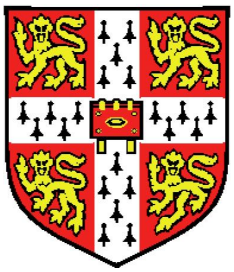
Writing a PhD Thesis

in LaTeX 2ε



Harish Bhanderi

Department of Engineering

University of Cambridge

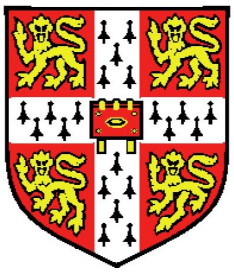
A thesis submitted for the degree of

Doctor of Philosophy

Yet to be decided

Writing a PhD Thesis

in LaTeX 2ε



Harish Bhanderi

Department of Engineering

University of Cambridge

A thesis submitted for the degree of

Doctor of Philosophy

Yet to be decided

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Detecting objects of interest from a complex scene is a basic perceptual skill in human beings and other animals. While in the area of computer science, object detection [Wikipedia, a] is a computer technology that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. And successful object detection methods play fundamental roles in a lot of application areas, which include video surveillance, driving assistance, image retrieval, etc.

Most modern detection methods fall into two categories. Some [Dalal & Triggs, 2005; Felzenszwalb et al., 2008; Ferrari et al., 2007; Lampert et al., 2008; Maji et al., 2008; Schneiderman & Kanade, 2004; Viola & Jones, 2004; Yeh et al., 2009] follow the sliding-window schema, and they detect objects by consider whether each of the sub-images contains an instance of the target object. The other methods [Barinova et al., 2010; Felzenszwalb & Huttenlocher, 2005; Fergus et al., 2003; Leibe et al., 2007, 2008; Leibe & Schiele, 2003; Maji & Malik, 2009; Mikolajczyk et al., 2006; Ohba & Ikeuchi, 1997] infer object centers based on local image features in a bottom-up manner. These methods start with detection of object parts, in the form of image patches, edgelets, or keypoints, and then infer about the target objects' status, like position, or label.

Humans still far outperform computers in the tasks of image-based recognition and detection. Only a few techniques are mature enough for daily applications, i.e., face detection [Degtyarev & Seredin, 2010] used in cameras. For years, in computer vision, lots of researchers focus on object detection from images. Their efforts include

proposing better image features [Mikolajczyk et al., 2005] or better models for object representation [Jégou et al., 2010], proposing better discriminative classifiers [Bengio, 2009] or better inference model [Teh et al., 2006], or proposing better searching techniques for exploring solution space [Lampert et al., 2009].

In this thesis, efforts are also made to improve performance of detection methods. These efforts try to explore how to use the information which are not made full use of by previous methods. Roughly, the efforts belong to two categories, the first category is exploring approaches of efficiently and effectively combing of motion information with appearance information, and the second category is exploring mutual information encoded in image features of the same object.

Most methods for detection mainly use either appearance information or motion information. And in our methods, we will address when the information from these two channels are well combined, detection performance can be much better.

The first method is developed mainly for real-time applications under limited computational power. This kind of scenarios include when using embedded sensors. This method can be considered as a two-step method. The first step deals with keypoints. It takes original data as input, and outputs keypoint clusters as detection hypotheses. This step detects, verifies, and clusters keypoints. The second step takes these keypoint clusters as input, verifies them by their appearance and motion information, and outputs the ones which pass verifications as detection results. Motion information plays a very important role in the method. The target objects are considered as possessing both special appearance patterns and motion patterns. When the second step verifies the detection hypotheses using appearance information, a biased classifier is used. This classifier produce more false alarms to pursue higher detection rate. Then motion information is used to filter out the false alarms. Also the pipeline of this method is optimized in a hierarchical way, that the later one step is, the more time consuming it is, and the fewer instances it will deal with. The method performs well under simple scene, i.e., data collected by infrared cameras in tunnel environment, and gives 100% detection rate 0% false alarm rate in one of the experiments. However, the performance of this method under complicated scene is not promising. And then we propose the second method.

