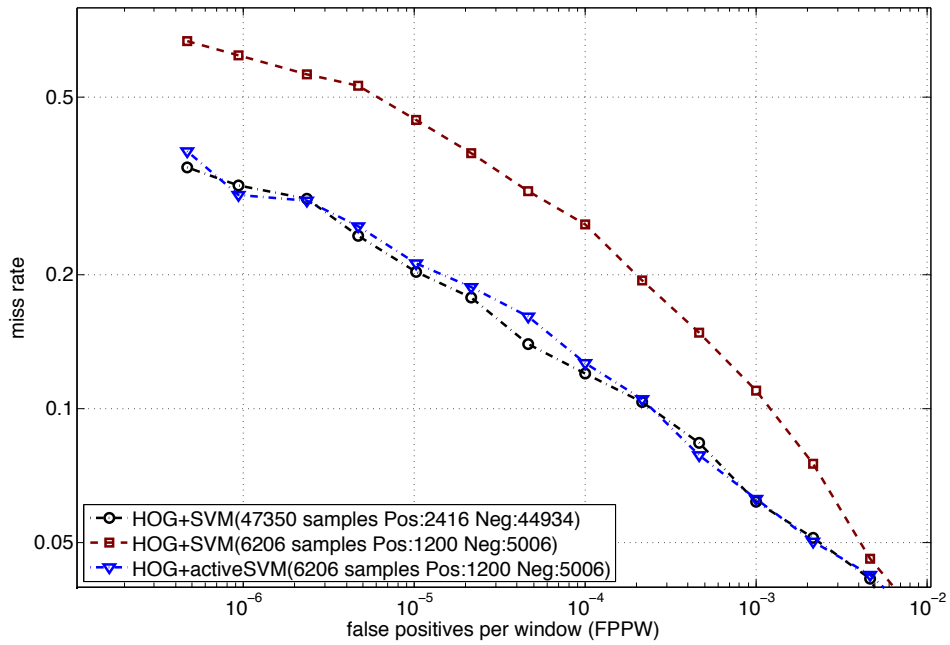
---

$\mathcal{L}$  is a set of labelled image windows and  $\mathcal{U}$  is a set of unlabelled windows.  $\theta$  is an SVM classifier

---

- 1: Train the initial SVM model  $\theta$  by randomly choosing  $m$  pedestrian patches and  $n$  non-pedestrian patches from  $\mathcal{L}$
  - 2: **while** if there are false positives from the negative data pool **do**
  - 3:   Scan all of the remaining unlabelled data  $\mathcal{U}$
  - 4:   Select the candidate data  $\mathcal{C}$  by adding misclassified instances:
    - $m$  false negatives from the positive data pool
    - $n$  false positives from the negative data pool
    - {This simulates the process in which the user corrects errors using our user interface.}
  - 5:   Stop selecting false negatives if we already have 1200 positive samples.
  - 6:   Add the candidate data  $\mathcal{C}$  to the training set.
  - 7:   Update the model  $\theta$  with the new training set.
  - 8: **end while**
- 

on our experiments, we set  $m = 50$  and  $n = 100$  and compare results with the best result reported by [Dalal and Triggs, 2005]. Following their approach, Figure 3.4 shows the result in a *Detection Error Tradeoff* curve on a log-log scale. Note that our actively trained SVM performs equally well with the best result by Dalal and Triggs. However, we use only 1200 actively selected pedestrian patches and only 5006 non-pedestrian patches, which is only 50% of the positive samples and 11% of the negative samples that Dalal and Triggs used. We restrict the amount of positive samples for training to be only 50% of the available positive data, mainly to show a large reduction of selecting positive samples. The randomly trained SVM uses 1200 pedestrian patches and 5006 non-pedestrian patches all selected randomly. It performs much worse than the other two. Our actively trained SVM selects 6206 samples and uses 4167 support vectors whereas the semi-actively trained SVM by Dalal and Triggs uses 4271 support vectors and our randomly trained SVM uses only 888 support vectors. This indicates that the significant reduction in labelling requirements can be achieved by active learning.



**Figure 3.4: DET curve on INRIA dataset.** These Detection Error Tradeoff curves show classification accuracy on the Dalal and Triggs pedestrian data. Our actively trained SVM uses only **6206** actively selected patches, as compared to the whole training set of 47350 patches for the original SVM. The actively trained SVM achieves a competitive performance with Dalal and Triggs while using much less data, while random selection of the same amount of data gives worse results.

## Chapter 4

# Experiments with Active Learning using an Internet Image Search Engine

In this section, we demonstrate that our prototype active learning GUI enables fast, efficient labelling to improve performance on data from the PASCAL VOC 2007. For comparison, we first prepare our baseline detectors by training a latent SVM with HOG [Felzenszwalb et al., 2009] on all twenty categories. The third column of Table 4.1 shows the average precision of each category. Due to randomness in the training procedure, those numbers do not exactly match with ones in [Felzenszwalb et al., 2009] even with the same parameter settings, but they differ by less than 1 percent.

### 4.1 An adaptive interface for active localization

In this section, we introduce the active localization interface for our interactive labelling system, ALOR (Active Learning for Object Recognition). Our interface iteratively presents the most informative query windows to the user and allows the user to correct mistakes made by the model. It is designed to work on any arbitrary user specified object category, using web images it automatically collects from major web search engines such as Google, Yahoo, and Flickr. Here, we demonstrate the interface on the “cow” category by using a set of images from the Google Image search engine.

We first collect a set of images from the Google Image search engine with query keywords such as “cow.” We then use our cow detector that is trained with VOC 2007 data as shown in Figure 3.2 and let the algorithm actively select each query. Figure 4.1 shows some screen shots of our system interface, which describes four major cases for labelling data. These cases are: (a) the window is indeed a cow, (b) the window is not a cow, (c) the window is ambiguous, perhaps due to partial overlap, so the window should be ignored for training, and finally (d) a user wishes to add a positive instance that was missed by the classifier or correct false positives. The rectangular boxes (in red) are queries selected by active learning with the minimum distance to the decision boundary. The green boxes are positive classifications in the current image that were not selected as queries. These green boxes are useful for giving the user feedback on the current classifier performance and to allow the addition of rare training instances that may lie far enough away from the decision boundary that they may otherwise never be detected or queried.

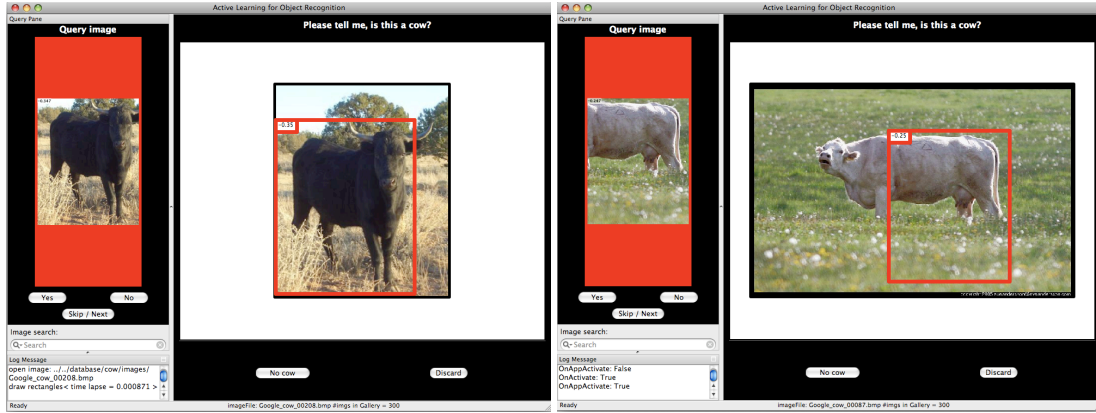
When a user wishes to add a training example that was not queried, they can simply click on the image. This will select the window with the highest classification confidence that contains the location of the mouse click. Therefore, the current classifier is still used to select the best-aligned training window, and user effort is minimized.

The interface is written in Python and C and utilizes multi-core CPUs to speed up the process and realize real-time interactions between a human user and a machine. Our interface currently takes less than a second (on an 8-core machine) to update and determine the next query windows upon receiving new labels, though the precise time depends on the number of dimensions of visual features and the resolution of the image. The interactive process has been sped up by precomputing image features, which we did in our experiments shown in Table 4.1.

## 4.2 Experiments

Next we perform a pool-based active learning session using Algorithm 3.1 with our active learning GUI to train these baseline detectors and improve their performance. With our interface, we collect training images from Flickr, which was a source for VOC 2007 images. We used the Flickr Image API to obtain only images from after the competition date of VOC 2007 to prevent the possibility of obtaining test data from the competition. Our interface allows a human user to search images based on a query word such as “cow,”





(a) Case 1: YES

(b) Case 2: NO



(c) Case 3: MAYBE

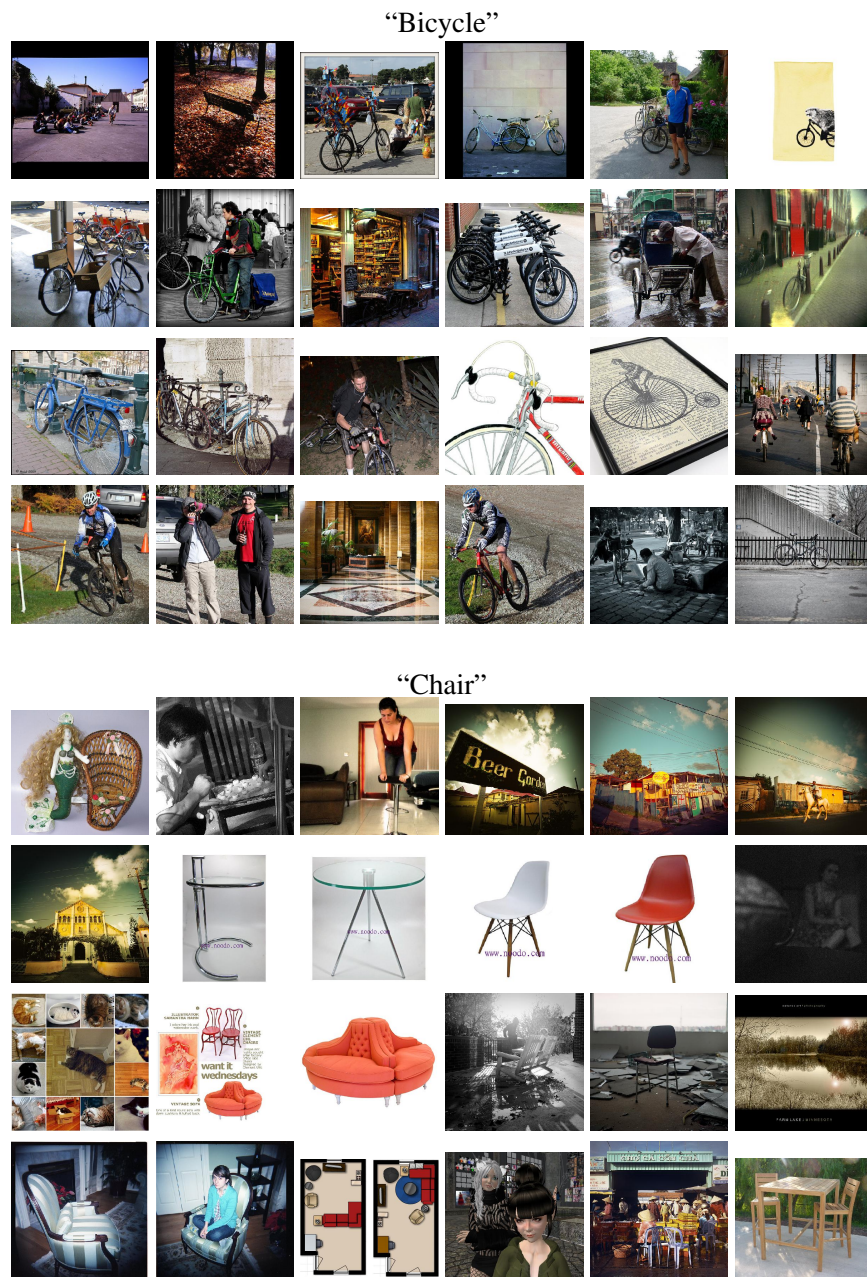
(d) Case 4: No target object

**Figure 4.1: Screen shots of our interactive labelling system with our actively trained LSVM.** This figure shows a set of screen shots of our interactive labelling system. In each case, the query window to be labelled is shown on the left, and the image from which it was selected is shown on the right (with the query window highlighted in red). (a) shows an instance when the query window is indeed a cow, so the appropriate answer is YES. (b) shows an example bounding window that is not a cow, so the answer is NO. Note that the bounding window partially covers a real cow, so it would not be obtained from the usual approach of labelling negative images. (c) shows a query window where it is hard to decide whether it is a cow, due to being largely occluded, misplaced, or incorporating multiple cows, so the answer is MAYBE, in which case we can just skip the query. (d) shows an instance in which an image does not contain any cow. In such case, a human user can specify that all windows in the image are negative instances and a machine selects only false positive windows.

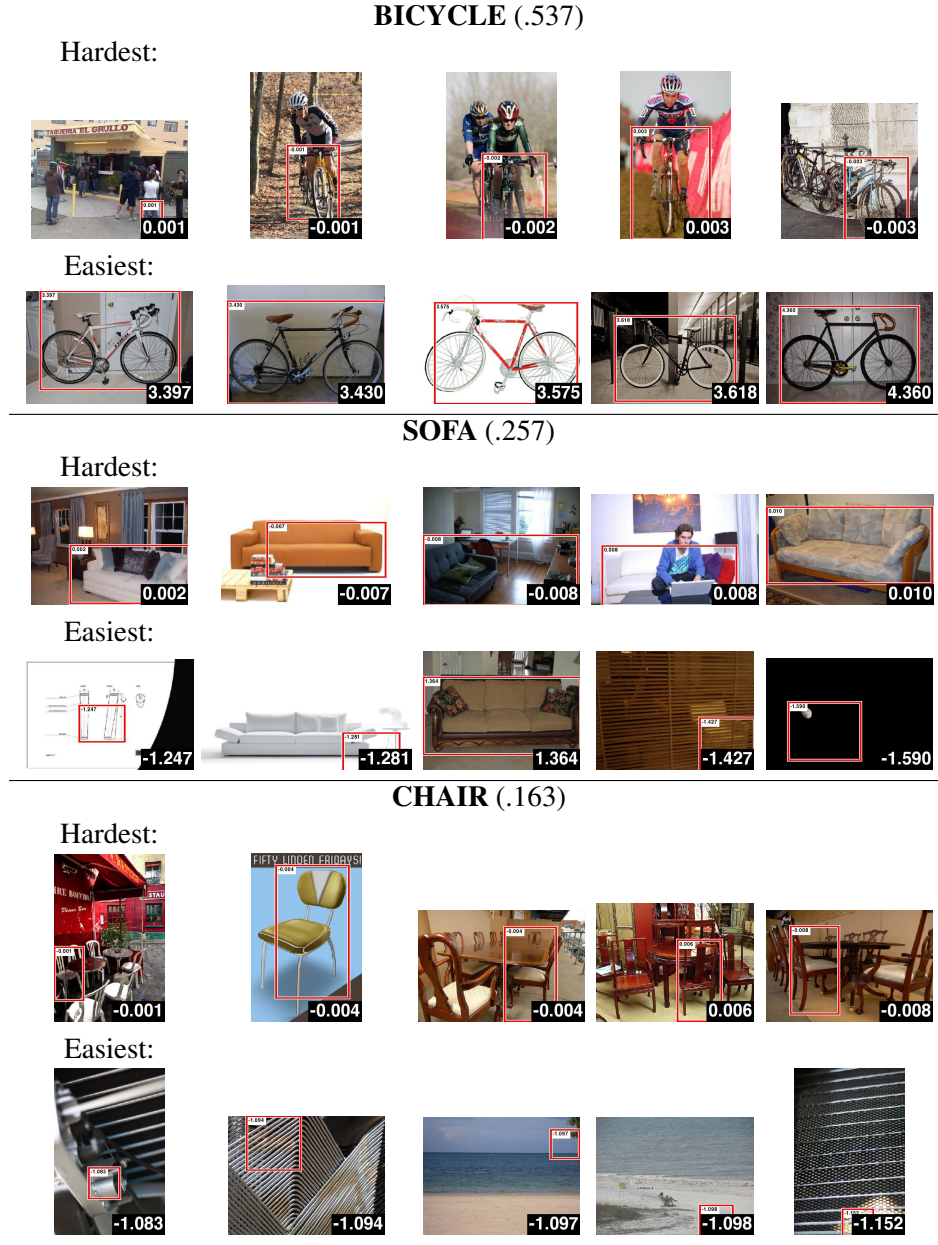
“motorbike,” etc. We collected the first 3,000 images from the Flickr database in each category. Figure 4.2 shows some example Flickr images by the keyword such as “bicycle” and “chair”. In order to further speed up the training process, we also cache visual features, using our baseline detectors to scan images, and sort them based on the uncertainty score in Equation 3.6 for the highest detection scoring window. It takes approximately 3-4 hours to search, download and scan 3,000 images (around 164 million bounding windows) to generate the set of most-uncertain query windows. However, this preprocessing can be done off-line with no human interaction. The effectiveness of active learning can be seen in Figure 4.3 which shows the top 5 *hardest* and *easiest* actively selected query windows in each category where ones that have a high uncertainty score are *hard* and ones that have a low uncertainty score are *easy*. The top 5 easiest query windows in the bicycle category contain target objects that are easily classified, whereas the top 5 easiest query windows in the chair category do not contain any target objects. In the sofa category, the top 5 easiest query windows include ones that contain easy target objects as well as ones without.

Table 4.1 summarizes the results and training data statistics. We downloaded in total 60,000 images from Flickr and annotated bounding windows in 300 actively selected images for each category based on uncertainty of the highest scoring detection window of each image. We used the highest scoring detection window for sorting images in order to get as many positive labels as possible from a large unlabelled data pool. We then conducted two simulations. One is a common active learning approach where we answer just 300 query windows based on our uncertainty sampling criteria (ALORquery). The other is a case of fully utilizing our interface where we are allowed to not only answer these 300 query windows but also add user selected query windows (ALORfull). It required only about 20 minutes per class for a person using our interface to improve the results from the first column and takes about 40 minutes to get the best performance improvement shown in the third column. Note that a user selects roughly 100 ~ 200 additional query windows in each category, which are mostly erroneous or missed detections that are never selected by the uncertainty criteria, but effectively presented by our interface. The best scores in bold already exceed the best competition results (of any method) over 17 of 20 categories from VOC2007 [Everingham et al., 2007]. The bottom row shows the mean AP score of all categories for each method.

Figure 4.4 shows the performance comparison of both ALORquery and ALORfull in a different number of training images. It shows that ALORfull consistently outperforms

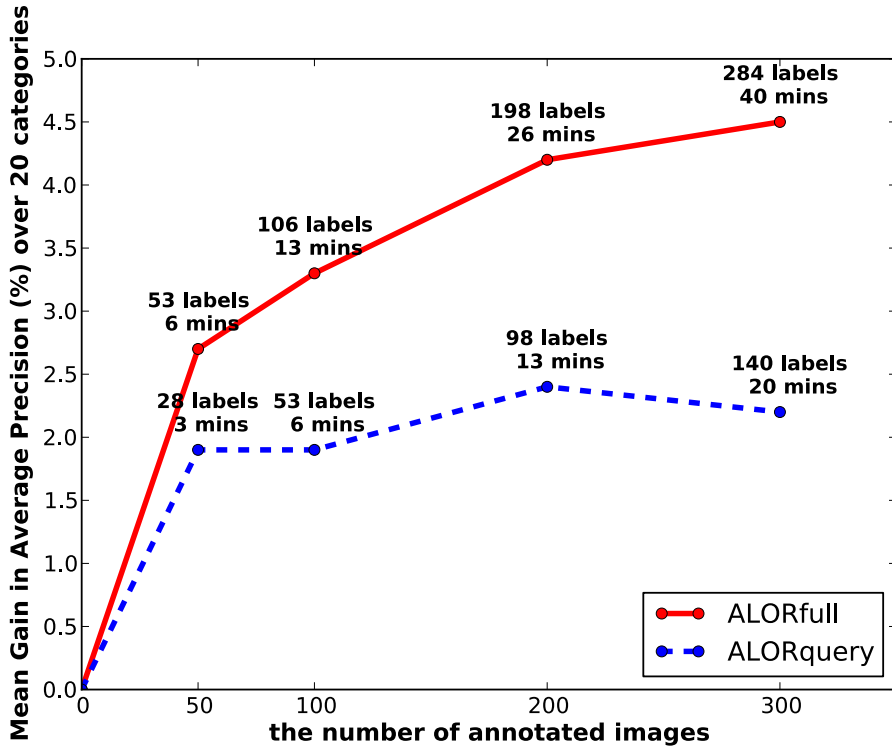


**Figure 4.2: Some images from Flickr.** This shows some example images returned from Flickr’s image search by using the keyword “bicycle” and “chair”. Notice a portion of unrelated images that are not suitable for training.



**Figure 4.3: Training images from Flickr, sorted for active learning.** This shows query windows for three categories. For each category, the top 5 hardest and easiest actively selected query windows are shown.

ALORquery, which indicates that user selected queries improve the overall detection performance. More results in Table 4.2 demonstrate an impact of user selected queries on the final classification performance. In the table, we show the result of ALORquery with 300 images and ALORfull with 100 and 300 images. For the cases of ALORfull, around 40 % of queries are selected by a user. Those queries are often quite challenging for a machine to choose, because they are often either erroneous or missed detections and distant from the decision boundary of a latent SVM. A human oracle is quite helpful in such cases.



**Figure 4.4: Performance comparison of ALORfull and ALORquery.** Each point represents the average number of additional positive labels and the average training time per category. By allowing a user to add additional queries, the performance is consistently better. Note that just 53 labels from ALORfull outperform 140 labels from ALORquery.

Figure 4.5 presents the gain of our best result for each category in the average precision. From our experiments, we can make several observations. First, in most of the categories, our active learning interface allows a user to quickly improve the performance



of the baseline detectors. Second, our user interface also allows a user to achieve a better performance than would be obtained by a simpler learning approach in which a user answers Yes/No/Maybe queries for selected windows. A lot of difficult machine-selected queries that a user is not sure about (4.1(c), for example) can be easily corrected with our interface. Third, with less than 40 minutes of user input, we can achieve significant performance improvement even over the best competition results. The PASCAL competition has a section in which users can provide their own data, but the difficulty of collecting such data means there have seldom been entries in that section. Our active learning approach and GUI would enable users to efficiently collect useful data for improved performance in such competitions or for real world applications.

In the boat, horse and person categories, our active learning approach did not provide much improvement for the relatively small additional amount of training data. We observe that both the boat and horse categories have a particularly heterogeneous dataset in both the size and shape of the object. Some of those instances are presented in Figure 4.6. The person category differs in that it already has so many object labels (4690, as opposed to the median 346) that it likely requires many more labels than the few hundred we provide to improve performance.

### 4.3 Conclusion

In chapters 3 and 4, we have presented an active learning system for object recognition and localization. This work differs from active learning work for image *classification* in that instead of learning a single classification for the image, our model can identify and localize objects, including multiple instances of the object of interest in a single image. Our experiments demonstrate that the active learning approach reduces the number of labels required to train an object classifier without reducing performance over state-of-the-art classification schemes. It also greatly reduces the human effort required to select image regions containing the object of interest by automatically finding the most useful windows in an image. Our system is fast enough to be used interactively, and we demonstrate a prototype GUI for active learning of object locations, which uses image windows to guide human labelling.

While our experiments show that our actively trained latent-SVM with HOG descriptors works well with active learning, the system does not depend on a specific classifier or feature set. If other classifiers, such as AdaBoost, or features are found to be more suitable

|         | LSVM+HOG      |      |  | ALORquery         |     |      |  | ALORfull          |     |      |
|---------|---------------|------|--|-------------------|-----|------|--|-------------------|-----|------|
|         | object labels | APs  |  | additional labels |     | APs  |  | additional labels |     | APs  |
|         |               |      |  | pos               | neg |      |  | pos               | neg |      |
| aerop.  | 306           | .268 |  | 209               | 27  | .283 |  | 293               | 12  | .282 |
| bicyc.  | 353           | .537 |  | 227               | 12  | .548 |  | 359               | 26  | .555 |
| bird    | 486           | .009 |  | 72                | 151 | .048 |  | 172               | 134 | .095 |
| boat    | 290           | .142 |  | 87                | 89  | .144 |  | 162               | 69  | .143 |
| bottle  | 505           | .245 |  | 133               | 100 | .259 |  | 300               | 93  | .294 |
| bus     | 229           | .376 |  | 108               | 26  | .435 |  | 281               | 29  | .504 |
| car     | 1250          | .463 |  | 97                | 104 | .468 |  | 186               | 138 | .489 |
| cat     | 376           | .143 |  | 108               | 3   | .209 |  | 282               | 0   | .224 |
| chair   | 798           | .163 |  | 139               | 66  | .161 |  | 373               | 93  | .180 |
| cow     | 259           | .173 |  | 133               | 63  | .184 |  | 430               | 92  | .254 |
| table   | 215           | .233 |  | 164               | 46  | .295 |  | 233               | 67  | .314 |
| dog     | 510           | .061 |  | 130               | 38  | .125 |  | 235               | 33  | .131 |
| horse   | 362           | .443 |  | 167               | 31  | .398 |  | 321               | 69  | .370 |
| mbike   | 339           | .393 |  | 229               | 8   | .414 |  | 348               | 50  | .431 |
| person  | 4690          | .344 |  | 47                | 183 | .342 |  | 122               | 218 | .344 |
| plant   | 514           | .117 |  | 136               | 26  | .153 |  | 370               | 15  | .169 |
| sheep   | 257           | .179 |  | 178               | 25  | .257 |  | 512               | 68  | .282 |
| sofa    | 248           | .196 |  | 161               | 49  | .230 |  | 212               | 56  | .257 |
| train   | 297           | .355 |  | 85                | 73  | .354 |  | 184               | 111 | .400 |
| tvmon   | 324           | .380 |  | 195               | 45  | .371 |  | 321               | 69  | .392 |
| Mean AP |               | .261 |  |                   |     | .283 |  |                   |     | .306 |

**Table 4.1: Average precision on VOC2007 and training data statistics.** This figure shows the result of our experiments and training data statistics. The first section shows the number of object labels in VOC2007 data and the result obtained by our implementation of the baseline latent-SVM with HOG [Felzenszwalb et al., 2009] on VOC2007 test data (**LSVM+HOG**). The middle section shows the total number of additional positive and negative labels and the AP score with these additional labels. We obtained the additional data by using models from **LSVM+HOG** and our active learning GUI on only 300 actively selected images (one query per image, 300 queries in total) out of 3000 web images for each category. This represents a common active learning approach with uncertainty sampling criteria where a user is not allowed to add any additional queries and is only queried by a machine (**ALORquery**). The last section shows additional labels and AP scores on the same 300 actively selected images. But a user is allowed to fully utilize our interface by not only answering queries from a machine but also adding his/her own queries with our interface (**ALORfull**). The best AP scores are in bold face.

| Method                  | Avg.<br>Training<br>Time<br>(mins) | user selected<br>queries |                  | machine selected<br>queries |                  | Avg.<br>additional labels |       | Mean<br>AP<br>(%) |
|-------------------------|------------------------------------|--------------------------|------------------|-----------------------------|------------------|---------------------------|-------|-------------------|
|                         |                                    | proportion<br>(%)        | Avg.<br>distance | proportion<br>(%)           | Avg.<br>distance | pos                       | neg   |                   |
| ALORquery<br>300 images | 20                                 | 0                        | NA               | 100                         | 0.283            | 140.1                     | 58.25 | 28.3              |
| ALORfull<br>100 images  | 13                                 | 42                       | 0.643            | 58                          | 0.200            | 106.85                    | 20.15 | 29.4              |
| ALORfull<br>300 images  | 40                                 | 43                       | 0.690            | 57                          | 0.296            | 284.8                     | 72.1  | 30.6              |

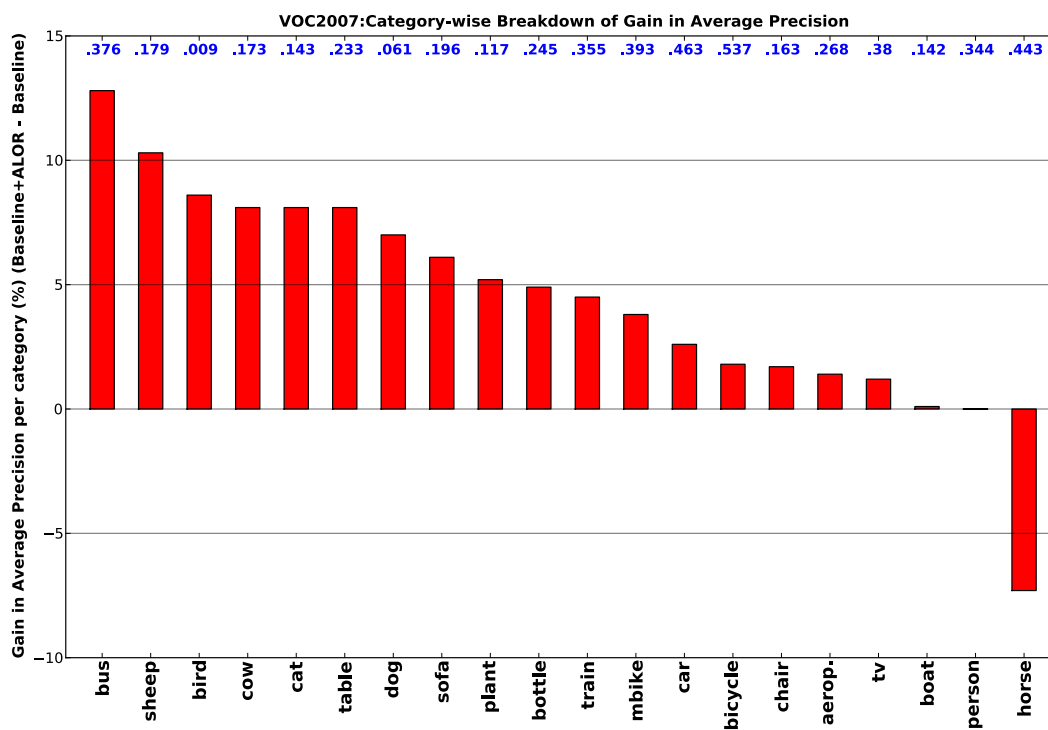
**Table 4.2: Query statistics for ALORquery and ALORfull.** The average distance represents the average distance per query to the decision boundary of a latent SVM. The average training time, additional labels, and mean APs are measured per category. The proportion of queries is the portion of the total number of queries in all 20 categories. Note how user-selected queries influence on the final performance in average precision.

to a domain, we can incorporate them into our framework. We also believe that other aspects of object classification can benefit from active learning, as the expense of labelling is a ubiquitous problem in machine learning. Fast, efficient labelling can mean cheaper experiments, faster development time, and higher-performance flexible object detectors.

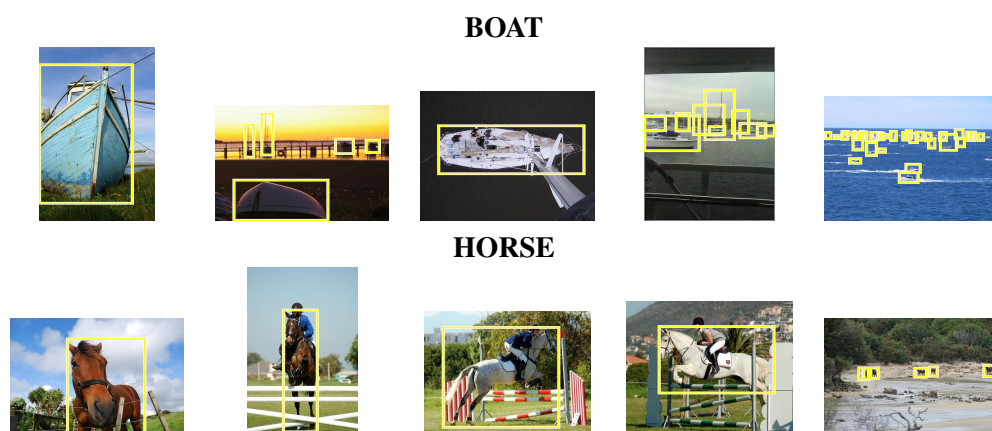
Last but not least, our adaptive interface with active learning can easily scale up by crowdsourcing and simulating live active learning with a human in a loop, which is similar to the work by [Vijayanarasimhan and Grauman, 2011]. In our experiments, one active learning session takes on average a few hours of computer time to train a mixture of deformable part models with a few hundred labelled bounding windows and around 2,000 negative images<sup>1</sup> and an extra few hours to select candidate queries from a relatively small pool of 3,000 unlabelled images. Therefore, there is an interval of at least around 5 hours before the next iteration per object category. Furthermore, the waiting time also increases roughly linearly with respect to the number of additional labels. Such an expensive computational time for both training an LSVM and scanning a pool of novel data limits our approach only to simulating a live active training based on a batch process rather than performing the real live training where the model is updated on-line as labels are added.

<sup>1</sup>The training time for a particular version of an LSVM [Felzenszwalb et al., 2009] totally depends on the size of the training data. Its complexity is  $O(NM)$  where  $N$  is the number of samples and  $M$  is the number of feature dimensions.





**Figure 4.5: Category-wise breakdown of gain in average precisions: VOC baseline vs VOC baseline + ALOR.** This figure is best viewed in colour. This shows the improvement for our active learning framework for each category, where a red bar represents an improvement in the average precision and a number in the blue font is the baseline average precision.



**Figure 4.6: Heterogeneous shapes and sizes.** This shows screen shots of VOC test images. The shots in the top row are for the boat category and those in the bottom row are for the horse category. Yellow boxes are ground truth.

## Chapter 5

# Multi-target Tracking in Sports Video

In chapters 3 and 4, we show that our active learning approach, which uses a novel interface to combine machine intelligence with human interventions, effectively improves a state-of-the-art classifier [Felzenszwalb et al., 2009]. The following Chapter 6 will extend the idea to a novel self-learning system that is capable of improving localization of objects of interest in sparsely labelled videos. Unlike static images, videos are much more structured because each image frame in a video sequence has a strong spatio-temporal relation to subsequent image frames. For exploiting such structured data, one of the major extensions we have is a multi-target tracking system which connects a spatio-temporal link of labels over multiple image frames.

In Section 5.1, we first introduce our previous work on a multi-target tracking system called the Boosted Particle Filter (BPF) [Lu et al., 2009; Okuma et al., 2004]. Section 5.2 compares the performance of BPF with a more recent collaborative work on another multi-target tracker [Lu et al., 2011], which also takes a tracking-by-detection approach by taking detection results of the Deformable Part Model [Felzenszwalb et al., 2009] and using a Kalman filter to produce tracklets (i.e., sequences of bounding windows). Instead of comparing theoretical differences between these two filtering approaches, we focus primarily on a practical aspect of tracking algorithms by addressing implementation differences, design issues and their impact on the performance.

## 5.1 Boosted particle filter

### 5.1.1 Statistical model

In non-Gaussian state-space models, the state sequence  $\{\mathbf{x}_t; t \in \mathbb{N}\}$ ,  $\mathbf{x}_t \in \mathbb{R}^{n_x}$ , is assumed to be an unobserved (hidden) Markov process with initial distribution  $p(\mathbf{x}_0)$  and transition distribution  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ , where  $n_x$  is the dimension of the state vector. In our case,  $\mathbf{x} = \{l_x, l_y, l_s\}$  where  $\{l_x, l_y\}$  represents the location of the player, and  $l_s$  represents the size of the player in the image coordinate system. The observations  $\{\mathbf{y}_t; t \in \mathbb{N}\}$ ,  $\mathbf{y}_t \in \mathbb{R}^{n_y}$ , are conditionally independent given the process  $\{\mathbf{x}_t; t \in \mathbb{N}\}$  with marginal distribution  $p(\mathbf{y}_t|\mathbf{x}_t)$ , where  $n_y$  is the dimension of the observation vector.

Letting  $\mathbf{y}_{1:t} \triangleq \{\mathbf{y}_1 \dots \mathbf{y}_t\}$  be the observation vectors up to time  $t$ , our goal is to estimate  $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ , the probability of the current state  $\mathbf{x}_t$  given  $\mathbf{y}_{1:t}$ , which can be solved by the following Bayesian recursion [Doucet et al., 2001]:

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{y}_{1:t}) &= \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})} \\ &= \frac{p(\mathbf{y}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_{t-1}}{\int p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})d\mathbf{x}_t} \end{aligned} \quad (5.1)$$

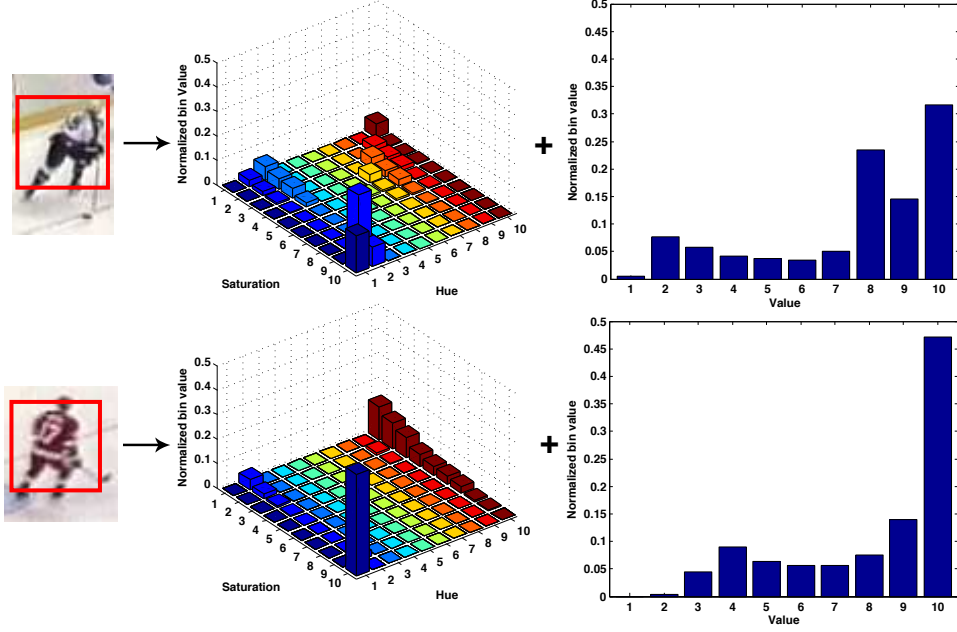
In our tracking system, this transition distribution  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$  is a combination of a first-order dynamic model and a second-order autoregressive dynamic model (i.e., a constant acceleration) with additive Gaussian noise. The observation likelihood  $p(\mathbf{y}_t|\mathbf{x}_t)$  is defined in the following section.

### 5.1.2 Observation likelihood

Our observation model consists of colour and shape information which is encoded by the HSV colour histogram and the HOG descriptor, respectively. We compute the observation likelihood by

$$p(\mathbf{y}_t|\mathbf{x}_t) \propto p_{hsv}(\mathbf{y}_t|\mathbf{x}_t) p_{hog}(\mathbf{y}_t|\mathbf{x}_t) \quad (5.2)$$

For the HSV colour model, we use a combination of a 2D colour histogram based on Hue and Saturation and a 1D colour histogram based on Value as it is shown in Figure 5.1.



Colour histogram of a player (TOP: white uniform BOTTOM: red uniform)

**Figure 5.1: HSV Colour histograms.** This figure shows two different colour histograms of selected rectangular regions. The first 2D histograms are Hue and Saturation histograms. The other 1D histograms are Value histograms. Both 2D and 1D histograms have Z axis and Y axis respectively for the normalized bin value (both histograms are normalized such that all bins sum to one). The player on top has a uniform whose colour is the combination of dark blue and white and the player on bottom has a red uniform. Although one can clearly see concentrations of colour bins due to the limited number of colours, this figure shows a clear colour distinction between two players.

The distribution of the colour likelihood is given as follows:

$$p_{hsv}(\mathbf{y}_t | \mathbf{x}_t) \propto e^{-\lambda_c \xi[\mathbf{K}^*, \mathbf{K}(\mathbf{x}_t)]} \quad (5.3)$$

where  $\mathbf{K}(\mathbf{x}_t)$  is the HSV colour histogram computed at  $\mathbf{x}_t$ ,  $\mathbf{K}^*$  is the template for the HSV colour histogram, and  $\xi(\cdot, \cdot)$  is the diffusion distance [Ling and Okada, 2006]. We fix the scaling constant  $\lambda_c = 10$  throughout our experiments.

We use a 3D histogram based on the magnitude of gradients in both the  $x$  and  $y$  direction

and their orientations for the HOG descriptor. Then the following likelihood distribution is given:

$$p_{hog}(\mathbf{y}_t|\mathbf{x}_t) \propto e^{-\lambda_s \xi[\mathbf{H}^*, \mathbf{H}(\mathbf{x}_t)]} \quad (5.4)$$

where  $\mathbf{H}(\mathbf{x}_t)$  is the HOG descriptor computed at  $\mathbf{x}_t$ ,  $\mathbf{H}^*$  is the template for the HOG descriptor, and  $\xi(\cdot, \cdot)$  is the diffusion distance [Ling and Okada, 2006]. We fix the scaling constant  $\lambda_c = 10$  throughout our experiments.

### 5.1.3 Particle filtering

Since the observation likelihood (Equation 5.3) is nonlinear and non-Gaussian, there is no analytical solution for the Bayesian recursion Equation 5.1. Instead, we seek an approximation solution, using particle filtering [Doucet et al., 2001].

In standard particle filtering, we approximate the posterior  $p(\mathbf{x}_t|\mathbf{y}_{1:t})$  with a Dirac measure using a finite set of  $N$  particles  $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}_{i=1}^N$ . To accomplish this, we sample candidate particles from an appropriate proposal distribution

$$\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t^{(i)}|\mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_{1:t}) \quad \text{for } i = 1 \dots N \quad (5.5)$$

In the simplest scenario, it is set as  $q(\mathbf{x}_t^{(i)}|\mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_{1:t}) = p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})$ , yielding the bootstrap filter [Doucet et al., 2001]. However, a smarter proposal distribution can be employed. The following section will discuss this issue.

The weights associated with these particles according to the following importance ratio:

$$w_t^{(i)} = w_{t-1}^{(i)} \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)}) p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_{1:t})} \quad (5.6)$$

We resample the particles using their importance weights to generate an unweighted approximation of  $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ . The particles are used to obtain the following approximation of the posterior distribution:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \sum_{i=1}^N w_t^{(i)} \delta_{\mathbf{x}_t^{(i)}}(\mathbf{x}_t) \quad (5.7)$$

#### 5.1.4 Boosted particle filter

It is widely accepted that proposal distributions that incorporate the recent observations (in our case, through the Adaboost detections) outperform naïve transition prior proposals considerably [Merwe et al., 2000; Rui and Chen, 2001]. Here, we introduce the Boosted Particle Filter [Lu et al., 2009; Okuma et al., 2004] that incorporates the current detections of hockey players to produce a better proposal distribution  $q(\mathbf{x}_t^{(i)} | \mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_{1:t})$ .