The second method belongs to methods based on Hough transforms. It extend the Implicit Shape Model [Leibe et al., 2008] to combine motion information. For training,

image features together with labels and offsets to object centers of sample images are considered as codes, and inserted into a codebook. For detection, image features are detected on the target image, and then matched against the codebook using image feature as key. The matched codes will indicate the labels and object centers. During the detection step, this method firstly do motion analysis, which results in grouping results of the image features on the target image. The grouping results are used during the inference for labels and centers of the target objects. It is assumed that image features with the same motion pattern, here in the same motion group, should belong to the same object. The inference procedure then prefers the label and position infers with more consistence in the same motion group. On two datasets, the proposed method outperform the state-of-the-art method.

While the second method performs well under complicated scene, it cannot give results in real time. This is due to the time-consuming property of methods based on Hough transforms. The third method aims at improving the efficiency of the second method. Also it tries to flatten the gap of appearance and positional information. This method does not use motion information. In methods based on Hough transforms, image features are used as key to query similar codes from the codebook, and in the third method, both appearance and position are used as key. The bottom-up property of Hough transform also ignore the relationship between different image features. Actually, the mutual information encoded in the image features of the same object is very informational. The third method consider objects as point sets of, i.e., 14-dimensional, while the first 12 dimensions are appearance information, and the last 2 dimensions are positional information. The training step is almost the same with Hough-transform methods, except for how a few parameters are trained. At the detection step, instead of using the appearance information of one single feature as query, the point set of a sub-image is used as query. Pyramid Matching is used for accelerating the querying. The procedure ensures the use of the mutual information encoded in the image features of the same object.

The thesis is organised as follows. In chapter 2, background and some very related work are reviewed. In chapter 3, the method aimed at efficient detection by combining motion and appearance information is introduced. In chapter 4, the method which extends the Implicit Shape Model to incorporate motion information is proposed, this method groups object parts for detection. In chapter 5, the method which detects by

Pyramid Match Score is presented. Chapter 6 concludes, and discusses about possible improvements of the proposed methods for future work.

Chapter 2

Background and Related Work

2.1 Vision in Humans and Computers

Everyday, new technologies emerge, make everyday life more convenient, and lead advances in human culture. Among these technologies, the emergence of text searching engines like google is a great milestone. Retrieving images [Baidu, Baidu] or other multi-media data in a way similar to retrieving text is even more attractive, especially for the popularity of smart phones nowadays. There are still technical obstacles for this beautiful outlook. To decide semantic meanings for images is one, and detection performance on images is another.

The computational power of humans is not any more competitive to computers, especially nowadays, when computational ability can be conveniently accessed from the cloud. Still when talking about the performance of visual recognition and detection under general situations, humans are champion. In computer vision area, besides bar scanners and optical character recognition, few detection methods are employed in real-world applications, except that face detection methods are employed in ordinary cameras.

So what stops the object detection methods of computers from performing as good as those of humans?

When compared with humans, computers will win at almost every aspect of hardware, as shown in 2.1. Besides visual sensors, new sensors are continuously developed for computers, and computers have so many choices for sensors of very high quality.

	Computers	Humans
Quality of sensors	*	
Computational ability	*	
Representative model		*
Decision procedure		*
Information fusion mechanism		*
Training quality		*

Table 2.1: Comparison between computers and humans.

What is worth mentioning is how Microsoft's Kinect advances pose recognition [Shotton et al., 2011]. Especially, for computational ability, [Merkle, 1989] believes human brain has a raw computational power between 10^{13} and 10^{16} operations per second, and modern computers can perform much better.

As for representative model, there is no obvious evidence which proves humans are doing better than computers. While representative models of humans have long been working as inspirations and benchmarks for those of computers [Cadieu et al., 2013]. However the author believe humans shall perform better than computers in the aspects of software, which then explains why humans perform better in multiple visual tasks. And only when the vision researchers also believe so, do they develop so many new detection methods. And the dividing of functionalities are generally based on computers, while in humans, two or more of them might work together.