##### Adaboost Detection

In order to detect hockey players in the current frame, we adopt the cascaded Adaboost algorithm of Viola and Jones [Viola and Jones, 2004], which was originally developed for detecting faces. In our experiments, a 23 layer cascaded classifier is trained to detect hockey players. In order to train the detector, a total of 5609 figures of hockey players is used. These figures are scaled to have a resolution of  $24 \times 24$  pixels. We hand annotate figures of hockey players to use for the training as shown in Figure 5.2. Unlike the detector used in [Okuma et al., 2004], our trained Adaboost classifier produces few false positives (i.e., a few false positives in several thousand frames) even alongside the edge of the rink where most false positives appeared in [Okuma et al., 2004]. More human intervention with a larger and better training set leads to better Adaboost detection results, although localization failures would still be expected in regions of clutter and overlap. The non-hockey-player sub-windows used to train the detector are generated from over 300 images manually chosen to contain nothing but the hockey rink and audience. Since our tracker is implemented for tracking hockey scenes, there is no need to include training images from outside the hockey domain. Suffice it to say, exploiting such domain knowledge greatly reduces the false positive rate of our detector.

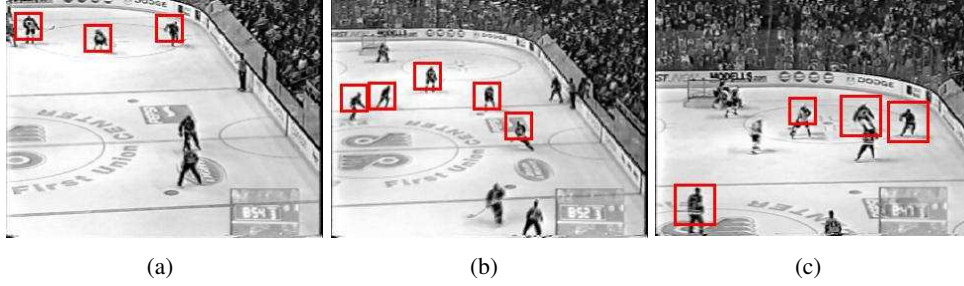
The results of using the cascaded Adaboost detector in our hockey dataset are shown in Figure Figure 5.3. The cascaded Adaboost detector performs well at detecting the players but often gets confused in a cluttered region with multiple players and ignores some of players.

##### Proposal Distribution with the Adaboost Detections

It is clear from the Adaboost detection results that they could be improved if we considered the motion models of the players. In particular, by considering plausible motions, the num-



**Figure 5.2: Training data for the Adaboost detector.** Training data for the Adaboost detector are hockey player patches in various poses



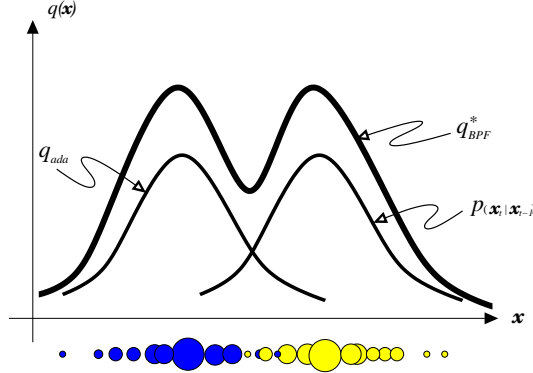
**Figure 5.3: Hockey player detection results.** This figure shows results of the Adaboost hockey detector. (a), (b), and (c) show mostly accurate detections. Please note that there are players who are not detected and some of the boxes do not cover the entire figure of the player (i.e., a box is too small to cover a lower part of the body)

ber of false positives could be reduced. For this reason, the Boosted Particle Filter (BPF) incorporates the Adaboost detection in the proposal mechanism of the particle filters. The expression for the proposal distribution is given by the following mixture.

$$q_{BPF}^*(\mathbf{x}_t^{(i)} | \mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_{1:t}) = \alpha_{ada} q_{ada}(\mathbf{x}_t^{(i)} | \mathbf{y}_t) + (1 - \alpha_{ada}) p(\mathbf{x}_t^{(i)} | \mathbf{x}_{t-1}^{(i)}) \quad (5.8)$$

where  $q_{ada}$  is a Gaussian distribution centered in the Adaboost detection with a fixed variance (See Figure Figure 5.4). The parameter  $\alpha_{ada}$  can be set dynamically without affecting the convergence of the particle filter (it is only a parameter of the proposal distribution and therefore its influence is corrected in the calculation of the importance weights). When  $\alpha_{ada} = 0$ , our algorithm reduces to the bootstrap particle filter. By increasing  $\alpha_{ada}$  we place more importance on the Adaboost detections. We can adapt the value of  $\alpha_{ada}$  depending on





**Figure 5.4: Mixture of Gaussians for the Proposal Distribution.**

tracking situations, including cross overs, collisions and occlusions. We set  $\alpha_{ada} = 1$  for implementing a boosted particle filter throughout our experiments.

Since there is no guarantee that the Adaboost detector detects all targets in the scene, the detection results can be sparse over time. The performance of BPF is, however, much better when there are many detections densely over time. One way to further improve the performance of BPF is to use an additional proposal mechanism other than the Adaboost detector. Thus, we use a mode-seeking algorithm similar to mean shift [Comaniciu and Meer, 2002] to find a local maximum of the HSV and HOG observation likelihoods and employ a Gaussian distribution centered in the local maximum as a new proposal distribution. This proposal is not as reliable as the Adaboost detections; however, it is often better than the transition distribution which cannot accurately model the dynamics of the targets due to the moving camera. Therefore, we use a mode-seeking proposal as an alternative whenever mixture proposals with the detection results are not available (i.e., no detections nearby a target).

### 5.1.5 Multi-target tracking

Boosted Particle Filters perform multi-target tracking by running multiple independent BPFs for every target in the scene. Briefly, the targets are detected and initialized by using the cascaded Adaboost detector described in Section 5.1.4. Then, BPF is applied to estimate the posterior distribution over  $x_{t+1}$ . To update the posterior distribution over  $\{s_{t+1}, z_{t+1}\}$ , we compute the mean  $\bar{x}_{t+1}$  of the posterior distribution  $p(x_{t+1} | y_{t+1})$ , and extract the image

patch  $\bar{\mathbf{y}}_{t+1}$  located in  $\bar{\mathbf{x}}_{t+1}$ .

There are also mechanisms to remove and merge the targets. The targets will be removed either when their Adaboost confidence is lower than a threshold, or when the bounding boxes are out of the image. The merge operation is performed when there is significant overlap between two bounding boxes. The mean of the two bounding boxes will be computed and the target with the lower Adaboost confidence will be removed.

## 5.2 Comparisons between BPF-Adaboost and KF-DPM

This section presents differences between the Boosted Particle Filter and a more recent Kalman filter tracker from [Lu et al., 2011] in terms of their implementations and tracking performance in our hockey video. Section 5.2.1 explains implementation differences between these two trackers. Section 5.2.2 presents tracking results of both trackers and compares their performance. Throughout this section, we abbreviate the Boosted Particle Filter as BPF-Adaboost and the Lu et al. [2011] tracker as KF-DPM since it uses the Deformable Part Model by [Felzenszwalb et al., 2009] for detecting hockey players instead of the Adaboost detector in the case of BPF.

### 5.2.1 Implementation differences

There are several implementation differences between BPF-Adaboost and KF-DPM. Here, we highlight the differences that particularly influence their tracking performance on our hockey video.

#### Player detection

BPF and KF-DPM are both tracking-by-detection approaches. However, these approaches use different models for detecting hockey players. BPF uses the Adaboost detector by Viola and Jones [2004] and KF-DPM uses the deformable part model (DPM) by Felzenszwalb et al. [2009]. Table 5.1 shows detection results by both detectors. The table clearly shows that the DPM detector has a better performance than the Adaboost detector in terms of precision and recall. This is mainly because the DPM detector has a rich representation of a deformable part model rather than a single template of Histogram of Oriented Gradients. The Adaboost detection result is used in the mixture proposal distribution for BPF as in Section 5.1.4. The DPM detection bounding windows are used as observations in filtering

process of the Kalman filter [Kalman, 1960].

| method   | Prec(%) | Rec(%) |
|--|---------|--------|
| Adaboost<br>[Viola and Jones, 2004]                        | 82.6    | 64.1   |
| Deformable Part Model (DPM)<br>[Felzenszwalb et al., 2009] | 84.7    | 77.7   |

**Table 5.1: Detection comparison of Adaboost and DPM.** This table compares detection results of the Adaboost detector and the DPM detector in static images from our hockey video. In both precision and recall, the DPM detector has a better performance. There is a noticeable difference in recall between these detectors.

### Filtering

Both BPF-Adaboost and KF-DPM use a probabilistic filtering method for tracking hockey players. As in Section 5.1.3, BPF-Adaboost uses an approximate solution, using particle filtering [Doucet et al., 2001] for the Bayesian Recursion in Equation 5.1, whereas KF-DPM uses the Kalman filter [Kalman, 1960]. The major differences, aside from theoretical differences between the particle filter and Kalman filter, are the observations  $\{\mathbf{y}_t; t \in \mathbb{N}\}$  and the transition distribution  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ .

BPF-Adaboost uses colour and shape information for the observations which are encoded by the HSV colour histogram and the HOG descriptor (Section 5.1.2) and incorporates Adaboost detections in the mixture proposal distribution (Section 5.1.4). KF-DPM uses only detection results from the DPM detector as the observations. Although KF-DPM uses less appearance information of the target objects in filtering, it benefits from being less influenced by misleading appearance information, especially when hockey players with a similar appearance are occluded by each other.

The dynamical system of the state sequences is different in BPF-Adaboost and KF-DPM. BPF-Adaboost uses a combination of a first-order dynamic model and a second-order autoregressive dynamic model (Section 5.1.1). KF-DPM uses a linear-Gaussian distribution  $p(\mathbf{x}_t|\mathbf{x}_{t-1}, t) = \mathcal{N}(\mathbf{x}_t|t\mathbf{a}_t + b_t, \Sigma_{\mathbf{x}})$ , where linear regression parameters  $(\mathbf{a}_t, b_t)$  are learned on-line from the history of posterior mean estimates. Therefore, BPF-Adaboost considers up to the second-order dynamics while KF-DPM considers the history (up to  $t - 10$  in our experiments) of posterior mean estimates.

### 5.2.2 Performance comparison

We tested BPF-Adaboost and KF-DPM in our hockey video that contains 1,000 frames. Table 5.2 defines metrics for evaluating tracking results. Table 5.3 shows results of performance comparison between these two trackers. In this table, KF-DPM clearly outperforms BPF. We suspect that implementation differences in Section 5.2.1 created this performance gap between these two trackers.

Firstly, KF-DPM uses better detection results by the Felzenszwalb et al. [2009] detector, which has a slightly higher precision and a much higher recall. Therefore, KF-DPM has an advantage of using much denser detection results and thus ends up with much higher precision and recall (by almost 30%) than BPF. Secondly, KF-DPM focuses more on the motion of targets rather than their appearance information. Its transition distribution considers a longer history of target trajectories and, as a result, reduces the number of identity switches (IDS). BPF, on the other hand, incorporates both the appearance information and the motion of targets. In our hockey data, the appearance information of hockey players seems to be misleading, especially in occlusions, because hockey players do have a similar shape and even a similar colour if they are in the same team.

Based on these comparisons, we chose KF-DPM as the tracker of our self-learning framework in the next chapter. In addition, KF-DPM is a simpler, light-weight probabilistic filtering approach which does not require a stochastic sampling process.

| Name         | Definition   |
|--------------|--|
| Precision(%) | No. of correctly matched objects / total No. of output objects.  |
| Recall(%)    | No. of correctly matched objects / total No. of ground-truth objects.  |
| FA           | No. of false alarms per frame. The smaller, the better.  |
| MT(%)        | Percentage of ground-truth trajectories which are correctly covered by tracker output for more than 80%                          |
| ML(%)        | Percentage of ground-truth trajectories which are correctly covered by tracker output for less than 20% The smaller, the better. |
| IDS          | Total No. of times that a tracked trajectory changes its matched ground-truth identity. The smaller, the better.                 |

**Table 5.2: Various metrics for tracking evaluation.**

| method                      | Prec(%)                | Rec(%)                 | FA                     | MT(%)                  | ML(%)                 | IDS                   |
|-----------------------------|------------------------|------------------------|------------------------|------------------------|-----------------------|-----------------------|
| BPF-Adaboost                | 65.5<br>( $\pm 21.4$ ) | 50.8<br>( $\pm 16.6$ ) | 2.30<br>( $\pm 1.42$ ) | 16.0<br>( $\pm 12.0$ ) | 25.7<br>( $\pm 9.6$ ) | 37.0<br>( $\pm 5.1$ ) |
| KF-DPM<br>[Lu et al., 2011] | 91.8                   | 79.7                   | 0.61                   | 56.3                   | 20.8                  | 27                    |

**Table 5.3: Performance comparison of BPF-Adaboost and KF-DPM in our hockey video.** This shows the tracking result of BPF-Adaboost and KF-DPM [Lu et al., 2011] over a hockey sequence of 1,000 frames. Although KF-DPM uses a Kalman filter with linear-Gaussian dynamics, its tracking results are better than BPF-Adaboost in all aspects. Note that we had 10 runs of BPF-Adaboost since it is a probabilistic method that uses stochastic sampling process. Means with one standard deviation are presented.

### 5.3 Conclusion

In this chapter, we especially focus on a practical aspect of tracking algorithms by analysing implementation differences and design issues between two different probabilistic tracking-by-detection algorithms — BPF-Adaboost with adaboost detections and KF-DPM with detections of deformable part models. It is important to know that these design choices matter significantly on the final performance of tracking algorithms. It is rather surprising that KF-DPM, which is a light-weight, simpler deterministic algorithm, outperforms a more complicated, feature rich implementation of BPF-Adaboost. Regardless of theoretical differences between these two filtering algorithms, we demonstrated how careful design choices and implementations impact largely on the performance of tracking algorithms.

It is possible to design and implement BPF differently with detections of the deformable part model, a different observation model and dynamical model. However, when a specific implementation of a simpler, deterministic algorithm such as KF-DPM works well, we choose not to engineer implementations of a rather more complex, approximation algorithm such as BPF-Adaboost.

## Chapter 6

# Self-Learning for Player Localization in Sports Video

Recent advances in object detection have enabled machines to detect many classes of objects. Statistical models based on the appearance of these classes such as faces, pedestrians, and cars — trained from extensive training sets and generalized with low error rates to unseen data in a highly generic manner — have been widely applied in the real world. Modern digital cameras often have a built-in face detection system to automatically focus on faces. Pedestrian detection has been employed for monitoring surveillance videos and supporting safer driving of cars. Behind the successful history of object detection systems, these statistical methods pose a major drawback — which is the issue addressed here, that they require a large amount of training data. In order to achieve performance levels that are high enough for practical applications, it is common that more than a million labelled instances are used for the training.

One way to resolve this issue is to employ abundant unlabelled data. In Chapters 3 and 4, we show that our active learning approach, which uses a novel interface to combine machine intelligence with human interventions, effectively improves a state-of-the-art classifier of [Felzenszwalb et al., 2009] by using additional unlabelled data from the Web. As the approach relies on input from a human oracle to improve its performance, there is still room to further reduce the amount of human labelling effort. An ideal solution is, if possible, to have no human involved in acquiring extra labels from novel data.

With abundant unlabelled data, crowdsourcing is a powerful tool to utilize extensive

human labour efficiently without much cost for obtaining abundant labels. LabelMe [Russell et al., 2008] and other interactive user interfaces on Amazon Mechanical Turk such as one by [Sorokin and Forsyth, 2008] and the Visipedia project [Welinder and Perona, 2010] are the front line of the work that addresses inexpensive acquisition of labels from a large pool of thousands of unlabelled images. Recently, crowdsourcing has also been utilized for annotating a collection of video data. Interactive annotation tools on the Web such as VATIC, a video annotation tool<sup>1</sup> by [Vondrick and Ramanan, 2011; Vondrick et al., 2010] and LabelMe video<sup>2</sup> [Yuen et al., 2009] have become publicly available in the computer vision community to foster large scale labelling of unlabelled video data. However, those crowdsourcing tools are designed primarily for reducing the overall labelling cost of time and money for obtaining labels from unlabelled data. They consider neither the impact of each label for improved performance of a classification model nor reducing the size of unlabelled data.

Another way to resolve the shortage of labelled data is to exploit both labelled and unlabelled data. There has been, especially in recent years, a significant interest in semi-supervised learning, which exploits both labelled and unlabelled data to efficiently train a classifier. The growing demand for more labelled data in recent statistical models has increased interest in semi-supervised learning mainly for its small labelling effort. Semi-supervised learning approaches have shown success in various domains of problems such as text classification [Nigam et al., 2000], handwritten digits recognition [Lawrence and Jordan, 2005] and object detection [Ali et al., 2011; Leistner et al., 2007; Rosenberg et al., 2005]. There is a vast amount of literature in methods of semi-supervised learning, which originally dates back to the work of [Scudder, 1965]. For additional background, we recommend a book of [Chapelle et al., 2006] and a comprehensive literature survey of [Zhu, 2008].

In this chapter, we address how to maximize the impact of labels by mainly selecting examples that are most likely misclassified by the current classification function, and to reduce the overall labelling cost by making the labelling process of novel data fully automatic. The objective is therefore to introduce a computer vision system that not only is capable of exploring unlabelled data and discovering an effective portion of the data for improving object detection, but also is capable of automatically labelling them. For achieving this, we

---

<sup>1</sup>The source code and software demo are available from <http://mit.edu/vondrick/vatic/>

<sup>2</sup>The source code and software demo are available from <http://labelme.csail.mit.edu/VideoLabelMe/>

consider related semi-supervised approaches in the context of object detection.

## 6.1 Weakly-supervised self-learning for player localization

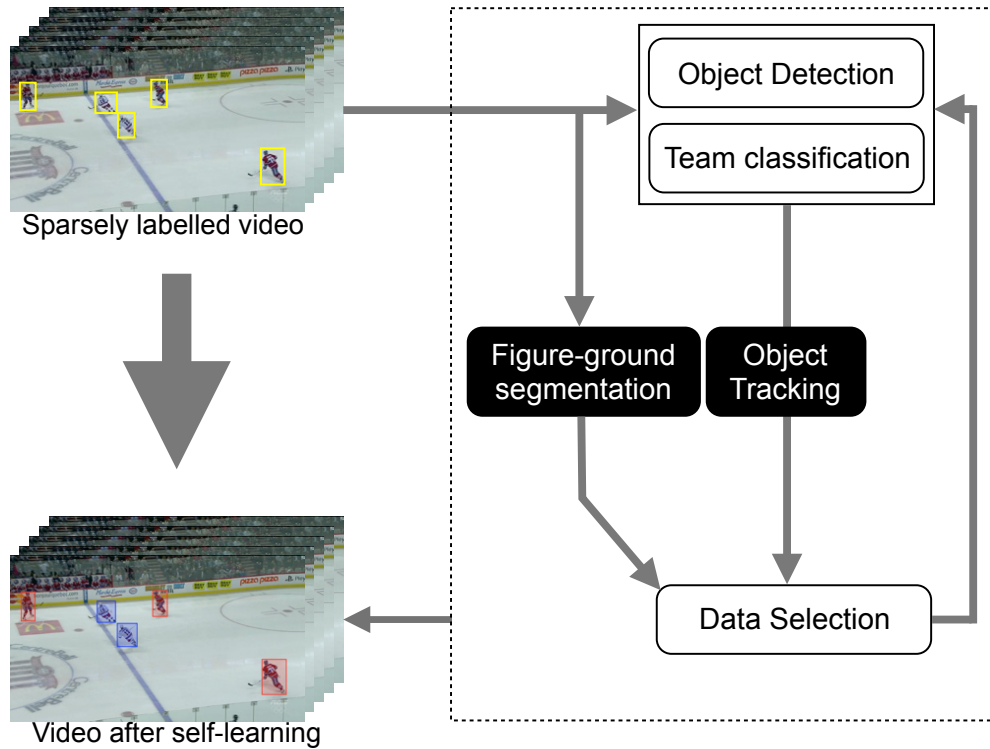
Given sparsely labelled video data that consists of  $n$  different video sequences  $\{V_i\}_{i=1}^n$ , the task is to train an initial model  $H : \mathcal{X} \mapsto \mathcal{Y}$  from a small set of labels  $\mathcal{L} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l)\}$  and exploit additional unlabelled data  $\mathcal{U} = \{\mathbf{x}_{l+1}, \dots, \mathbf{x}_{l+m}\}$  for improving the model, assuming that  $\mathbf{x} \in \mathcal{X}$ ,  $y \in \mathcal{Y}$ , and  $l \ll m$ . In this chapter, we will use hockey video data for learning an appearance-based model of hockey players. This can be viewed as a weakly-supervised learning problem because we deal with videos without localization of the target objects. Unlike most previous semi-supervised learning methods, we allow an unconstrained, unknown number of players that appear in each frame of a video sequence.

We propose a semi-supervised learning approach of self-learning, which is the most traditional semi-supervised learning method [Chapelle et al., 2006], to lower the requirement for extensive labelling. Self-learning is a wrapper algorithm that repeatedly uses a supervised learning method. It starts with a small set of labels to train the initial model. In each iteration, the model is used to evaluate unlabelled data and to obtain predictions. The model is then retrained with a selected portion of predictions as additional labels. This process is repeated until some stopping criterion is met.