Human babies are taken good care of, and trained to perform very simple visual tasks for months with large amount of examples, which makes the training quality very solid. The performance of new born babies also lead to considerations about to what degree are the visual abilities decided by genes. If the training procedure has taken as long as millions of years, are there possibilities for computers to win in future?

The success of auditory speech recognition and the successful deployment in industry [Wikipedia, b] encourage vision researchers. Also new sensors advance recognition performance, such as Kinect, which can hopefully fill the gap of representative models' quality between computers and humans. Object recognition methods based on 2D images with 3D model [Kise & Kashiwagi, 2011] also makes it more possible for computers to share same representative models. Computational power can be used to make up with the short slab of training. For example, training one model for relative small amount of time with thousands of computers [Le et al., 2012] is feasible with parrel

training. The resent deep learning [Bengio, 2009] methods try to fusion representative model with decision procedure to act more like what humans might do. These new achievements are expected to fill up the gap between humans and computers in aspects of representative model, decision procedure, and training quality.

While the methods proposed in this thesis explore the information which can be further made use of, i.e., fusion of temporal and visual information, and the mutual information of different parts of the same object. The efforts here shall belong to decision procedure, and information fusion mechanism.

2.2 Artificial Intelligence

AI (Artificial Intelligence) includes a very wide range of topics, of which, computer vision is comparably difficult.

The author of this thesis is positive about computers can outperform humans in visual tasks for the reasons in the previous section. Then if we move towards the correct direction, one step forward will lead to one step nearer, and vice versa.

Needless to say, the core of Artificial Intelligence is operations at high abstraction levels. When talking about the abstraction level of computer vision, the area of text mining can be used as a benchmark, though such comparisons will not be fair. Texts themselves are at a higher abstraction level than images. And the achievements in text understanding will remind the researchers of computer vision about the unsatisfactory results of turning images to semantic labels.

Together with lots of researchers in computer vision, the author of this thesis believe achievements in abstraction of a higher level will help with the results in low abstraction levels. Something which can only be proven by time is that if the computer vision problems of low abstraction levels draw too much attention, it will not benefit the area.

Taking the problems of multi-class detection as example, routines of different types will be discussed. One routine is to explore detection of object from multiple classes at the same time. This kind of routine has been followed a lot by researchers, especially those encouraged by some recent competitions on visual tasks, e.g., VOC2012 [PASCAL, PASCAL]. Significant amount of engineering efforts are needed to win this kind of competitions. Another routine is that development of mechanisms of high abstrac-

tion level for single-class detection and mechanisms to separate objects from different classes. The problem with the first routine is that, methods will be so limited to the training dataset, and mechanisms at higher levels which are very valuable are not easy to explore. The author's approval towards competitions like VOC2012 is very limited.

Actually, perceptual mechanisms in humans are consistent with the above arguments. Recall how humans are capable detecting previously unseen objects, while they acquire knowledge of target objects through textual descriptions.

2.3 Basics of Detection Methods

In the previous section, the possible reason why humans perform better in detection is discussed. The main limits of computers are of software. Based on this, researchers present new detection methods or new methods for supporting detection, which includes: 1) new image feature or new representative model, 2) new decision procedures, or 3) better searching techniques in solution space.

2.3.1 Image Features

Image features are very important. Simple image features include features in the form of keypoints, image patches, edges, silhouette, shape [Belongie et al., 2002], or textures. There are also frameworks for combining multiple different image features or information from multiple channels [Jégou et al., 2010; Ojala et al., 1996; Shechtman & Irani, 2007; Tuzel et al., 2006], and these belong to image features at middle level [Pinto et al., 2009]. At a even higher level, image features can be organized in patterns. For example, modeling human body as a stick model [Meeds et al., 2008].

The invariance under illumination, scale, and rotation are basic requirement for image features, while some keypoint features [Bay et al., 2006; Lowe, 2004; Matas et al., 2002; Mikolajczyk & Schmid, 2004; Tomasi & Kanade, 1991] fulfill this requirement. Image features which encode global positional information perform well in human detection, i.e., Histograms of Oriented Gradients [Dalal & Triggs, 2005]. Differentials at different orders can also be used as descriptive features [Tuzel et al., 2006].

2.3.2 Decision Procedure

Generally, robustness and computational efficiency are the two main pursues in proposing image features. Based or image features, how decisions are made are also of great importance. Dimension reduction methods like principle component analysis, and feature selection methods can be considered as a glue layer between representative models and decision making procedure, which enhance both robustness and processing efficiency.

Discriminative and generative methods are the two main categories for decision making. Support vector machine, and boosting methods belong to discriminative methods. Gaussion processes [Rasmussen & Williams, 2005], Dirichlet models [Blei et al., 2003; Ferguson, 1973; Teh et al., 2006], and Bayesian graphical models [Attias, 2000] usually belong to generative methods. The difference between discriminative and generative models lay in how they use training examples to estimate parameters of the model, and how the model is used to make decisions.

One very promising method for decision making is deep learning [Bengio, 2009]. This can be considered as a special form of neural network. One of its very attractive property is it can take raw data as input and output decision results, like, object label. It deals with extraction of image feature, feature selection, and decision based on features in a unified one under the same framework. Also it can share knowledge from other domain like [Pan & Yang, 2010].

The core of machine learning methods is still the model. Training data are used to estimate parameters of the model, and the model is then used to make decisions on the testing data. The model of deep learning is a multi-layer network. The later one layer is, usually it has less nodes, and the information it deals with is more abstract.

The earlier layers of deep-architecture networks act as feature extraction module. The first layer defines rules of how to make abstraction on the raw data. And these layers can be trained using data from other domains. This is why deep learning methods can make use not only domain-specific information. Another aspect which separate deep learning methods from traditional neural networks is during the training procedure. Traditional neural networks are trained as a whole, which means the inferring for lots of parameters at the same time, and training quality requires extremely large amount of data. The training procedure for deep learning methods is layer-wised.

There is a merit for evaluate the training quality for each layer. And after one layer is trained, the output can be used to train the next layer. This means that, each time only a few parameters need to be estimated, which is more robust and efficient, and avoid the requirement for very large amount of data. When the number of layers increase, the abstraction level of the original data enhance, and decisions are easier.

2.3.3 Solution Space Explore

Besides proposing representative models and decision procedures, there are lots of work [Dean et al., 2013; Pirsiavash & Ramanan, 2012; Rahtu et al., 2011] focusing on how the solution space from the decision making procedure can be explored. The methods following the sliding-window schema need to consider each differently scaled sub-image of a target image, to answer wether this sub-image contains an object of interest or not, this amount is extremely large, and even enumerate all the sub-windows is computationally infeasible.

Currently method of [Isukapalli & Greiner, 2003] and its variants [Yeh et al., 2009] for sliding window search work very efficiently, and give best detection hypothesis in polynomial time in experiments. There are variants [Lehmann et al., 2011] which share the same strategy for methods based on Hough transforms. These methods employ branch-and-bound mechanisms during the search for the best detection hypothesis. While map-and-reduce is popular is distributed computations, the underline philosophy is very similar.

Take the testing process of linear support vector machine as an example. When a sub-image is described as a bag of features, each feature will contribute to a specific dimension on the final descriptive vector. And when use this vector by support vector machine for decisions, it is simple linear add-up of the decisions of each single feature. So decision score for each image feature can be calculated. Thus the upper-bound and low-bound for all sub-images contained in a certain rectangle can be estimated, which can be used to filter out lots of rectangles definitely do not contain the best solution. This filtering out is the core for efficiency.

2.4 Advances in Object Detection

Three methods will be proposed in the thesis, methods very related to each method will be further reviewed in the corresponding chapter. Here we review some recent advances in object detection.