We adopt part-based learning approach for learning the shape of sports players and the colour of their team. In order to use the temporal coherence of moving objects in videos, we use a tracking-by-detection approach based on a Kalman filter and combine motion information of objects with detection hypotheses based on the shape and colour of objects. For pruning detection hypotheses that are difficult to validate based on the motion and appearance of objects, the playing field is segmented from sports players based on a segmentation model of the colour of the field. Our novel criterion for selecting the unlabelled data therefore combines cues from figure-ground segmentation, object detection and tracking. Figure 6.1 shows multiple stages of our self-learning framework. We show that our approach is simple, yet effective in exploiting the unlabelled data for learning the appearance of sports players in broadcast sports videos.

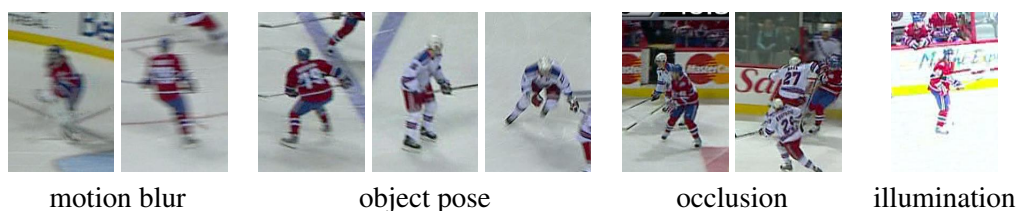
There are several reasons why we particularly focus on sports player detection in sports videos. First, sports videos are highly structured because the domain knowledge is rather specific (e.g. team colours, the player uniform, the colour of the playing field). But they are





**Figure 6.1: System overview.** The figure is best viewed in colour. This shows the overview of our self-learning system. Black boxes mean that models are not updated during the training process and treated as a black box. The system takes a sparsely labelled video with a small set of fully labelled image frames (ground-truth labels are shown in yellow boxes) as input and trains initial classification models. Our self-learning approach uses these models to explore the unlabelled portion of data, collecting additional training labels and update these models for improved performance. This process is repeated multiple times and produces a more complete set of labels in colour-specific tracklets (i.e., blue for the New York Rangers and red for the Montreal Canadiens) in the video.

still challenging enough to be an interesting problem. For example, Figure 6.2 shows several major challenges for detecting hockey players. Secondly, videos in sports — especially team sports such as hockey (6 on-field players per team), basketball (5 on-field players per team), and soccer (11 on-field players per team) — are a rich source of labels for learning the appearance of sports players since each frame of a video almost always contains multiple labels. Exploiting such data leads to an efficient label acquiring process for training a high performance object detector. Thirdly, accurate localization of sports players is a fundamental requirement for tackling other interesting problems such as action recognition and player recognition. Last but not least, to the best of our knowledge, our work is the first large scale study of a self-learning framework for learning the appearance of sports players in videos. We first explore hockey videos and then explore basketball videos later in this chapter.



**Figure 6.2: Challenges in player detection.**

### 6.1.1 System overview

Given a small set of labelled data, our goal is to improve classifiers automatically by discovering additional labels from unlabelled data. Throughout the self-learning process, we maintain two disjoint sets: a set of labelled bounding windows  $\mathcal{L}$  and a set of unlabelled bounding windows from unlabelled images  $\mathcal{U}$ .

Our self-learning system has several stages as shown in Figure 6.1. The training procedure starts from initializing a small set of labelled images and a large set of unlabelled images from sparsely labelled video data. Then the system iterates over the following steps. First, a small set of labelled data is used to train initial models for detecting players and classifying their team colour (Section 6.3 and Section 6.4.1) Second, these appearance-based models are applied to the unlabelled data and generate a set of detection bounding windows. Third, these bounding windows are linked by a tracking algorithm and generate

a set of tracklets (Section 6.5). Finally figure-ground segmentation is applied to validate these tracklets. The resulting set of tracklets is used as additional labels to re-train current classification models. Algorithm 6.1 summarizes this iteration process.

---

**Algorithm 6.1 : Self-learning for player localization in videos**

---

Given training video sequences  $\{V_i\}_{i=1}^n$ , randomly select  $m$  labelled images that contain an initial set of labelled data  $\mathcal{L} = \{(x_1, y_1, c_1), \dots, (x_l, y_l, c_l)\}$  where  $x$  is a window descriptor,  $y$  is a class label, and  $c$  is a team colour label. The number of self-learning sessions is set as  $n_s = 5$ .

---

- 1: **Initialize**  $\mathcal{U}$  with all image frames that are unlabelled in  $\{V_i\}_{i=1}^n$ .
  - 2: **for**  $n_s$  self-learning sessions **do**
  - 3:   **Training classifiers:**  
       Given labelled data  $\mathcal{L}$ , train a part-based player detector (Section 6.3) and team colour classifiers. (Section 6.4.1)
  - 4:   **Player detection and team classification:**  
       Run the player detector for unlabelled data  $\mathcal{U}$  to get detection bounding windows.  
       Run team colour classifiers on those detection bounding windows and eliminate ones that do not have any of team colour labels Figure 6.4.
  - 5:   **Player tracking:**  
       Run a Kalman filter to link detection bounding windows (Section 6.5) and obtain a set of  $k$  tracklets  $\{\mathcal{T}\}_{j=1}^k$ .
  - 6:   **Data selection:**  
       Select a new dataset  $\mathcal{L}_{new}$ . (Section 6.6)  
        $\mathcal{L} = \mathcal{L} \cup \mathcal{L}_{new}$
  - 7: **end for**
- 

## 6.2 Semi-supervised learning in videos

Many algorithms in semi-supervised learning assume that the unlabelled data are independent samples. However, frames of video are strongly related due to spatio-temporal dependencies. In object detection, the task is to find one or more bounding windows, each of which precisely determines the location and size of an instance of an object class in an image. In a video sequence, the trajectory of object instances, defined by the location of the bounding windows, suggests the spatio-temporal structure of subsequent labels.

In order to exploit the dependent structure of the video data, several tracking-by-detection approaches [Babenko et al., 2009; Kalal et al., 2010; Leistner et al., 2011] have been pro-

posed to learn an appearance model of an object from videos. These approaches have the stringent assumption of having only one instance of the target object class in each frame of a video sequence. Such an assumption strictly limits applications to detection of a single instance of the target object class, where an instance with the highest confidence is identified as a positive label and all remaining instances are labelled as negative. For learning the appearance of an object class such as pedestrians or faces, videos that contain multiple pedestrians in each frame are much more effective than videos with one person in each frame, because they have a dense collection of potential labels. But localization of multiple target objects remains difficult, and it prevents most of tracking-by-detection approaches from exploiting unlabelled data that are available from such videos. Nonetheless, there are a few approaches that have considered exploiting unlabelled video data with multiple target objects.

Ramanan et al. [2007] proposed a semi-supervised method for building a large collection of labelled faces from archival video of the television show Friends. Their final collection contains 611,770 faces, which is the largest existing collection of faces to date in academia. Their approach used the Viola and Jones face detector to detect faces, grouping them with colour histograms of body appearance (i.e, hair, face, and torso) and tracking them using a part-based colour tracker for multiple people in videos. Although their approach is effective with large scale data, they performed only one iteration of exploring the unlabelled data for building a large collection of faces and never used the acquired collection for improving the classifiers they used.

Very recently, [Ali et al., 2011] implemented self-learning on sparsely labelled videos, which allows any number of instances of the target object class. Their approach exploits spatio-temporal information of objects for improving an appearance-based object detector. Given a sparse labelling of the video sequence, an initial model is trained by a boosting algorithm with a small set of labelled instances. The rest of the unlabelled portion of the video is used for collecting additional labels that are consistent with the current classification model and with the constraint of continuous motion of target objects. Based on the motion constraint, their approach extracted admissible trajectories of the target objects and used them to distinguish positive instances from negative ones. After a few iterations of this process, they showed noticeable improvement of performance on pedestrian detection in videos as well as on cell detection in microscopy image sequences.

To the best of our knowledge, the work of [Ali et al., 2011] has been the first work that

addresses localization of multiple target objects in videos for improving object detection. However, their approach differs significantly from ours. They used a boosting algorithm to exploit the temporal coherence of videos, whereas we adopt a latent SVM formulation for learning the appearance of objects in our self-learning framework and use figure-ground segmentation as additional information to validate the unlabelled data. Furthermore, we use broadcast footage of sports which is much more challenging than their surveillance data of a stationary camera view, where pedestrians are walking by with very simple, predictable motions without any interactions.

### 6.3 Player detection

In order to detect hockey players, we adopt the recent latent SVM (LSVM) approach of [Felzenszwalb et al., 2009]. The goal of a supervised learning algorithm is to take  $n$  training samples and design a classifier that is capable of distinguishing  $M$  different classes. For a given training set  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$  with  $\mathbf{x}_i \in \mathbb{R}^N$  and  $y_i \in \{-1, +1\}$  in their simplest form with two classes, LSVM is a classifier that scores a sample  $\mathbf{x}$  with the following function,

$$f_{\beta}(\mathbf{x}) = \max_{\mathbf{z} \in \mathbf{Z}(\mathbf{x})} \beta \cdot \Phi(\mathbf{x}, \mathbf{z}) \quad (6.1)$$

Here  $\beta$  is a vector of model parameters and  $\mathbf{z}$  are latent values. The set  $\mathbf{Z}(\mathbf{x})$  defines possible latent values for a sample  $\mathbf{x}$ . Training  $\beta$  then becomes the optimization problem already presented in Equation 3.2. For obtaining a binary label for  $\mathbf{x}$ , we have the decision function,  $\text{sign}(h(\mathbf{x}))$ , where

$$h(\mathbf{x}) = f_{\beta}(\mathbf{x}) + b \quad (6.2)$$

The function  $h$  gives a raw LSVM score in real values. We approximate the posterior probability  $P(y = 1|\mathbf{x})$  of the decision function in a parametric form of a sigmoid [Lin et al., 2003; Platt, 2000]<sup>3</sup>.

$$P(y = 1|\mathbf{x}) \approx P(y = 1|f) = \frac{1}{1 + \exp(fA + B)} \quad (6.3)$$

---

<sup>3</sup>Lin et al. [2003] extended the original work of Platt [2000] and proposed an improved algorithm that theoretically converges and avoids numerical difficulties.

where  $f = f_\beta(\mathbf{x})$ .

## 6.4 Colour classification

In sports, colour information such as the colour of the teams and the colour of the playing field is one of the most useful sources of domain knowledge. We use a part-based colour classification to identify the team colour of each player and figure-ground segmentation to distinguish pixels of players from those of the playing field.

### 6.4.1 Team classification

Our shape-based deformable part model (DPM) gives a tight bounding window of the object (i.e., a hockey player) as well as a set of smaller bounding windows of its corresponding parts. Given these bounding windows as prior knowledge, the model learns a colour classification function based on deformable parts with the following function:

$$f_\gamma(\mathbf{x}) = \gamma \cdot \Phi(\mathbf{x}, \mathbf{z}_\beta) \quad (6.4)$$

where  $\gamma$  is a vector of model parameters and  $\mathbf{z}_\beta$  are latent values specified by the shape-based DPM detector. Following [Lu et al., 2009; Okuma et al., 2004; Pérez et al., 2002], we use Hue-Saturation-Value (HSV) colour histograms. Thus, a feature vector  $\mathbf{x}$  is composed of a set of HSV colour histograms, each of which has  $N = N_h N_s + N_v$  bins and corresponds to a unique part of deformable part models. A distribution  $K(\mathbf{R}) \triangleq \{k(n; \mathbf{R})\}_{n=1, \dots, N}$  of the colour histogram in a bounding window  $\mathbf{R}$  is given as follows:

$$k(n; \mathbf{R}) = \eta \sum_{d \in \mathbf{R}} \delta[b(d) - n] \quad (6.5)$$

where  $d$  is any pixel position within  $\mathbf{R}$ , and  $b(d) \in \{1, \dots, N\}$  as the bin index.  $\delta$  is the delta function. We set the size of bins  $N_h$ ,  $N_s$ , and  $N_v$  as 10. The normalizing constant  $\eta$  ensures that all the bin values are  $[0, 1.0]$ . It is important to note that  $K(\mathbf{R})$  is not a probability distribution and is only locally contrast normalized<sup>4</sup>,  $\max K(\mathbf{R}) = 1.0$ . For obtaining a binary label for  $\mathbf{x}$ , we have the decision function,  $\text{sign}(h(\mathbf{x}))$ , where

---

<sup>4</sup>In our experiments which are not shown here, we tested our classification model with the distribution of the colour histograms which are normalized to be probability distributions. However, results were much worse than ones with local contrast normalization.

$$h(\mathbf{x}) = f_{\gamma}(\mathbf{x}) + b \quad (6.6)$$

We train a colour model for each team label: “MTL” for Montreal Canadiens, “NYR” for New York Rangers, and “referee” for referees. Figure 6.3 shows two component deformable part models for the Montreal Canadiens team. The posterior probability of the decision function for each colour classification model is approximated by fitting a sigmoid function [Lin et al., 2003; Platt, 2000]. Finally, our team colour classification function is formulated as the maximum likelihood of three binary colour classification models.

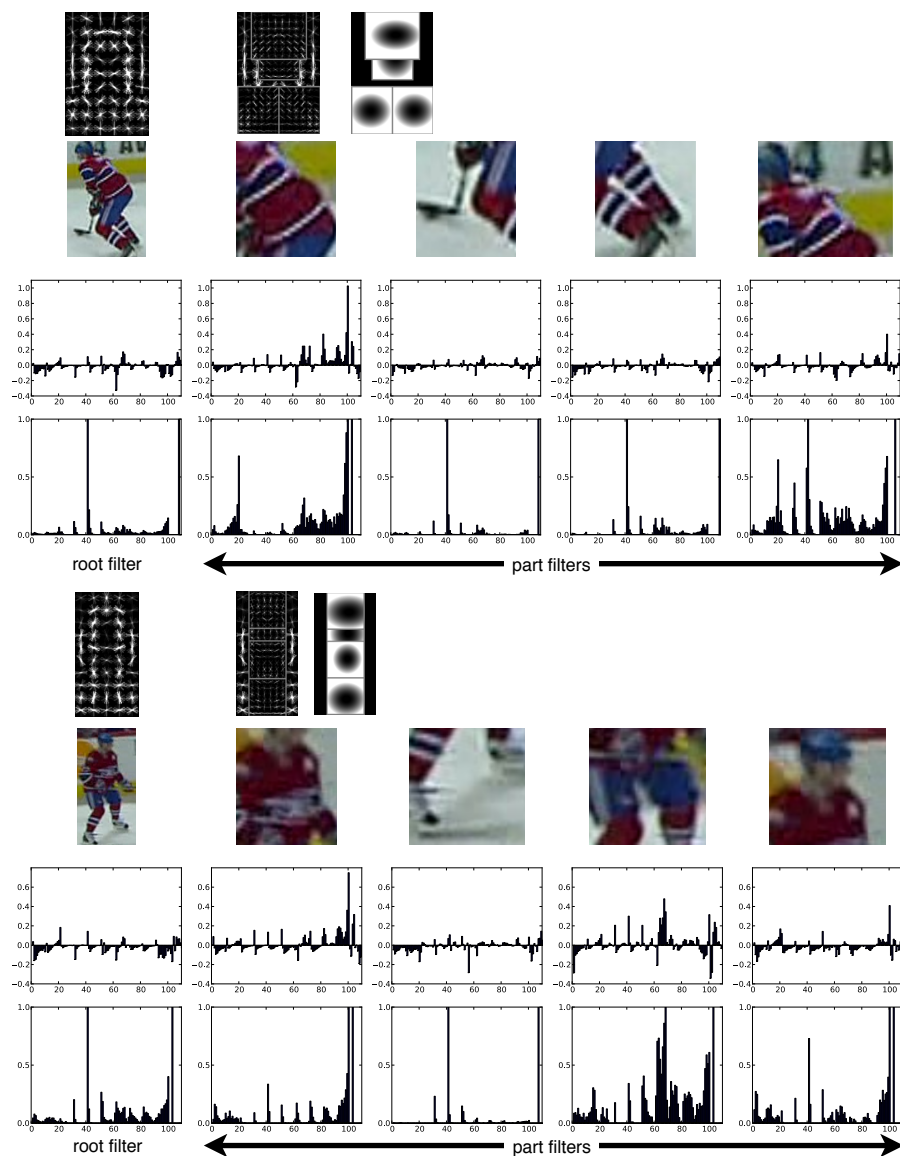
$$y^* = \operatorname{argmax}_{y \in \mathcal{Y}} P(y|\mathbf{x}, \mathbf{z}_{\beta}) \quad (6.7)$$

where  $y$  is a team label and  $\mathcal{Y} = \{\text{“MTL”, “NYR”, “ref”, “others”}\}$ . These part-based colour models are highly discriminative since it uses the learned latent values  $\mathbf{z}_{\beta}$  (i.e., location and size of multiple parts of an object) based on the shape-based DPM detector. Furthermore, these colour models are efficiently trained because they do not have to optimize over a large space of latent values, which is the bottleneck of training the latent SVM.

For team colour classification, part-based colour models are particularly effective when two teams, the Montreal Canadiens and the New York Rangers, have a similar distribution of colours (e.g., red and blue) in their uniform (Figure 6.2). Figure 6.3 shows how multi-part weighted histograms preserve the spatial information of colour distributions, where a single holistic representation cannot. In the figure, there are two different part-based colour models for the Montreal Canadiens, where each model has weighted multi-part colour histograms. Parts with more discriminative colour are learned to have higher weights. Figure 6.4 shows results of team colour classification, which improves detection results of the shape-based model by suppressing those detection windows that do not have the learned team colour labels. In this case, we had 79% precision and 57% recall without team classification (a) and 89% precision and 54% recall with team classification by suppressing false positive detection windows (b).

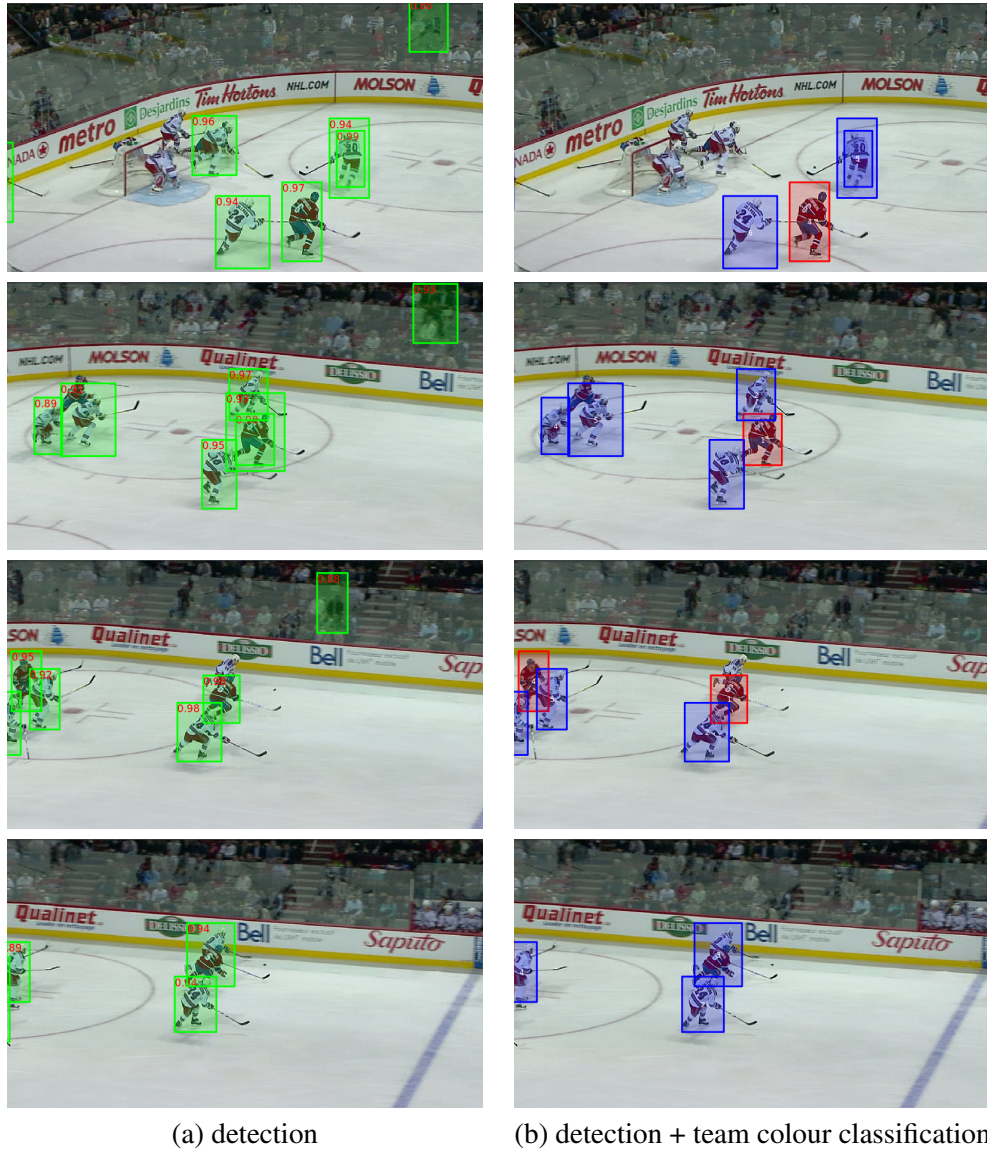
#### 6.4.2 Figure-ground segmentation

For most sports, the background is relatively uniform and distinctive because the playing field is specifically designed to be distinguishable from objects on the field. We developed



**Figure 6.3: Colour-based deformable part models.** This shows a mixture of two part-based colour models for the Montreal Canadiens team. For each model, the top row shows the root filter, part filters, and deformation model. The second row shows corresponding image regions of the object. The distribution of their learned weights and HSV colour histograms are shown respectively in the third and fourth row. Note noticeably higher weights on those parts that are particularly *discriminative* for classification (e.g., the second column in the top, the second and fourth in the bottom)





**Figure 6.4: Player detection and team colour classification results.** This figure is best viewed in colour. This shows results of player detection and team colour classification. Detection bounding windows are shown in green boxes in (a) with their detection confidence in the upper left corner of these bounding windows, and with team colour classification in red (Montreal Canadiens) and blue boxes (New York Rangers) in (b). Note that team colour classification suppresses false positive detections in the background.

an interactive labelling tool to learn a figure-ground segmentation model based on a boosting algorithm. Given a small set of manually labelled foreground pixels and background pixels on the first image, we used the OpenCV<sup>5</sup> implementation of Gentle Adaboost to learn a set of 150 weighted decision trees<sup>6</sup> where the maximum depth of these trees is 10. We then use the initial model on an additional few images, interactively labelling wrongly classified pixels and update the model with these additional labels. The process is repeated a few times with no more than 5 images.