Detection is drawing a lot of attention [Barinova et al., 2010; Dalal & Triggs, 2005; Felzenszwalb & Huttenlocher, 2005; Felzenszwalb et al., 2008; Fergus et al., 2003; Ferrari et al., 2007; Lampert et al., 2008; Leibe et al., 2007, 2008; Leibe & Schiele, 2003; Maji et al., 2008; Maji & Malik, 2009; Mikolajczyk et al., 2006; Ohba & Ikeuchi, 1997; Schneiderman & Kanade, 2004; Viola & Jones, 2004; Yeh et al., 2009], and it will continue to. While some methods are unique and very heuristic.

Instead of proposing class-specific methods, [Alexe et al., 2010] try to evaluate how like a sub-image contains an object of any class. In the method objects are defined by very general properties, which include having closed boundary, being different from surroundings, and sometimes being unique and salient in the image. By combining saliency detection, color contrasting, edge detection, and image segmentation methods in a Bayesian framework, they give convincing performance in general-purpose object detection.

Bag of image features [Jégou et al., 2010] is an important advance for object detection. Before the method, potential objects are described using feature extracted from raw pixels, while the method describes objects using object components. In the work following, [Yang et al., 2009] added a biased sampling component for describe each object. Instead of described by one group of features, the objects are described by several groups of features, and then decisions are made using multi-label multi-class classification.

Some pioneer work also consider about recognizing objects in 3D space. The method proposed in [Kise & Kashiwagi, 2011], tracks keypoints of the same object, and generate feature accordingly, which will include 3D information of the object, and feed this feature to decision step, which makes the results very appealing.

Also excellent performance of deep learning inspired new methods to reconsider about object detection. The discover of invariants and learning of a detector from unlabeled data are explored by [Le et al., 2012].

Still, one of the challenging subjects in object detection is rotation-aware detection.

Lots of object detection methods ignore scale and rotation changes by using of scale- and rotation-invariant image features, i.e. SIFT. The primary direction of the SIFT feature is used by [Mikolajczyk et al., 2006] for locally estimate the rotation angle of object. This method heavily rely on the SIFT feature. Object is represented by a graph with features as nodes in [Jiang & Yu, 2009], and scale- and rotation-invariant object match is made by the matching of two graphs. Instead of being a general method for object detection, this method is mainly used for object matching. Most rotation-aware methods follow [Rowley et al., 1998]. This method firstly trains a neural network for rotation estimation. According to the rotation estimation result, the tested sub-image is rotated to a normalized angle and then fed into other classifiers for verification of object hypothesis. Still efficient methods to robustly deal with object rotation in the detection procedure are preferred.

Chapter 3

Efficient Detection by Combining of Appearance and Temporal Information

3.1 Introduction

From time to time, efficient detection methods under limited computational power are very applicable in various areas, such as driving assistance using embedded sensors. Here, a method pursuing efficiency and real-time detection is proposed. The proposed detection method makes use of both appearance and motion information of the target objects. By well optimizing of the detecting pipeline, the method works in real time, and gives 100% detection rate and 0% false alarm rate in one experiment of data collected by infrared cameras.

As always, detection performance and efficiency are the two important aspects of this method. Challenges come from the need to distinguish noisy objects as shown in Figure 3.1 and the real-time requirement. In experiments, besides the target object a lot of noisy objects also exist. Some of the noisy objects cannot even by distinguished from the target objects by only appearance. The clutter property of the sensed data makes the detection challenging. The proposed method meets this challenge by making use of both appearance and temporal information of the target objects. There are two main steps in the method.

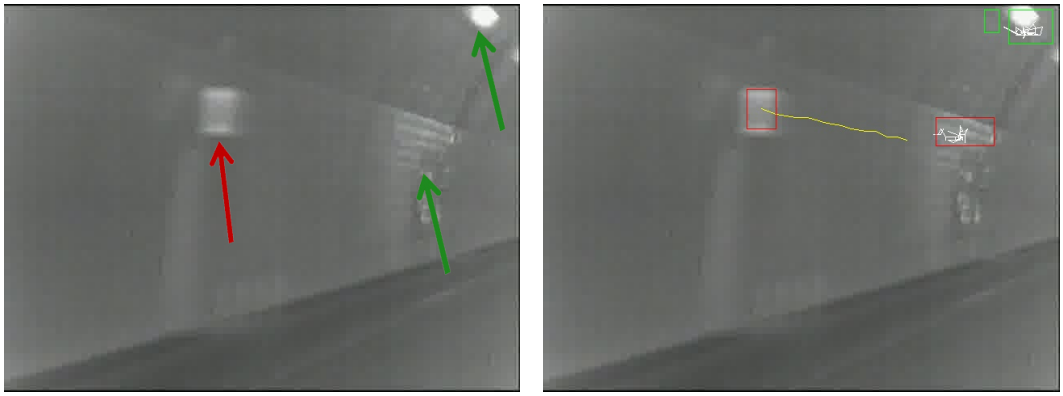
The first step deals with keypoints, and it takes original data as input, and outputs keypoint clusters as detection hypotheses. In this step, keypoints are detected, verified and then clustered. To detect keypoints, all points on each frame are uniformly sampled and filtered with pre-set intensity thresholds. Then the keypoints are verified by a simple keypoint appearance model powered by k-means. At the end of the first step, the keypoints are clustered based on the Euclidean distance.

The second step takes the keypoint clusters as input, verifies them by appearance and temporal information, and outputs the ones pass verifications as detection results. In the second step, the keypoint clusters are labeled based on appearance by an adaboost machine, which is trained using intensity histograms of keypoint clusters from target objects and keypoint clusters from noisy objects. The keypoint clusters are also tracked by temporal association through frames. Motion information encoded in the trajectories are used to further verify the keypoint clusters. Finally, the keypoint clusters which pass both appearance and temporal verifications are decided as target objects.

This pipeline is designed also with consideration of the requirement for efficiency. The method deals with the large amount information contained on one frame following a hierarchical manner. The later one step is, the more time-consuming it is and the fewer instances it deals with. From an image containing 10^5 pixels, 10^4 points go through keypoint detection step of testing by intensity thresholds. Then in average, 10^3 keypoints are detected, and verified, leaving about 10^2 to be clustered. Afterwards, fewer than 10 keypoint clusters are left, which are dealt with by the very time-consuming steps of generating image features and tracking.

The advantage of the method is its ability to give promising detection results from cluttered data in real time. Besides, the method is also a successful attempt to combine bottom-up and classification methods, and a successful attempt to combine both appearance and temporal information.

Firstly, the method is proposed with application scenario of data collected by infrared cameras, then it is also extended to corporate with data collected by ordinary cameras. This chapter is organized as follows: section 3.2 reviews related work, section 3.3 introduces application background, section 3.4 proposes pipeline of the method, section 3.5 gives experimental results, section 3.6 gives results by extending the method for data collected by ordinary cameras, and section 3.7 concludes.



(a) (b)

Figure 3.1: Original data and detection results. In (a), the red arrow points to the target object: emergency telephone indicator, and the green arrows point to noisy objects. In (b), red rectangles mark detection hypotheses labeled as positive using appearance information, and green rectangles mark negative ones. Yellow trajectories mark detection hypotheses labeled as positive using temporal information, and white trajectories mark negative ones.