After training, we have the final decision function of the boosting model,  $H(\mathbf{d}_i) = \text{sign}(\sum_{l=1}^{150} \alpha_l h_l(\mathbf{d}_i))$  where  $h_l$  is a weak feature,  $\alpha_l$  is the corresponding weight and  $\mathbf{d}_i$  denotes a pixel location  $(x_i, y_i)$  in an image  $I_j$ . The value of  $H(\mathbf{d})$  is positive when  $\mathbf{d}$  is classified as a foreground pixel which belongs to a player, and otherwise negative when  $\mathbf{d}$  is classified a background pixel which belongs to the playing field. Figure 6.5 shows the result of figure-ground segmentation. Although it is not perfect, the result is strong enough to eliminate most obvious false positive instances and refining a set of additional instances of objects from unlabelled data.

In our informal experiments which are not shown here, we tested our self-learning sessions with and without figure-ground segmentations in order to see the impact of our proposed figure-ground segmentation model. The results showed that the impact of figure-ground segmentations was small but not negligible, and produced an improved detection performance with a few percent increase in average precision. This small increase was due to the elimination of a small number of false positive detections on the play field that are difficult to distinguish solely by the team classification.

We also tested a saliency measure called “objectness” in the recent work of [Alexe et al., 2010] because it has been used in state-of-the-art weakly supervised approaches for localizing generic objects [Deselaer et al., 2010; Lee and Grauman, 2011; Vezhnevets et al., 2011]. “Objectness” is a trainable saliency measure that represents *unique* characteristics of objects in terms of several image cues such as multi-scale saliency of the spectral domain of an image, colour contrast, edge density and closed boundaries of an object. It combines

<sup>5</sup>We used the latest release of a library of programming functions for real time computer vision, OpenCV 2.3.1, which is accessible on the Web: <http://opencv.willowgarage.com/wiki/>.

<sup>6</sup>Learning and inference of the model can be further sped up by using decision stumps (i.e., one level decision tree) instead of multi-level decision trees, or reducing the number of weak features. But the speed is not an issue in our experiments since the figure-ground segmentation is done off-line.

these image cues in a naïve Bayes model and facilitates separation between the appearance of the foreground class (i.e., hockey player in this case) and the background class (i.e., all encompassing background) where a higher “objectness” score to rectangular bounding windows indicates the presence of the object of interest. However, “objectness” did not work well in a hockey video mainly due to a small size of hockey players and weak contrast of the colour of hockey players and the rink. Figure 6.6 shows failure results of “objectness” on our hockey data. We trained the model from a set of 20 randomly selected hockey images, using their off-the-shelf program<sup>7</sup> with default parameters.

## 6.5 Player tracking

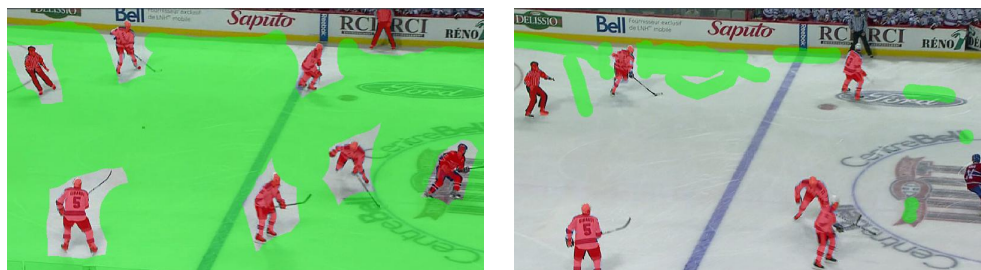
Once detected players have their team label, the next step is to associate detected bounding windows into a set of “tracklets” where a tracklet represents a sequence of bounding windows that share the same identity over time. To achieve this, we employ a tracking-by-detection approach and adopt the tracking system of [Lu et al., 2011] based on a Kalman filter [Kalman, 1960]. For multi-target tracking in sports, we have also developed a particle filtering approach, called the boosted particle filter (BPF) [Okuma et al., 2004]. However, we used Lu et al. [2011]’s Kalman filter tracking system because it is simpler than BPF and also works better in sports data. In Chapter 5, we explain comparison results of BPF and Lu et al. [2011]’s Kalman filter tracker in a hockey video.

In our self-learning process, we do not update parameters of a tracking model and treat player tracking as a black box. Therefore, our system also works with other tracking-by-detection approaches such as a data-driven MCMC [Khan and Shah, 2006], or the unscented Kalman filter [van der Merwe et al., 2000].

Given a set of predicted detection bounding windows with their predicted team label  $\{(\mathbf{R}_1, c_1), \dots, (\mathbf{R}_n, c_n)\}$ , we divide the set into three subsets based on their team labels.  $\mathbf{R} = \{x, y, w, h\}$  denotes a rectangular box with the centre  $(x, y)$ , width  $w$  and height  $h$ . The team colour label is denoted by  $c \in \{\text{“MTL”}, \text{“NYR”}, \text{“ref”}\}$ . In each set of colour-specific detection bounding windows, tracklets are initialized with the first predicted bounding window for each target. To associate any detected bounding window to tracklets, bi-partite matching is performed based on the following cost function:

---

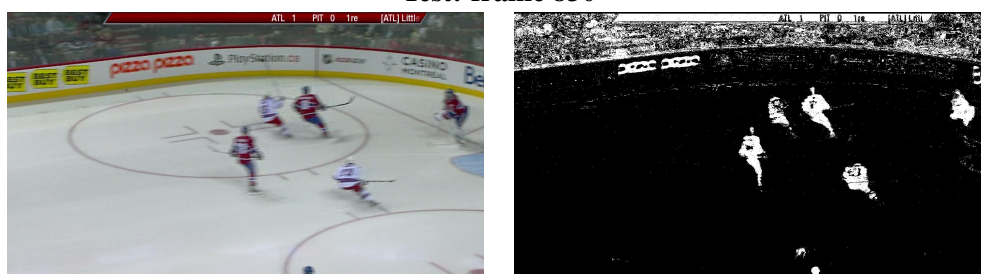
<sup>7</sup>The source code is available from <http://www.vision.ee.ethz.ch/~calvin/objectness/>



**Training: interactively labelled pixels**

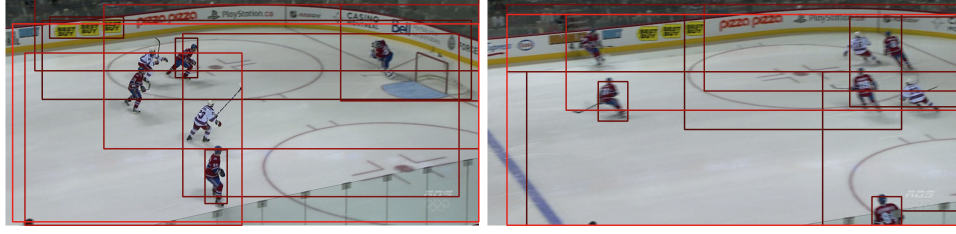


**Test: frame 830**



**Test: frame 970**

**Figure 6.5: Figure-ground segmentation results.** This is best viewed in colour. It shows results of figure-background segmentation on the hockey rink. The first row shows the mask image of interactively labelled pixels where the red colour represents the foreground and the green colour represents the background. Note that the image on the right shows sparse labels where a user can choose misclassification pixels by the initial model that is trained on the first set of labels on the left. The segmentation model is trained with 5 training images. The middle and bottom rows show the results of segmentation by the trained model.



**Figure 6.6: Results of “objectness” on hockey data.** This shows results of “objectness” on two hockey images. The model is trained on a separate set of randomly selected 20 images with ground truth annotations of bounding windows for hockey players. Each image shows 10 highest scoring “objectness” bounding windows in red where the brighter means the stronger “objectness”. Note that “objectness” fails to find localization bounding windows for most of the hockey players. There are a few successful cases only with players in the Montreal Canadiens (i.e., red jersey) who have a higher colour contrast than ones in the New York Rangers (i.e., white jersey).

$$C_{i,j} = \sqrt{(\mathbf{R}_i - \mathbf{D}_j)^T (\mathbf{R}_i - \mathbf{D}_j)} \quad (6.8)$$

where  $C_{i,j}$  is the cost of associating the  $i$ -th prediction bounding window  $\mathbf{R}_i$  with the  $j$ -th detection bounding window  $\mathbf{D}_j = \{x', y', w', h'\}$ .

In a Kalman filter, the state sequence  $\{\tilde{\mathbf{x}}_t; t \in \mathbb{N}\}$ ,  $\tilde{\mathbf{x}}_t \in \mathbb{R}^{n_{\tilde{\mathbf{x}}}}$ , is assumed to be a linear dynamical process with a Gaussian initial distribution  $p(\tilde{\mathbf{x}}_0)$  and a Gaussian transition distribution  $p(\tilde{\mathbf{x}}_t | \tilde{\mathbf{x}}_{t-1})$ , where  $n_{\tilde{\mathbf{x}}}$  is the dimension of the state vector. Here,  $\tilde{\mathbf{x}} = \{l_x, l_y, l_w, l_h\}$  where  $\{l_x, l_y\}$  denotes the centre of the bounding box and  $\{l_w, l_h\}$  are its width and height in the image coordinate system. The observations  $\{\tilde{\mathbf{y}}_t; t \in \mathbb{N}\}$ ,  $\tilde{\mathbf{y}}_t \in \mathbb{R}^{n_{\tilde{\mathbf{y}}}}$ , are conditionally independent given the process  $\{\tilde{\mathbf{x}}_t; t \in \mathbb{N}\}$  with a Gaussian marginal distribution  $p(\tilde{\mathbf{y}}_t | \tilde{\mathbf{x}}_t)$ , where  $n_{\tilde{\mathbf{y}}}$  is the dimension of the observation vector. In our case,  $\tilde{\mathbf{y}}$  is a detection bounding window that is associated by the bi-partite matching.

Following Lu et al. [2011]’s tracking system, the observation model is assumed to be a Gaussian distribution centred at a detection window,  $p(\tilde{\mathbf{y}}_t | \tilde{\mathbf{x}}_t) = \mathcal{N}(\tilde{\mathbf{y}}_t | \tilde{\mathbf{x}}_t, \Sigma_{\tilde{\mathbf{y}}})$ . The transition distribution is assumed to be a linear-Gaussian: Given the current time index  $t$ ,  $p(\tilde{\mathbf{x}}_t | \tilde{\mathbf{x}}_{t-1}, t) = \mathcal{N}(\tilde{\mathbf{x}}_t | \hat{\mathbf{x}}, \Sigma_{\tilde{\mathbf{x}}})$  where the predictive bounding box  $\hat{\mathbf{x}}$  is defined as follows:

$$\hat{\mathbf{x}} = t\mathbf{a}_t + \mathbf{b}_t \quad (6.9)$$

where linear regression parameters  $(\mathbf{a}_t, \mathbf{b}_t)$  are both  $4 \times 1$  vectors. These regression parameters are learned on-line based on the previous  $M$  posterior mean estimates by minimizing the following least-square function:

$$\min \sum_{i=1}^M (1 - \alpha)^i \|(t - i)\mathbf{a}_t + \mathbf{b}_t - \tilde{\mathbf{x}}_{t-i}\|^2 \quad (6.10)$$

where  $0 \leq \alpha < 1$  is a positive constant. When there aren't enough training data (e.g.,  $M < 10$ ), the first-order auto-regressive dynamic model (i.e.,  $\hat{\mathbf{x}} = \tilde{\mathbf{x}}_{t-1}$ ) is used. The parameters of  $\Sigma_{\tilde{\mathbf{y}}}$  and  $\Sigma_{\tilde{\mathbf{x}}}$  are set manually. After one-pass filtering through all detection bounding windows, we obtain a set of  $k$  tracklets  $\{\mathcal{T}\}_{j=1}^k$ .

## 6.6 Data selection

Thus far, a set of tracklets  $\{\mathcal{T}\}_{j=1}^k$  is obtained by combining detection and tracking results of hockey players. These tracklets are used as a pool of candidate data  $\mathcal{C}$  from which we collect a set of training labels for improving generalized performance of classification models. Since this selection process is fully automatic, we need a selection criterion which effectively discovers additional training labels without accumulating incorrect labels.

Our selection criterion combines several image cues including detection, colour classification, tracking of players, and pixel-wise figure-ground segmentations. The selection process is performed with the following steps. First, we prune away short tracklets with less than 10 bounding windows because these tracklets are often produced by very sparse detection results, and most likely include incorrect labels. After pruning, we have a refined set of tracklets  $\{\mathcal{T}\}_{j=1}^m$  where  $m < k$ . We initialize a pool of candidate data  $\mathcal{C}$  with bounding windows of these tracklets. Second, we compute the shape confidence of these predicted bounding windows by running our shape-based DPM detector on each bounding box. Third, we compute a foreground score  $a_f \in [0, 1.0]$  to measure a proportion of foreground pixels (i.e., player pixels) within each predicted bounding window  $\mathbf{R}_p$  in the candidate data  $\mathcal{C}$ :

$$a_f = \frac{1}{\text{area}(\mathbf{R}_p)} \sum_{\mathbf{d}_i \in \mathbf{R}_p} f(\mathbf{d}_i) \quad (6.11)$$

where  $area(\mathbf{R}_p)$  denotes the area of the bounding window  $\mathbf{R}_p$  in terms of the total number of pixels within the window, and  $f$  is a binary function which uses the decision value of our figure-ground segmentation model  $H$  as follows:

$$f(\mathbf{d}_i) = \begin{cases} 1 & \text{if } H(\mathbf{d}_i) \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (6.12)$$

We use a foreground score  $a_f$  to determine whether or not the corresponding predicted bounding window  $\mathbf{R}_p$  is added to a set of additional data  $\mathcal{L}_{new}$ . For making this decision, we use labelled data and derive a set of two thresholds  $\tau_{lower} = \mu_{a_f} - \sigma_{a_f}$  and  $\tau_{upper} = \mu_{a_f} + \sigma_{a_f}$  where  $\mu_{a_f}$  is a mean foreground score and  $\sigma_{a_f}$  is a standard deviation. These thresholds represent how likely  $\mathbf{R}_p$  contains the foreground object in terms of the proportion of foreground pixels within the window and are computed based on all positive instances in ground-truth data. Consequently, we add a predicted bounding window  $\mathbf{R}_p$  to  $\mathcal{L}_{new}$  if  $\tau_{lower} \leq a_f \leq \tau_{upper}$ .

The selected candidate data  $\mathcal{L}_{new}$  is added to labelled data  $\mathcal{L}$  by simply taking the intersection of these two datasets,  $\mathcal{L} = \mathcal{L} \cap \mathcal{L}_{new}$ . This intersection produces many bounding windows that significantly overlap with each other. We reduce these duplicates by prioritizing those instances in  $\mathcal{L}_{new}$  and discarding existing instances in  $\mathcal{L}$ . Assuming that classification models improve every iteration, we utilize this process for eliminating some of the incorrect localization labels. However, such an assumption may not hold if the selection process accumulates too many noisy labels. In the following experiments, we empirically show that our assumption still holds in our self-learning framework.

---

**Algorithm 6.2 : Data selection**

Given a set of tracklets  $\{\mathcal{T}\}_{j=1}^k$  and a figure-ground segmentation model  $H$ , the goal is to select a portion of data as candidate labels for the next iteration of self-learning as described in Algorithm 6.1. Every iteration, we set the maximum number of additional labels to be added as  $n_{max} = 2000$ .

---

1: **Tracklet selection:**

Discard any tracklet whose length is less than 10.

Initialize a pool of candidate data  $\mathcal{C}$  with a set of all bounding windows from  $\{\mathcal{T}\}_{j=1}^m$ .

2: **Estimate the shape confidence of selected tracklets:**

Run our shape-based DPM detector for each bounding window in  $\mathcal{C}$ .

Sort them in the ascending order of the predicted shape confidence.

3: **Apply figure-ground segmentation:**

For each bounding window  $R_p$  in the sorted order, compute figure-ground segmentation score  $a_f$  Equation 6.11.

4: **Final selection:**

Select a new dataset  $\mathcal{L}_{new}$ .

Merge  $\mathcal{L}_{new}$  with the existing dataset  $\mathcal{L}$ :  $\mathcal{L} = \mathcal{L} \cup \mathcal{L}_{new}$

but only selects no more than  $n_{max}$  additional labels that do not overlap with the current set of labels.

---

## 6.7 Experiments

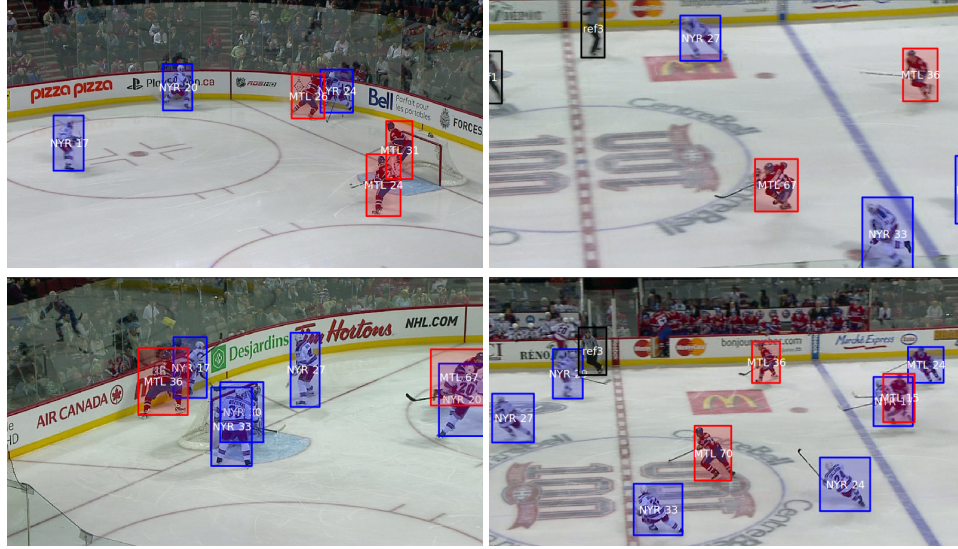
This section summarizes our self-learning experiments on hockey data.

### 6.7.1 Data

Our system was tested on our hockey dataset that consists of 7 different video sequences which sum to 4,627 image frames of a broadcast footage. The data are split into two separate sets: 3 sequences (2,249 frames in total) for training and 4 sequences (2,378 frames in total) for testing. In the training data, the annotations are given in rectangular boxes with the category label, identification (i.e., the number of their jersey) and team colour label. Figure 6.7 shows screen shots of annotation examples in different image frames. There are complex interactions among hockey players (i.e., occlusions and collisions), constant movements of a broadcast camera and rapid motions of hockey players. They are potential challenges for achieving localization of hockey players.

In our experiments, we prepared 8 different sets of fully labelled images: 7 sets of  $m$  randomly selected fully labelled images where  $m = \{5, 10, 20, 40, 100, 500, 1000\}$  and the





**Figure 6.7: Ground-truth annotation data for hockey players.** This shows example ground-truth annotation data of hockey players. The resolution of these images is  $960 \times 540$ . Red bounding boxes are for the Montreal Canadiens and blue bounding boxes are for the New York Rangers. Black bounding boxes are for referees. Each player and referee has a unique combination of a team name and a jersey number as they are shown in white characters at the centre of the bounding box.

fully supervised set of all 2,249 images. For each initial labelled dataset, we first trained the initial shape-based DPM detector and part-based colour classifiers. Then we applied our self-learning framework to collect additional training labels from the unlabelled data and improve initial classifiers iteratively up to four iterations. In the following sections, we show results of player detection and tracking and improved performance with our system.

### 6.7.2 Player detection

We use the PASCAL VOC criterion [Everingham et al., 2010] for evaluating our detection results because it has been well defined and widely used in the vision community. The predicted detection  $R_p$  must satisfy the overlap ratio  $o$  between  $R_p$  and the matched ground-

truth bounding box  $\mathbf{R}_{gt}$  to be over 50%:

$$o = \frac{area(\mathbf{R}_p \cap \mathbf{R}_{gt})}{area(\mathbf{R}_p \cup \mathbf{R}_{gt})} \quad (6.13)$$

where  $area(\mathbf{R}_p \cap \mathbf{R}_{gt})$  represents the intersection of the predicted bounding box  $\mathbf{R}_p$  and the matched ground-truth bounding box  $\mathbf{R}_{gt}$  and  $area(\mathbf{R}_p \cup \mathbf{R}_{gt})$  for the union of these bounding boxes. The threshold of the overlap ratio is set at 50% to take some inaccuracies in the ground-truth bounding boxes into account for evaluation. For instance, manually labelled ground-truth bounding boxes may be somewhat subjective in a way that legs or arms in some poses of hockey players may not be bounded by a rectangular window.

Detection results of a method are assigned to the ground-truth objects where a higher confidence output has priority over the assignment in the case of multiple detections to the same object. That is, multiple detections except for the highest scoring one become false positive detections.

For the overall performance evaluation, the shape of the precision-recall curve is computed by average precision (AP). Precision means the proportion of the output bounding windows that have the positive class. Recall is defined as the proportion of the correctly matched objects among all positive object instances in the ground-truth data. The AP represents the shape of the precision-recall curve, and is defined as the mean precision over a set of 11 equally spaced recall intervals  $\{0, .1, .2, \dots, 1.0\}$  [Everingham et al., 2010]:

$$AP = \frac{1}{11} \sum_{r \in \{0, .1, \dots, 1.0\}} p_{interp}(r) \quad (6.14)$$

The precision  $p_{interp}$  at each recall level  $r$  is *interpolated*. The interpolation takes the maximum precision measured for a method for which the corresponding recall exceeds  $r$ :

$$p_{interp}(r) = \max_{\tilde{r}: \tilde{r} \geq r} p(\tilde{r}) \quad (6.15)$$

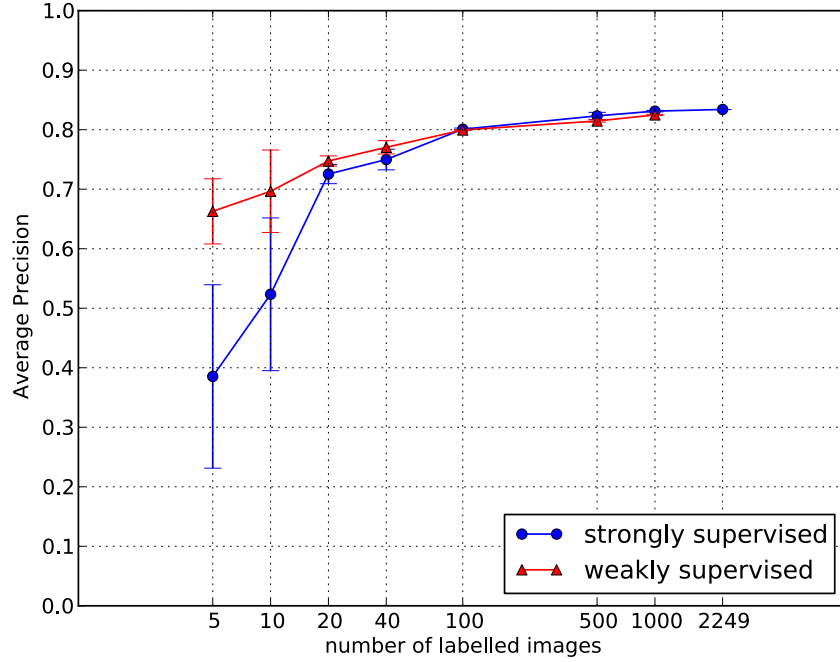
where  $p(\tilde{r})$  is the measured precision at recall  $\tilde{r}$ . The AP requires precisions at all levels of recall and penalizes a method that has high precision only at a specific level of recall. Therefore, average precision is a fair evaluation criterion of classification methods in object detection.

Figure 6.8 shows the result of our system on our hockey data. We ran the entire process

three times and show the mean and variance for each labelled dataset. The blue line shows the baseline performance based on only fully supervised data. The red line shows the performance after our system collected additional data from unlabelled frames of the video. The results show a large performance gain — about 20% in the mean average precision — in cases with a small amount of labelled images (e.g., 5 and 10 labelled images). However, there is almost no performance improvement in larger labelled datasets (e.g., 100, 500, and 1,000 labelled images).

Table 6.1 shows the average number of labels used for each labelled dataset. In the table, our self-learning system collected around 7,000 labels on average in the case of only 5 labelled images. A dataset of 1,000 labelled images also contains around 7,000 labels on average. Using our self-learning system on 5 labelled images gives as much training data as using a dataset of 1,000 labelled images. However, our self-learning system with 5 labelled images will not achieve the level of performance that is competitive with one of 1,000 labelled images as discussed below.

We suggest that there are two reasons for this performance gap. Firstly, our self-learning system does not collect additional negative data because the presence of multiple object instances of the same class in an image makes it difficult to distinguish player regions from non-player regions in the image. Negative data are sampled from image regions that do not significantly overlap with ground truth bounding windows. Therefore, our system never used the unlabelled data as the source of negative data. However, it is possible to collect negative data by filtering out some of detected bounding windows that neither belong to trajectories of target objects nor agree with the shape and colour of the target object class. The idea is similar to [Ali et al., 2011]. But our approach uses different models, and works on videos that do not have a stationary background. Secondly, our additional positive data contain noisy labels as shown in Figure 6.10. Noisy labels are mostly due to imperfect localization that is especially evident in the least confident bounding windows. These noisy labels may have had a negative impact on the detection performance. Therefore, collecting more negative data may improve the detector and reduce noisy labels, improving the performance even more. We consider collecting more negative data as a future direction of this work.



**Figure 6.8: Detection result of our weakly self-learning system in hockey videos.**

The blue line shows the baseline performance based on only labelled datasets. The red line shows the performance after four self-learning iterations of collecting additional labels from unlabelled data. The x-axis is in a logarithmic scale. Note a large performance gain in the cases of 5 and 10 fully labelled images.

|     | number of labelled images |      |      |      |      |       |       |       |
|-----|---------------------------|------|------|------|------|-------|-------|-------|
|     | 5                         | 10   | 20   | 40   | 100  | 500   | 1000  | 2249  |
| SSL | 38                        | 71   | 142  | 284  | 723  | 3612  | 7028  | 16004 |
| WSL | 6909                      | 7616 | 7925 | 8131 | 8578 | 11510 | 14109 |       |

**Table 6.1: Average number of labels used for each labelled dataset.** This table corresponds to Figure 6.8. SSL is for the strongly supervised case and WSL is for the weakly supervised case.

### 6.7.3 Player tracking

Once some hockey players are detected and given their team label, we ran a Kalman filter based tracker to take the detected bounding windows as input and to produce tracklets as output (Section 6.5). For evaluating the tracking results, we used several metrics from [Li et al., 2009]. Table 6.2 shows the name and definition of each evaluation metric.

Table 6.3 shows tracking results for four labelled datasets ( $\{5, 20, 100, 500\}$  images). For each labelled dataset, we present tracking results in two scenarios. One is based on the detected bounding windows of a model that is trained on labelled dataset (SSL). The other is based on a model that is trained on both labelled and unlabelled data with our self-learning system (WSL).

| Name         | Definition  |
|--------------|---|
| Precision(%) | No. of correctly matched objects / total No. of output objects.   |
| Recall(%)    | No. of correctly matched objects / total No. of ground-truth objects.   |
| FA           | No. of false alarms per frame. The smaller, the better.   |
| MT(%)        | Percentage of ground-truth trajectories which are correctly covered by tracker output for more than 80%                           |
| ML(%)        | Percentage of ground-truth trajectories which are correctly covered by tracker output for less than 20%                           |
| IDS          | Total No. of times that a tracked trajectory changes its matched ground-truth identity. The smaller, the better.                  |
| Wrong IDs(%) | $\frac{\text{total No. of objects that have a wrong ID}}{\text{total No. of all ground-truth objects}}$ . The smaller, the better |

**Table 6.2: Tracking evaluation metrics**

Several important points are observed from this table. Firstly, there is not much performance difference in most cases. But in the case of 5 labelled images, there are performance differences that indicate a substantial increase in the number of successfully tracked bounding windows. A large gain in recall and a small decrease in precision indicate a denser collection of tracking bounding windows. A large gain in MT and a large decrease in ML indicate that there are more targets that are correctly tracked for a longer period of time. A denser collection of tracked bounding windows resulted in an increase in FA, IDS and Wrong IDs. Figure 6.9 shows the result of these performance differences. In the figure, more hockey players are discovered and tracked successfully after four self-learning itera-

| method | Prec(%)               | Rec(%)                 | FA                     | MT(%)                  | ML(%)                  | IDS                | Wrong IDs(%)           |
|--------|-----------------------|------------------------|------------------------|------------------------|------------------------|--------------------|------------------------|
| SSL    | 89.5<br>( $\pm 8.2$ ) | 44.1<br>( $\pm 23.6$ ) | 0.21<br>( $\pm 0.02$ ) | 18.7<br>( $\pm 14.4$ ) | 43.9<br>( $\pm 23.9$ ) | 13<br>( $\pm 7$ )  | 18.9<br>( $\pm 11.4$ ) |
| WSL    | 89.1<br>( $\pm 6.3$ ) | 66.4<br>( $\pm 5.0$ )  | 0.66<br>( $\pm 0.50$ ) | 38.0<br>( $\pm 10.5$ ) | 27.0<br>( $\pm 0.9$ )  | 36<br>( $\pm 24$ ) | 32.0<br>( $\pm 4.1$ )  |

5 labelled images

|     |                       |                       |                        |                       |                       |                    |                       |
|-----|-----------------------|-----------------------|------------------------|-----------------------|-----------------------|--------------------|-----------------------|
| SSL | 91.5<br>( $\pm 1.7$ ) | 82.0<br>( $\pm 0.5$ ) | 0.56<br>( $\pm 0.13$ ) | 61.2<br>( $\pm 1.5$ ) | 15.4<br>( $\pm 0.8$ ) | 33<br>( $\pm 3$ )  | 39.6<br>( $\pm 1.5$ ) |
| WSL | 90.6<br>( $\pm 3.2$ ) | 82.2<br>( $\pm 2.6$ ) | 0.67<br>( $\pm 0.28$ ) | 64.3<br>( $\pm 4.9$ ) | 15.8<br>( $\pm 2.9$ ) | 35<br>( $\pm 12$ ) | 39.9<br>( $\pm 2.3$ ) |

20 labelled images

|     |                       |                       |                        |                       |                       |                   |                       |
|-----|-----------------------|-----------------------|------------------------|-----------------------|-----------------------|-------------------|-----------------------|
| SSL | 93.7<br>( $\pm 1.1$ ) | 83.8<br>( $\pm 0.5$ ) | 0.41<br>( $\pm 0.08$ ) | 63.6<br>( $\pm 1.5$ ) | 13.8<br>( $\pm 0.0$ ) | 27<br>( $\pm 4$ ) | 35.9<br>( $\pm 1.0$ ) |
| WSL | 93.0<br>( $\pm 0.8$ ) | 85.2<br>( $\pm 0.5$ ) | 0.47<br>( $\pm 0.07$ ) | 66.8<br>( $\pm 1.9$ ) | 13.1<br>( $\pm 0.5$ ) | 31<br>( $\pm 5$ ) | 37.5<br>( $\pm 1.7$ ) |

100 labelled images

|     |                       |                       |                        |                       |                       |                   |                       |
|-----|-----------------------|-----------------------|------------------------|-----------------------|-----------------------|-------------------|-----------------------|
| SSL | 94.5<br>( $\pm 0.9$ ) | 84.9<br>( $\pm 0.3$ ) | 0.35<br>( $\pm 0.07$ ) | 64.1<br>( $\pm 1.9$ ) | 13.5<br>( $\pm 0.5$ ) | 25<br>( $\pm 1$ ) | 38.8<br>( $\pm 1.1$ ) |
| WSL | 95.1<br>( $\pm 0.3$ ) | 85.1<br>( $\pm 0.2$ ) | 0.31<br>( $\pm 0.02$ ) | 68.7<br>( $\pm 1.6$ ) | 14.2<br>( $\pm 0.5$ ) | 22<br>( $\pm 1$ ) | 40.9<br>( $\pm 2.9$ ) |

500 labelled images

|     |      |      |       |      |     |    |      |
|-----|------|------|-------|------|-----|----|------|
| SSL | 92.8 | 87.1 | 0.504 | 73.9 | 9.8 | 25 | 41.8 |
|-----|------|------|-------|------|-----|----|------|

fully labelled images

**Table 6.3: Hockey tracking results.** This table shows tracking results in two scenarios: One is based on detection bounding windows of a model that is trained with labelled data (SSL), and the other is based on detection bounding windows of a model that is trained with our self-learning framework on both labelled and unlabelled data (WSL). Table 6.2 explains each evaluation metric.

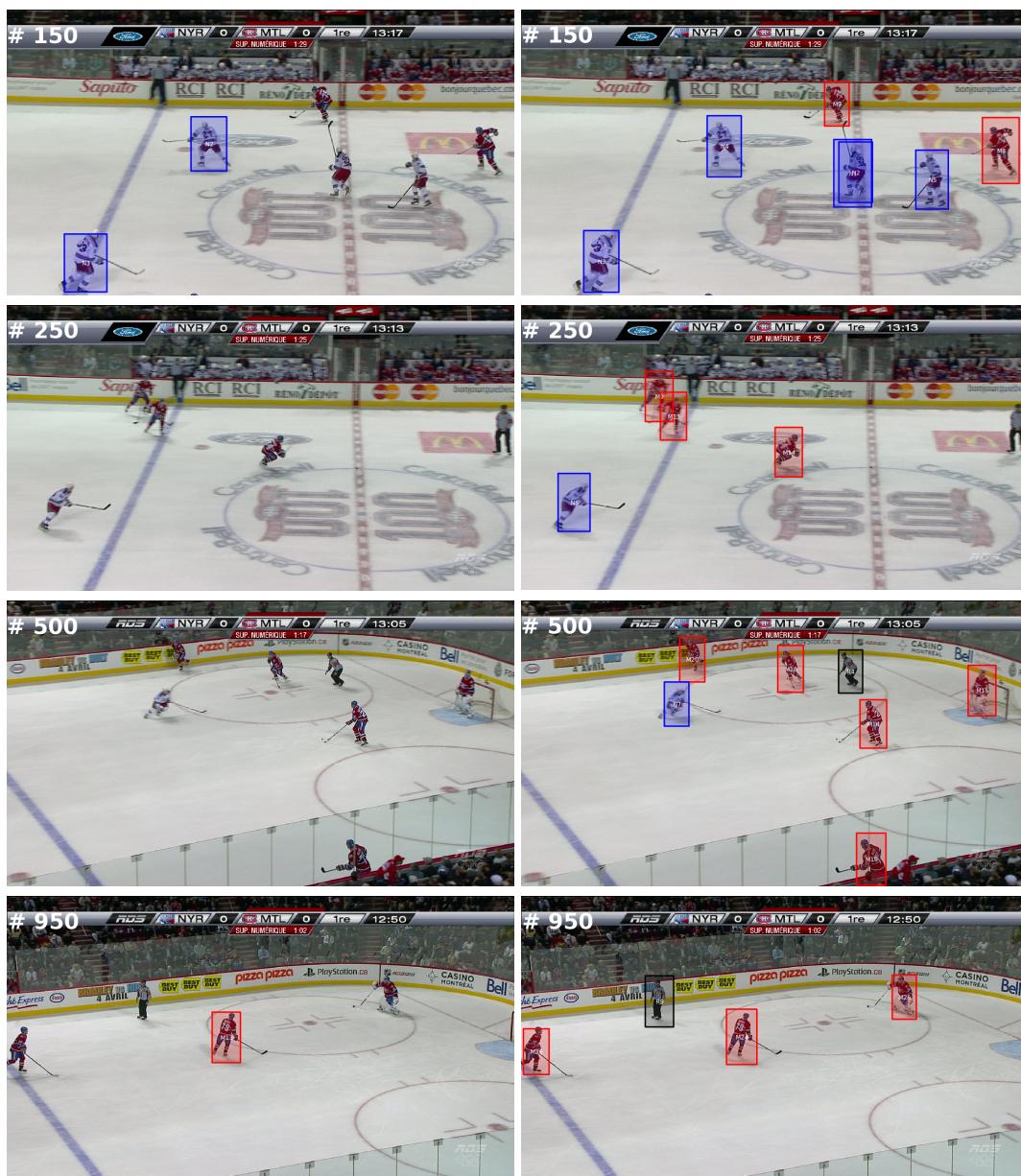
tions of our system in the case of 5 labelled images. Secondly, the performance of tracking hockey players quickly converges to the best performance in the case of fully labelled images (e.g., compare one in 500 labelled images and one in fully labelled images). This fast convergence is also evident in the detection result of Figure 6.8.

#### 6.7.4 Data selection

Thus far, we show that our self-learning approach improves the performance of detecting and tracking hockey players in smaller labelled datasets such as 5 and 10 labelled images. Figure 6.10 shows representative candidate bounding windows in each iteration of the self-learning process. The figure shows the most confident bounding windows with a high detection score and the least confident bounding windows with a low detection score among candidate bounding windows that are selected by our data selection algorithm Algorithm 6.2. The localization of hockey players is improved slowly in each iteration. The difference is especially obvious between the iteration 1 and 4, where there is an improvement of 12% in the average precision.

#### 6.7.5 Computational time

Our experiments were performed on a 8-core (Intel Xeon 2.66GHz) machine with 32GB of RAM. Figure 6.11 shows the average total number of CPU hours to complete our experiments for both the strongly supervised and weakly supervised cases. The weakly supervised case had four additional learning iterations on top of the strongly supervised case which required only one iteration for training and testing. It took about a week of CPU time to run our system on all labelled datasets, where over 80% of time was spent for training a detector and running it on both training and test images to obtain detection bounding windows. It takes about 7 to 10 seconds to run our DPM detector on an image of  $960 \times 540$ . For speeding up the detection process, the size prior of hockey players was estimated from training data and used to focus computational resources within a limited range of scales — in our case,  $[\mu_s - \sigma_s, \mu_s + \sigma_s]$  where  $\mu_s$  is the mean size and  $\sigma_s$  is a standard deviation. When there is a large amount of labelled images (e.g., 500 and 1,000 labelled images), it took over 24 CPU hours on a 8-core machine without much performance improvement in detecting and tracking hockey players.

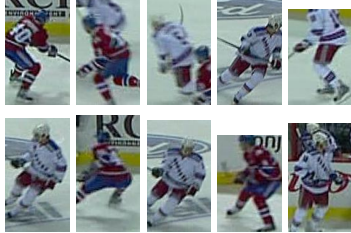
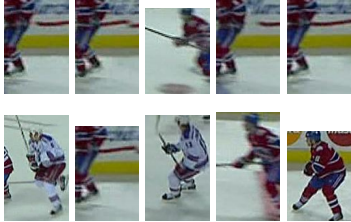
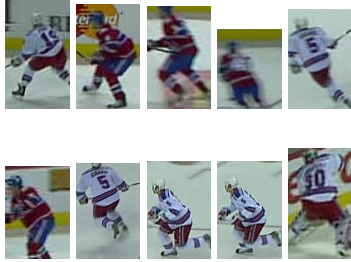







(a) Prior to self-learning

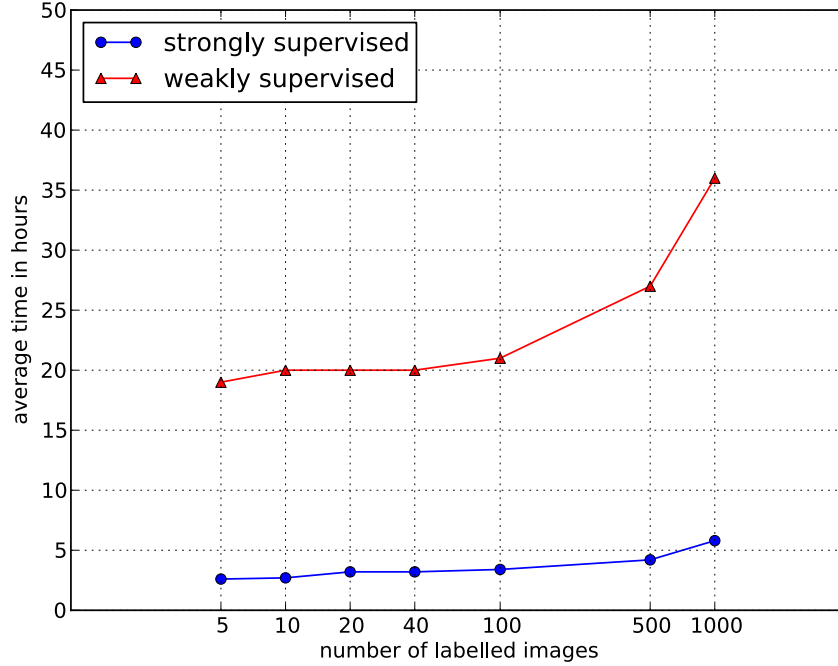
(b) After self-learning

**Figure 6.9: Screenshots of our tracking result in hockey videos.** This shows our tracking results on the test data. Column (a) uses detection inputs of a detector that is trained with 5 labelled images (SSL). Column (b) uses a detector that is trained with both labelled and unlabelled data (WSL). Note that more players are discovered and tracked successfully after four self-learning iterations. Frame numbers are shown in the upper left corner of each image.



|                 | most confident  | least confident  |
|-----------------|---|--|
| Iter 1<br>(.55) |    |    |
| Iter 2<br>(.62) |    |    |
| Iter 3<br>(.64) |   |   |
| Iter 4<br>(.67) |  |  |

**Figure 6.10: Most confident and least confident candidate bounding windows in hockey videos.** This shows the most confident (i.e., highest scoring detection) and the least confident (i.e., lowest scoring detection) candidate bounding windows that are selected from unlabelled images in the training data by Algorithm 6.2. The average precision of our detection model on the test data for each iteration is shown in the parentheses. Note how the localization of hockey players is improved from the iteration 1 to 4.



**Figure 6.11: Total train and test time in hockey data.** This shows the total computational time for training and testing our system on hockey videos. For the weakly supervised case, there are four additional learning iterations, which costs an additional number of hours on top of the strongly supervised case which requires only one iteration for training and testing. The x-axis is in a logarithmic scale.

## 6.8 Application to other sports video

### 6.8.1 Basketball

We also tested our system on basketball data, which consists of 7 different video sequences which sum to 4,818 image frames of a broadcast footage. The data are split into two separate sets: 3 sequences (2,486 frames in total) for training and 4 sequences (2,332 frames in total) for testing. The annotations are given in the same way as our hockey data. The only difference is that there are only two team labels, {"Celtics", "Lakers"} (i.e., no "referee" label). Following our experimental settings in hockey data, we also prepared 8 different sets of fully labelled images: 7 sets of  $m$  randomly selected fully labelled images where

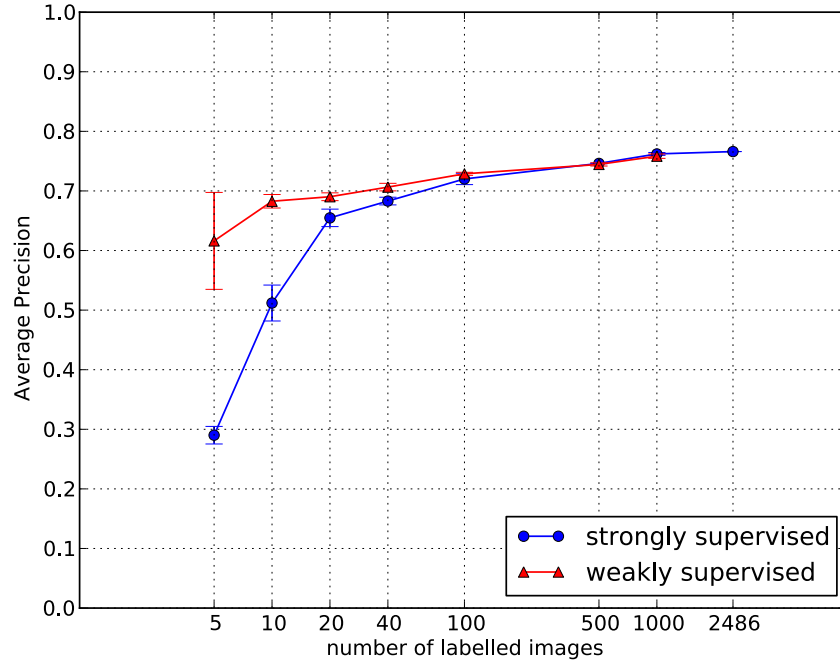
$m = \{5, 10, 20, 40, 100, 500, 1000\}$  and the fully supervised set of all 2,486 images. For each initial labelled dataset, we first trained the initial shape-based DPM detector and part-based colour classifiers. Then we applied our self-learning framework to collect additional training labels from the unlabelled data and improve initial classifiers iteratively up to four iterations. We ran the system three times on the basketball data.

Overall, our experimental results were similar between hockey and basketball data. Here, we highlight some important points:

- **Detection:** Figure 6.12 shows an improvement in the performance of detection when there is a smaller amount of labelled dataset such as 5 and 10 labelled images. The improvement is slightly better in basketball data because there is a clear separation of the performance gap between the strongly supervised case and weakly supervised case.
- **Tracking:** Table 6.5 shows the similar trend as Table 6.3. There is an increase in the number of tracking bounding windows in the case of 5 labelled images as shown in Figure 6.13. The tracking performance quickly converges with 500 labelled images.
- **Data selection:** Figure 6.14 shows an interesting result. The figure does not clearly show an improvement in localization of basketball players, but the performance of our detector in the average precision was improved in every iteration. There seems to be a small improvement in localization of the least confident bounding windows.
- **Computational time:** Experiments in basketball data took longer than hockey data. Figure 6.15 shows the average total time of CPU hours each run took. Since the average size of basketball players is bigger than that of hockey players, detection took longer in basketball data because there are more sliding bounding windows for a model to evaluate.

## 6.9 Conclusion

In this chapter, we empirically show that our approach is particularly effective when there is a very small amount of labelled data. In our experiments, our approach exploits both labelled and unlabelled data in sparsely labelled videos of sports games and provided a mean



**Figure 6.12: Detection result of our weakly self-learning system in basketball videos.** The blue line shows the baseline performance based on only labelled datasets. The red line shows the performance after four self-learning iterations of collecting additional labels from unlabelled data. The x-axis is in a logarithmic scale. Note a large performance gain in the cases of 5 and 10 fully labelled images.

|     | number of labelled images |      |      |      |      |       |       |       |
|-----|---------------------------|------|------|------|------|-------|-------|-------|
|     | 5                         | 10   | 20   | 40   | 100  | 500   | 1000  | 2486  |
| SSL | 46                        | 92   | 190  | 377  | 952  | 4687  | 9345  | 22958 |
| WSL | 7438                      | 7793 | 8021 | 8232 | 8791 | 12606 | 17083 |       |

**Table 6.4: Average number of labels used for each labelled dataset.** This table corresponds to Figure 6.12. SSL is for the strongly supervised case and WSL is for the weakly supervised case.

| method | Prec(%)               | Rec(%)                | FA                     | MT(%)                 | ML(%)                  | IDS                | Wrong IDs(%)          |
|--------|-----------------------|-----------------------|------------------------|-----------------------|------------------------|--------------------|-----------------------|
| SSL    | 95.3<br>( $\pm 1.5$ ) | 32.3<br>( $\pm 6.4$ ) | 0.15<br>( $\pm 0.06$ ) | 7.5<br>( $\pm 2.0$ )  | 35.8<br>( $\pm 15.0$ ) | 11<br>( $\pm 4$ )  | 12.0<br>( $\pm 4.4$ ) |
| WSL    | 92.2<br>( $\pm 2.9$ ) | 68.8<br>( $\pm 3.7$ ) | 0.56<br>( $\pm 0.27$ ) | 25.8<br>( $\pm 1.2$ ) | 2.5<br>( $\pm 2.0$ )   | 50<br>( $\pm 15$ ) | 39.3<br>( $\pm 6.0$ ) |

5 labelled images

|     |                       |                       |                        |                       |                      |                   |                       |
|-----|-----------------------|-----------------------|------------------------|-----------------------|----------------------|-------------------|-----------------------|
| SSL | 94.6<br>( $\pm 0.6$ ) | 71.4<br>( $\pm 1.3$ ) | 0.38<br>( $\pm 0.04$ ) | 28.3<br>( $\pm 1.2$ ) | 0.8<br>( $\pm 1.2$ ) | 39<br>( $\pm 3$ ) | 37.2<br>( $\pm 2.0$ ) |
| WSL | 93.6<br>( $\pm 0.7$ ) | 77.6<br>( $\pm 0.8$ ) | 0.50<br>( $\pm 0.07$ ) | 41.7<br>( $\pm 4.7$ ) | 0.0<br>( $\pm 0.0$ ) | 47<br>( $\pm 6$ ) | 38.8<br>( $\pm 2.5$ ) |

20 labelled images

|     |                       |                       |                        |                       |                      |                   |                       |
|-----|-----------------------|-----------------------|------------------------|-----------------------|----------------------|-------------------|-----------------------|
| SSL | 95.6<br>( $\pm 0.8$ ) | 75.2<br>( $\pm 0.8$ ) | 0.33<br>( $\pm 0.06$ ) | 34.2<br>( $\pm 5.1$ ) | 0.8<br>( $\pm 1.2$ ) | 36<br>( $\pm 6$ ) | 39.2<br>( $\pm 1.5$ ) |
| WSL | 96.1<br>( $\pm 0.1$ ) | 75.8<br>( $\pm 1.0$ ) | 0.29<br>( $\pm 0.01$ ) | 38.3<br>( $\pm 3.1$ ) | 0.0<br>( $\pm 0.0$ ) | 39<br>( $\pm 2$ ) | 39.4<br>( $\pm 3.0$ ) |

100 labelled images

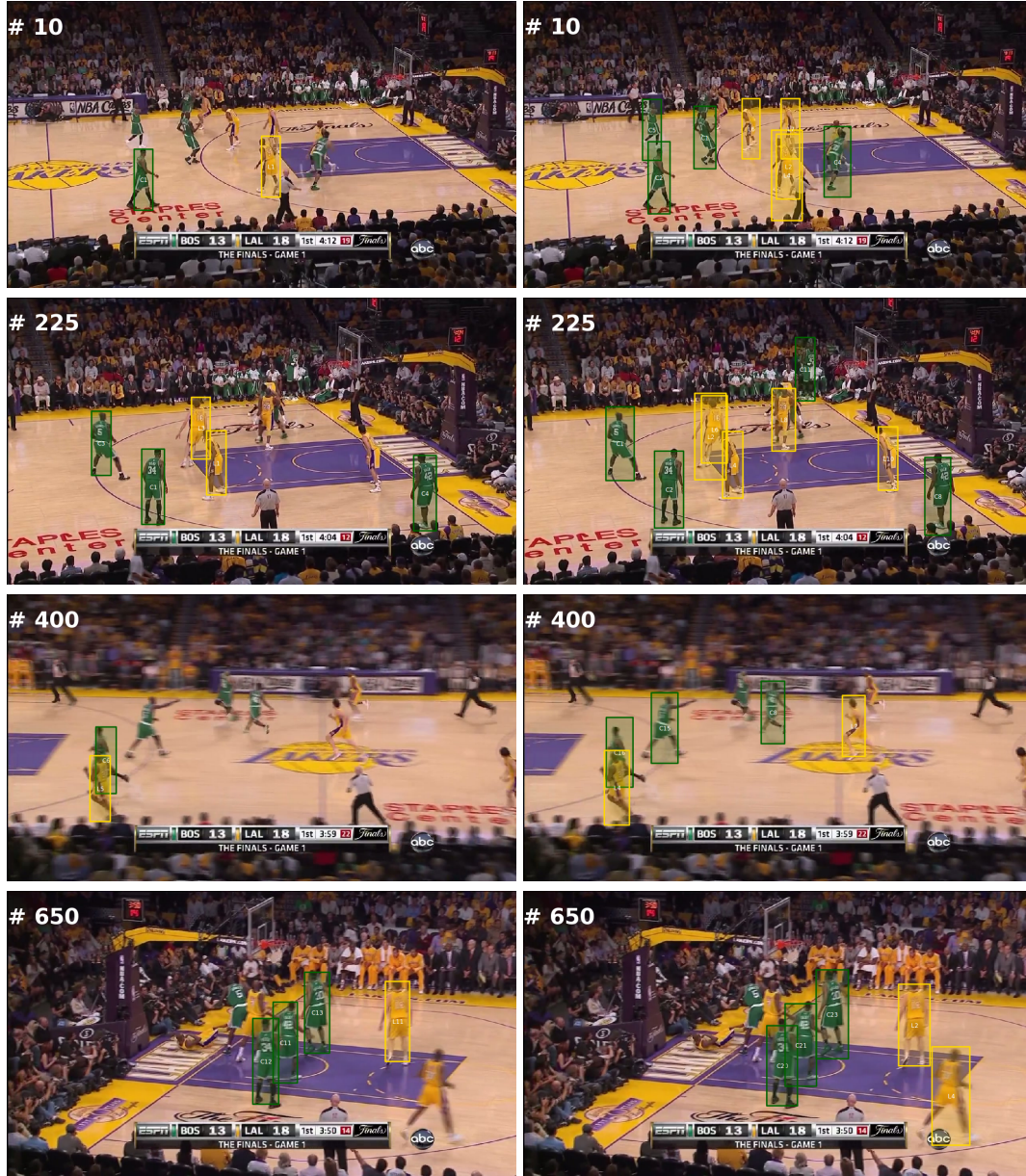
|     |                       |                       |                        |                       |                      |                   |                       |
|-----|-----------------------|-----------------------|------------------------|-----------------------|----------------------|-------------------|-----------------------|
| SSL | 96.1<br>( $\pm 1.1$ ) | 77.1<br>( $\pm 0.4$ ) | 0.30<br>( $\pm 0.09$ ) | 41.7<br>( $\pm 3.1$ ) | 0.0<br>( $\pm 0.0$ ) | 39<br>( $\pm 6$ ) | 40.4<br>( $\pm 2.5$ ) |
| WSL | 96.6<br>( $\pm 0.5$ ) | 78.4<br>( $\pm 0.5$ ) | 0.26<br>( $\pm 0.04$ ) | 44.2<br>( $\pm 4.2$ ) | 0.0<br>( $\pm 0.0$ ) | 35<br>( $\pm 3$ ) | 42.0<br>( $\pm 1.0$ ) |

500 labelled images

|     |      |      |       |      |     |    |      |
|-----|------|------|-------|------|-----|----|------|
| SSL | 96.1 | 80.2 | 0.306 | 47.5 | 0.0 | 34 | 37.4 |
|-----|------|------|-------|------|-----|----|------|

fully labelled images

**Table 6.5: Basketball tracking results** This table shows tracking results for two cases: One is based on detection bounding windows of a model that is trained with labelled data (SSL), and the other is based on detection bounding windows of a model that is trained with our self-learning framework on both labelled and unlabelled data (WSL). Table 6.2 explains each evaluation metric.

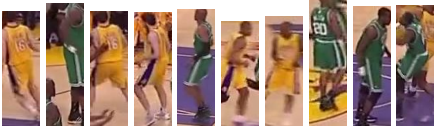
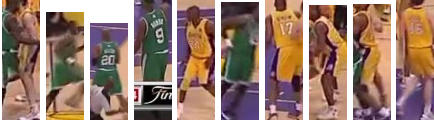




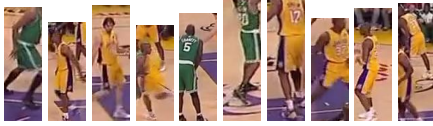


(a) Prior to self-learning

(b) After self-learning

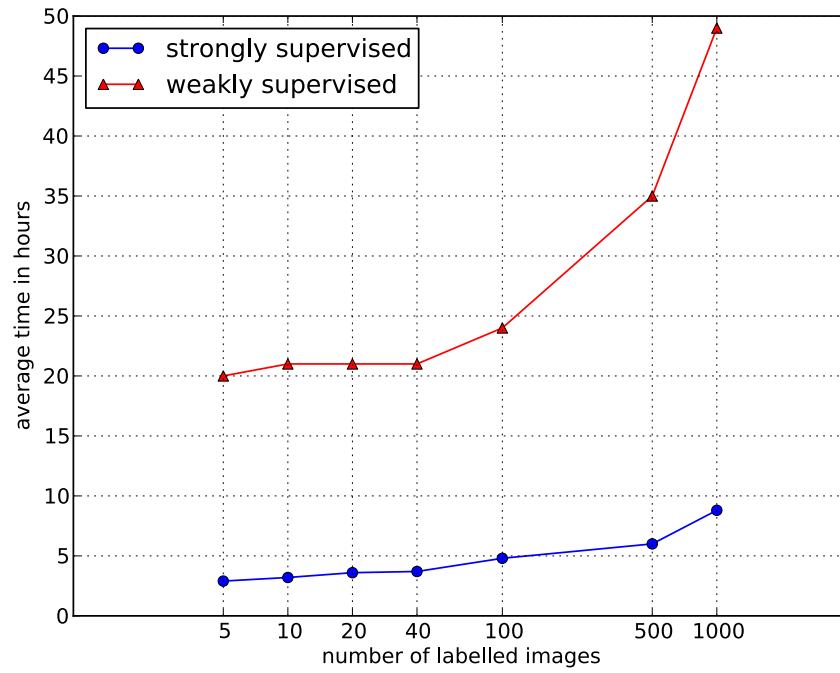
**Figure 6.13: Screenshots of our tracking result in basketball videos.** This shows our tracking results on the test data. Column (a) uses detection inputs of a detector that is trained with 5 labelled images (SSL). Column (b) uses a detector that is trained with both labelled and unlabelled data (WSL). Note that more players are discovered and tracked successfully after four self-learning iterations. Frame numbers are shown in the upper left corner of each image.



|                 | most confident  | least confident  |
|-----------------|---|--|
| Iter 1<br>(.58) |  |  |
| Iter 2<br>(.60) |  |  |
| Iter 3<br>(.66) |  |  |
| Iter 4<br>(.67) |  |  |

**Figure 6.14: Most confident and least confident candidate bounding windows in basketball videos.** This shows the most confident (i.e., highest scoring detection) and the least confident (i.e., lowest scoring detection) candidate bounding windows that are selected from unlabelled images in the training data by Algorithm 6.2. The average precision of our detection model on the test data for each iteration is shown in the parentheses. Unlike hockey data, there is no obvious improvement in localization. However, the performance of our detector in the average precision improved in each iteration.

performance improvement of over 20% in the average precision for detecting sports players when videos contained very few labelled images. We adopt self-learning with a novel selection criterion that combines several image cues such as the appearance information (i.e., shape and colour) of players, the constraints on their motions, and the colour of the playing field for discovering additional labels automatically from unlabelled data.



**Figure 6.15: Total train and test time in basketball data.** This shows the total computational time for training and testing our system on basketball videos. For the weakly supervised case, there are four additional learning iterations, which costs an additional number of hours on top of the strongly supervised case which requires only one iteration for training and testing. The x-axis is in a logarithmic scale.



## Chapter 7

# Conclusions

This thesis has explored how to reduce the human effort in labelling images for the training of object detectors. Statistical models based on the appearance of object classes such as faces, pedestrians, and cars have been widely applied in the real world. However, these statistical models have a major drawback in that they require a large amount of training data. One way to resolve this issue is to make the process of label acquisition more efficient by reducing costly human labelling efforts. To achieve this, we provide two solutions in the thesis.

In Chapters 3 and 4, we show that our active learning approach, which uses a novel interface to combine machine intelligence with human interventions, effectively improves a state-of-the-art classifier by using additional unlabelled data from the Web. This novel approach combines active learning with an adaptive interface to efficiently collect labels for improving the generalized performance of classifiers. We are also the first to test an active learning approach on the PASCAL 2007 dataset. Our experimental results show that our approach improves a state-of-the-art classifier of [Felzenszwalb et al., 2009] by actively selecting its own training data from a small portion (10%) of the image data pool, which consists of over 60,000 images downloaded from Flickr image search.

As the approach relies on a small amount of input from a human oracle to improve its performance, there is still room to further reduce the amount of human labelling effort. An ideal solution is to have no humans involved in labelling novel data. Chapter 6 introduces a novel self-learning framework that automates the label acquisition process for improving models for detecting players in broadcast footage of sports games. Unlike most previ-

ous self-learning approaches of improving appearance-based object detectors, we allow an unknown, unconstrained number of target objects any time in a video sequence. Our self-learning approach combines several image cues such as the appearance information (i.e., shape and colour) of players, the constraints on their motions, and the colour of the playing field for discovering additional labels automatically from unlabelled data. Our experimental results show that our approach is particularly effective when there is very little labelled data.

## 7.1 Toward the future of crowdsourcing

With abundant unlabelled data, crowdsourcing is a powerful tool to utilize extensive human labour efficiently without much cost, but it is by no means without any drawback. Quality control of crowdsourced annotations has been very difficult because a variety of anonymous workers set different levels of annotation quality. An additional infrastructure is often required to organize these annotations, which also costs human labour.

To surmount this difficulty, most of the current crowdsourcing approaches focus on how to reduce the overall cost of time and money for obtaining labels from unlabelled data without losing the quality. LabelMe [Russell et al., 2008] and other interactive user interfaces on Amazon Mechanical Turk such as the one by [Sorokin and Forsyth, 2008] and the Visipedia project [Welinder and Perona, 2010] are the front line of the work that addresses inexpensive acquisition of labels from a large pool of thousands of unlabelled images. In recent years, however, ImageNet [Deng et al., 2009] has demonstrated extremely large scale high-quality labelling<sup>1</sup> and introduced a large-scale dataset that utilizes crowdsourcing labour of Amazon Mechanical Turk for collecting over 12 million images with ground-truth annotations.

Crowdsourcing has also been utilized for annotating a collection of video data. Interactive annotation tools on the Web such as VATIC, a video annotation tool<sup>2</sup> by [Vondrick and Ramanan, 2011; Vondrick et al., 2010], and LabelMe video<sup>3</sup> [Yuen et al., 2009] have become publicly available in the computer vision community to foster large scale labelling of unlabelled video data.

---

<sup>1</sup>The ImageNet has been created through the crowdsourcing platform of CrowdFlower <http://crowdflower.com/>, CrowdFlower is the world's largest enterprise for crowdsourcing platform business and provides access to 1.5 million contributors worldwide.

<sup>2</sup>The source code and software demo are available from <http://mit.edu/vondrick/vatic/>

<sup>3</sup>The source code and software demo are available from <http://labelme.csail.mit.edu/VideoLabelMe/>

However, those crowdsourcing tools consider neither the impact of each label for improved performance of a classification model nor reducing the amount of unlabelled data. Our proposed approaches, including our adaptive interface with active learning in Chapters 3 and 4 and self-learning framework in sports video in Chapter 6, address how to reduce the cost of human labour for labelling data by reducing the overall amount of unlabelled data as well as maximizing the impact of each label for improving the performance of the object detection model.

Our adaptive interface with active learning demonstrated that the current annotation standard of having only “YES” or “NO” answer is not as effective as allowing a user to interact with the current classification model and direct queries by indicating where the query should be in the form of bounding windows. We also showed that the quality of the label data is important in order to maximize the performance improvement of a model. Active learning with uncertainty sampling criterion is quite effective in eliminating unlabelled data that are not effective for improved performance of a model, as we selected only the most useful 10% (6,000 images) of 60,000 unlabelled images from the Web. Current crowdsourcing approaches may waste their resource on a large amount of labelled data that do not give a further improvement of a model.

Our self-learning system realizes fully automatic acquisition of labels if a small amount of label data is available. Ideally, the label acquisition process should be fully automatic, which will be a difficult or impossible goal to achieve in general. Although we showed such possibility in sports video, there are still many challenges that need to be resolved in order to realize fully automatic acquisition of labels for solving the problem of generic object detection.

In the near future, crowdsourcing will become a new paradigm of large scale study in object detection. Vijayanarasimhan and Grauman [2011] have already attempted to train an object detector with crowdsourced annotations from the Web and showed promising results. It is very likely that more work will follow them and use crowdsourcing to obtain labelled data for training statistical models. With the lessons we learned from this thesis, we now believe that the future of crowdsourcing would be to combine human labour, machine intelligence, and crowdsourcing all together. Then we would be able to not only reduce the size of redundant unlabelled data, but also to maximize the impact of each label quite effectively in massively parallel platforms of inexpensive human labour.

## 7.2 Tradeoff between complexity and scalability

A tradeoff between complexity and scalability of models has become a major issue in designing a machine vision system for recognizing generic objects, since researchers in academia now have access to a large-scale database such as the ImageNet and the LabelMe. This poses an interesting question — an overwhelming amount of data with simple algorithms may be enough to solve problems without even using more sophisticated, complex algorithms. Recently in linguistics, speech recognition and machine translation systems based on n-gram language models outperformed systems based on grammars and phrase structure. In computer vision, Torralba et al. [2007] have shown that, by using a large-scale dataset of 79 million tiny ( $32 \times 32$ ) image patches, simple non-parametric methods such as nearest-neighbour algorithms can give reasonable performance on object recognition tasks based on Euclidian distance of intensity. The work became a milestone in introducing the power of large data with scalable simple models. When we enrich models for improving their generalized performance, we need to make careful design decisions so that models are still scalable with a large amount of data. In practical real-world applications, complex models are often not preferred because they are expensive in learning and inference, and are not scalable with a large amount of data.

## 7.3 Future directions

We conclude by introducing some avenues for further exploration:

- **An adaptive interface with crowdsourcing:** An immediate extension to our adaptive interface (Chapter 4) is to use the interface on the crowdsourcing platform of Amazon Mechanical Turk. Unlike the conventional interface that only provides a question to a worker, our interface allows real-time interactions between a worker and a machine — letting a worker correct or add queries based on the current classification function. As shown in our experiments from Chapter 4, queries that correct misclassifications of the current classifier have a much larger impact on the final classification performance. It is interesting to see how crowdsourcing workers would interact with the current classification model and how much impact our interface would have for improving the performance of the model.

- **More self-learning experiments with larger data:** In Chapter 6, we used a set of 7 video sequences which sum to 4,627 image frames of hockey data and another set of 7 video sequences which sum to 4,818 image frames of basketball data. An obvious next step is to scale up these experiments. We manually annotated a much larger dataset. For hockey, we have annotation data for the entire first period which contains 36 sequences of short video clips that sum to 35,317 image frames (258,435 bounding boxes) in total. For basketball, we have the entire first quarter that contains 30 sequences of short video clips that sum to 20,579 image frames (184,821 bounding boxes) in total. It will be interesting to see how a much larger pool of unlabelled data influences the degree of performance improvement in our self-learning iterations.
- **Exploiting more negative data:** Our self-learning framework in Chapter 6 currently does not collect any additional negative instances. The presence of multiple target objects and audience surrounding the playing field makes it difficult to extract regions of negative instances (i.e., non-player objects). However, it is possible to collect negative data by filtering out some of detection bounding windows that do not belong to trajectories of target objects.
- **Exploring more data in sports:** Given a small set of labelled data, our self-learning framework is capable of training and improving a statistical model for detecting sports players in broadcast footage. However, our model works well only for those data that have a similar distribution as the training data. Even in the same sport, there are data that are different in view points, the resolution of the image, and teams that play in a game. Figure 7.1 shows screen shots of various video data. It is time-consuming and not desirable to obtain labels every time we have new data in a different game. There are two straightforward ways to apply our self-learning approach for reducing labelling effort. One is to manually obtain a very small amount of labels from new data and to apply our approach to it. The other is to use models that are already trained with other data for a similar class of objects. For our experiments in Chapter 6, we used data from Figures 7.1(a) and 7.1(d). We can use these models as the initial model of our self-learning framework to explore more data such as Figures 7.1(b), 7.1(c) and 7.1(e) where our initial models may initially have poor performance. We could use a small amount of labelled data if the data contain different teams with a different colour jersey, we cannot use previously trained colour models.



(a) High Definition (HD) broadcast ( $960 \times 540$ )



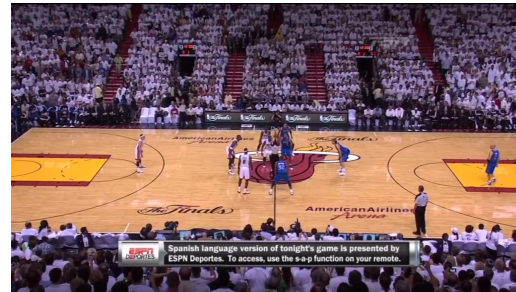
(b) NTSC broadcast  
( $320 \times 240$ )



(c) HD hand-held digital camera ( $852 \times 480$ )



(d) HD broadcast ( $960 \times 540$ )



(e) HD broadcast ( $960 \times 540$ )

**Figure 7.1: Variability of data in broadcast footage of sports games.** This figure presents various examples from hockey and basketball games. Each screen shot contains different teams in a different game, and has a different resolution of an image. The caption specifies the resolution of an image in parentheses.

- **Application to other domains of data:** Last but not least, our self-learning approach is not limited to sports data. For instance, our self-learning framework can improve models for detecting vehicles or pedestrians without much labelling effort. Computer-based visual recognition systems have been applied for detecting near-by obstacles such as other vehicles and pedestrians around a vehicle when other sensors are not as effective. Smart cars have a built-in pedestrian detection system in order to warn drivers when there are people on the road while driving, and when there are pedestrians on the sidewalks while parking. Our approach can be applied to training these models effectively with sparsely labelled videos.

# Bibliography

- Abramson, Y. and Freundndom, Y. (2005). Semi-automatic visual learning (Seville): A tutorial on active learning for visual object recognition. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition, Workshop*.
- Alexe, B., Deselaers, T., and Ferrari, V. (2010). What is an object ? In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Ali, K., Hasler, D., and Flueret, F. (2011). Flowboost - appearance learning from sparsely annotated video. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Arora, H., Loeff, N., Forsyth, D. A., and Ahuja, N. (2007). Unsupervised segmentation of objects using efficient learning. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Babenko, B., Yang, M.-H., and Belongie, S. (2009). Visual tracking with online multiple instance learning. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Ballard, D. H. (1981). Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122.
- Belongie, S., Malik, J., and Puzich, J. (2001). Matching shape. In *IEEE International Conference on Computer Vision*.
- Berg, T. L. ., Sorokin, A., Wang, G., Forsyth, D. A., Endres, I., and Farhadi, A. (2010). Its all about the data. *Proceedings of the IEEE*, 98:1434 – 1452.
- Bourdev, L., Maji, S., Brox, T., and Malik, J. (2010). Detecting people using mutually consistent poselet activations. In *The 11th European Conference on Computer Vision*.
- Chapelle, O., Schölkopf, B., and Zien, A., editors (2006). *Semi-Supervised Learning*. The MIT Press.



- Chum, O. and Zisserman, A. (2007). An exemplar model for learning object classes. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Cohn, D., Ghahramani, Z., and Jordan, M. I. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145.
- Collins, B., Deng, J., Li, K., and Fei-Fei, L. (2008). Towards scalable dataset construction: An active learning approach. In *The 10th European Conference on Computer Vision*.
- Comaniciu, D. and Meer, P. (2002). Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619.
- Crandall, D. J. and Huttenlocher, D. P. (2006). Weakly supervised learning of part-based spatial models for visual object recognition. In Aleš Leonardis, H. B. and Pinz, A., editors, *The 9th European Conference on Computer Vision*, volume 3951 of *LNCS*, pages 16–29. Springer.
- Dalal, N. (2006). *Finding People in Images and Videos*. PhD thesis, INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE.
- Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*, pages 886–893.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Desai, C., Ramanan, D., and Fowlkes, C. (2009). Discriminative models for multi-class object layout. In *IEEE International Conference on Computer Vision*.
- Deselaer, T., Alex, B., and Ferrari, V. (2010). Localizing objects while learning their appearance. In *The 11th European Conference on Computer Vision*.
- Deselaer, T. and Ferrari, V. (2010). Global and efficient self-similarity for object classification and detection. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Dollár, P., Wojek, C., Schiele, B., and Perona, P. (2009). Pedestrian detection: A benchmark. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*, Miami, Florida.

- Dorkó, G. and Schmid, C. (2003). Selection of scale-invariant parts for object class recognition. In *IEEE International Conference on Computer Vision*.
- Doucet, A., de Freitas, N., and Gordon, N. (2001). *Sequential Monte Carlo Methods in Practice*. Springer-Verlag New York, Inc.
- Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2).
- Everingham, M., van Gool, L., Williams, C., Winn, J., and Zisserman, A. (2007). The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>.
- Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2009). Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Felzenszwalb, P. F. and Huttenlocher, D. P. (2005). Pictorial structures for object recognition. *International Journal of Computer Vision*, 61:55–79.
- Fergus, R., Perona, P., and Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Fergus, R., Perona, P., and Zisserman, A. (2007). Weakly supervised scale-invariant learning of models for visual recognition. *International Journal of Computer Vision*, 71.
- Fischler, M. A. and Elschlager, R. A. (1973). The representation and matching of pictorial structures. *IEEE Trans. on Computer*, 22(1):67 – 92.
- Galleguillos, C., Babenko, B., Rabinovich, A., and Belongie, S. (2008). Weakly supervised object localization with stable segmentations. In *The 10th European Conference on Computer Vision*.
- Grauman, K. and Darrell, T. (2007). The pyramid match kernel: Efficient learning with sets of features. *Journal of Machine Learning Research*, 8:725–760.
- Guestrin, C., Krause, A., and Singh, A. P. (2005). Near-optimal sensor placements in gaussian processes. In *The 22nd International Conference on Machine Learning*.
- Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey Vision Conference*.

- Hewitt, R. and Belongie, S. (2006). Active learning in face recognition: Using tracking to build a face model. In *IEEE CVPR Workshop on Vision for Human Computer Interaction (V4HCI)*, pages 157–157.
- Jurie, F. and Schmid, C. (2004). Scale-invariant shape features for recognition of object categories. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Kadir, T. and Brady, M. (2001). Saliency, scale and image description. *International Journal of Computer Vision*, 45(2):83 – 105.
- Kalal, Z., Matas, J., and Mikolajczyk, K. (2010). P-n learning: Bootstrapping binary classifiers by structural constraints. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME Journal of Basic Engineering*, 82:35 – 45.
- Kapoor, A., Grauman, K., Urtasuna, R., and Darrell, T. (2007). Active learning with Gaussian Processes for Object Categorization. In *IEEE International Conference on Computer Vision*, pages 1–8.
- Khan, S. M. and Shah, M. (2006). A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *The 9th European Conference on Computer Vision*.
- Lampert, C. H., Blaschko, M. B., and Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Lawrence, N. D. and Jordan, M. I. (2005). Semi-supervised learning via gaussian processes. In *Advances in Neural Information Processing Systems*.
- Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*, pages 2169–2178.
- Lee, Y. J. and Grauman, K. (2011). Learning the easy things first: Self-paced visual category discovery. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Lehmann, A., Leibe, B., and van Gool, L. (2009). Feature-centric efficient subwindow search. In *IEEE International Conference on Computer Vision*.

- Leibe, B., Leonardis, A., and Schiele, B. (2007). Robust object detection by interleaving categorization and segmentation. *International Journal of Computer Vision*, 77(1-3):259–289.
- Leistner, C., Godec, M., Schuster, S., Saffari, A., Werlberger, M., and Bischof, H. (2011). Improving classifiers with unlabeled weakly-related videos. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Leistner, C., Grabner, H., and Bischof, H. (2007). Semi-supervised boosting using visual similarity learning. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Lepetit, V. and Fua, P. (2006). Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479.
- Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proc. of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*.
- Li, Y., Huang, C., and Nevatia, R. (2009). Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Lin, H.-T., Lin, C.-J., and Weng, R. C. (2003). A note on platts probabilistic outputs for support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University.
- Lindeberg, T. (1998). Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79 – 116.
- Ling, H. and Okada, K. (2006). Emd- $l_1$ : An efficient and robust algorithm for comparing histogram-based descriptors. In Aleš Leonardis, H. B. and Pinz, A., editors, *The 9th European Conference on Computer Vision*, volume 3953 of *LNCS*, pages 330–343. Springer.
- Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision*.
- Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.
- Lu, W.-L., Okuma, K., and Little, J. J. (2009). Tracking and recognizing actions of multiple hockey players using the boosted particle filter. *Image and Vision Computing*, 27(1-2):189–205.

- Lu, W.-L., Ting, J.-A., Murphy, K. P., and Little, J. J. (2011). Identifying players in broadcast sports videos using conditional random fields. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Mahdaviani, M., de Freitas, N., Fraser, B., and Hamze, F. (2005). Fast computational methods for visually guided robots. In *IEEE International Conference on Robotics and Automation*.
- Maji, S., Berg, A. C., and Malik, J. (2008). Classification using intersection kernel support vector machines is efficient. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Mallapragada, P. K., Jin, R., Jain, A. K., and Liu, Y. (2007). Semiboost: Boosting for semi-supervised learning. Technical report, Department of Computer Science and Engineering, Michigan State University.
- Mallat, S. G. (1989). A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674 – 693.
- Markoff, J. (2010). Google cars drive themselves, in traffic. New York Times.
- Martin, D. R., Fowlkes, C. C., and Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530 – 549.
- Merwe, R., Doucet, A., Freitas, N., and Wan, E. (2000). The unscented particle filter. Technical report *cued/f-infeng/tr 380*, Cambridge University Engineering Department, 20 Ames St. Cambridge, MA 02139.
- Mikolajczyk, K. and Schmid, C. (2002). An affine invariant interest point detector. In *The 7th European Conference on Computer Vision*.
- Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630.
- Moosmann, F., Larlus, D., and Jurie, F. (2006). Learning saliency maps for object categorization. In *The 9th European Conference on Computer Vision*.
- Mori, G. and Malik, J. (2003). Recognizing objects in adversarial clutter: Breaking a visual captcha. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.

- Murphy, K. P., Torralba, A. B., and Freeman, W. T. (2003). Using the forest to see the trees: A graphical model relating features, objects, and scenes. In *Advances in Neural Information Processing Systems*.
- Mutch, J. and Lowe, D. G. (2006). Multiclass object recognition with sparse, localized features. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Nguyen, M. H., Torresani, L., de la Torre, F., and Rother, C. (2009). Weakly supervised discriminative localization and classification: a joint learning process. In *IEEE International Conference on Computer Vision*.
- Nigam, K., McCallum, A. K., and Tom Mitchell, S. T. (2000). Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2/3):103 – 134.
- Okuma, K., Brochu, E., Lowe, D. G., and Little, J. J. (2011). An adaptive interface for active localization. In *International Conference on Computer Vision Theory and Applications*.
- Okuma, K., Taleghani, A., de Freitas, N., Little, J. J., and Lowe, D. G. (2004). A boosted particle filter: Multitarget detection and tracking. In *The 8th European Conference on Computer Vision*, pages 28–39.
- Opelt, A., Pinz, A., Fussenegger, M., and Auer, P. (2006a). Generic object recognition with boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):416–431.
- Opelt, A., Pinz, A., and Zisserman, A. (2006b). A boundary-fragment-model for object detection. In *The 9th European Conference on Computer Vision*, volume 3952 of *LNCS*, pages 575–588. Springer.
- Papageorgiou, C. and Poggio, T. (2000). A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33.
- Pérez, P., Hue, C., Vermaak, J., and Gangnet, M. (2002). Color-Based Probabilistic Tracking. In *The 7th European Conference on Computer Vision*, volume 2350, pages 661–675, Copenhagen, Denmark. Springer-Verlag.
- Platt, J. C. (2000). Probabilities for SV machines. In *Advances in Large Margin Classifiers*. MIT Press.
- Qi, G.-J., Hua, X.-S., Rui, Y., Tang, J., and Zhang, H.-J. (2008). Two-dimensional active learning for image classification. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.

- Ramanan, D., Baker, S., and Kakade, S. (2007). Leveraging archival video for building face datasets. In *IEEE International Conference on Computer Vision*.
- Rosenberg, C., Hebert, M., and Schneiderman, H. (2005). Semi-supervised self-training of object detection models. In *Seventh IEEE Workshop on Applications of Computer Vision*.
- Rowley, H. A., Baluja, S., and Kanade, T. (1998). Neural networks based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):22 – 38.
- Rui, Y. and Chen, Y. (2001). Better Proposal Distributions: Object Tracking Using Unscented Particle Filter. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*, pages 11–13, Kauai, Hawaii. IEEE.
- Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77:157 – 173.
- Sadeghi, A. and Farhadi, A. (2011). Recognition using visual phrases. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Schohn, G. and Cohn, D. (2000). Less is more: Active learning with support vector machines.
- Schölkopf, B. and Smola, A. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, Cambridge, MA, USA.
- Scudder, H. J. (1965). Probability of error of some adaptive pattern-recognition machines. In *IEEE Transactions on Information Theory*.
- Settles, B. (2010). Active learning literature survey. Technical Report 1648, University of Wisconsin-Madison.
- Shechtman, E. and Irani, M. (2007). Matching local self-similarities across images and videos. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Siddiquie, B. and Gupta, A. (2010). Beyond active noun tagging: Modeling contextual interactions for multi-class active learning. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Sorokin, A. and Forsyth, D. (2008). Utility data annotation with amazon mechanical turk. In *Workshop on Internet Vision*.

- Sudderth, E. B., Torralba, A. B., Freeman, W. T., and Willsky, A. S. (2005). Learning hierarchical models of scenes, objects, and parts. In *IEEE International Conference on Computer Vision*.
- Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.
- Tong, S. and Koller, D. (2001). Support vector machine active learning with applications to text classification. *Machine Learning*, 2:45–66.
- Torralba, A., Fergus, R., and Freeman, W. T. (2007). 80 million tiny images: a large dataset for non-parametric object and scene recognition. Technical Report MIT-CSAIL-TR-2007-024, Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology.
- Tur, G., Hakkani-Tür, D., and Schapire, R. E. (2005). Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, 45:171–186.
- van der Merwe, R., Doucet, A., de Freitas, N., and Wan, E. A. (2000). The unscented particle filter. In *Advances in Neural Information Processing Systems*.
- Vedaldi, A., Gulshan, V., Varma, M., and Zisserman, A. (2009). Multiple kernels for object detection. In *IEEE International Conference on Computer Vision*.
- Vezhnevets, A., Ferrari, V., and Buhmann, J. M. (2011). Weakly supervised semantic segmentation with a multi-image model. In *IEEE International Conference on Computer Vision*.
- Vijayanarasimhan, S. and Grauman, K. (2011). Large-scale live active learning: Training object detectors with crawled data and crowds. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Vijayanarasimhan, S., Jain, P., and Grauman, K. (2010). Far-sighted active learning on a budget for image and video recognition. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Vijayanarasimhan, S. and Kapoor, A. (2010). Visual recognition and detection under bounded computational resources. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*.
- Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154.
- Viola, P. A., Jones, M. J., and Snow, D. (2005). Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161.



- Vondrick, C. and Ramanan, D. (2011). Video annotation and tracking with active learning. In *Advances in Neural Information Processing Systems*.
- Vondrick, C., Ramanan, D., and Patterson, D. (2010). Efficiently scaling up video annotation with crowdsourced marketplaces. In *The 11th European Conference on Computer Vision*.
- Welinder, P. and Perona, P. (2010). Online crowdsourcing: rating annotators and obtaining cost-effective labels. In *Workshop on Advancing Computer Vision with Humans in the Loop*.
- Winn, J., Criminisi, A., and Minka, T. (2005). Object categorization by learned universal visual dictionary. In *IEEE International Conference on Computer Vision*.
- Yuen, J., Russell, B., Ce Liu, B., and Torralba, A. (2009). Labelme video: Building a video database with human annotations. In *IEEE International Conference on Computer Vision*.
- Zhang, H., Berg, A. C., Maire, M., and Malik, J. (2006a). SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*, pages 2126–2136.
- Zhang, J., Marsza, M., Lazebnik, S., and Schmid, C. (2006b). Local features and kernels for classification of texture and object categories: a comprehensive study. In *Beyond Patches workshop, in conjunction with CVPR*.
- Zhang, L., Tong, Y., and Ji, Q. (2008). Active image labeling and its application to facial action labeling. In *The 10th European Conference on Computer Vision*.
- Zhu, X. (2008). Semi-supervised learning literature survey. Technical report, Department of Computer Science, University of Wisconsin, Madison.
- Zhu, X., Lafferty, J., and Ghahramani, Z. (2003). Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *Proc. of the ICML 2003 Workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*.