3.2 Related Work

Most modern detection methods fall into two categories. Some [Dalal & Triggs, 2005; Felzenszwalb et al., 2008; Ferrari et al., 2007; Lampert et al., 2008; Maji et al., 2008; Schneiderman & Kanade, 2004; Viola & Jones, 2004; Yeh et al., 2009] follow the sliding-window schema, and they detect objects by consider whether each of the sub-images contains an instance of the target object. Classifiers are usually employed by these methods. The other methods [Barinova et al., 2010; Felzenszwalb & Huttenlocher, 2005; Fergus et al., 2003; Leibe et al., 2007, 2008; Leibe & Schiele, 2003; Maji & Malik, 2009; Mikolajczyk et al., 2006; Ohba & Ikeuchi, 1997] infer object centers based on local image features in a bottom-up manner. The proposed method makes advantages of both frameworks. Following the bottom-up manner, keypoints are detected, verified, and clustered. After these steps, the keypoint clusters are considered as detection hypotheses. Then following the sliding-window schema, the keypoint clusters are verified by their appearance and temporal information using discriminative methods.

Previous methods [Russakovsky & Ng, 2010] also consider the combination of the

two frameworks. Detection hypotheses are gained using Hough transform and then verified by support vector machines in [Maji & Malik, 2009; Yarlagadda et al., 2010]. The methods in [Gall & Lempitsky, 2009; Okada, 2009], use randomized decision trees for both decisions whether local features belonging to foreground objects or not and decisions of their Hough votes. The method proposed in [Lehmann et al., 2011] describes both frameworks in the same manner. While giving state-of-the-art detection performance, they can't meet the requirement for efficiency as this method does.

This work is also related to data association methods at time dimension. In tracking, the main attention is focused on solving a data association problem to explain conflicts in data as well as recovering from tracking failures within a low time cost. In [Yang et al., 2007], the joint likelihood maximization is represented by the Nash Equilibrium in a game. The main contribution is the time complexity of solving a game-theoretic problem is low and make trackers compete for the region that is in a conflict. In [Zhang et al., 2008], the MAP corresponds to the max flow of a well designed graph. The method has the advantage of taking into consideration of missed detections on middle frames and the low time cost. In [Leibe et al., 2007], the main idea is to firstly connect very faithful detection response pairs, then solve the data association problem via low-time-complexity Hungarian algorithm, and refine the result in a EM manner at the last step. And the main concern of these work is solving the high time complexity in the data association problem.

This work is also related to feature grouping methods [Yarlagadda et al., 2010], methods detecting using trajectories [Brostow & Cipolla, 2006; Brox & Malik, 2010], tracking methods [Huang et al., 2008; Leibe et al., 2007], and methods integrating appearance and temporal information [Wojek et al., 2010].

3.3 Application background

When developing the method, potential application scenarios are within consideration. The method aims to perform detection in real time, and serve as an effective unit in positioning automobiles in tunnel environment.

Positioning of automobiles acts an important role in autonomous driving, and it is also of great importance for driving assistance, vehicle navigation, etc. When GPS sensors function properly, it is usually easy for an automobile to estimate the position

of itself. While in tunnel environment, for most of the time, there is no GPS signals available. A new positioning system which functions properly in tunnel environment is very necessary [Davidson et al., 2008].

In most tunnels which appear on express ways in Japan, there are lots of signs which appear at equal intervals. Attention can be focused on the emergency telephone indicators, which appear every 200 meters in tunnels. The absolute coordinates of the emergency telephone indicators can be obtained by the method of [Ono et al., 2012]. While travelling in tunnels, if the emergency telephone indicators can be sensed, and the distance from the automobile to the indicators can be estimated, then the absolute position of the automobile can be inferred. Detection methods i.e. [Wojek et al., 2010] based on ordinary cameras fail due to darkness. In experiments infrared cameras, which are usually equipped by recent intelligent vehicles and suitable in dark environment, are used.

In tunnel environment, besides the target objects, a lot of noisy objects also appear, e.g. ordinary lights, other vehicles, and other vehicles' shadows. So to well distinguish target objects is of importance. And such kind of applications usually require real-time detections. And the proposed method successfully meet the requirements.

3.4 Detection Pipeline

The method can be considered as a two-step method. The first step deals with keypoints. It takes original data as input, and outputs keypoint clusters as detection hypotheses. The second step takes these keypoint clusters as input, verifies them by their appearance and motion information, and outputs the ones which pass verifications as detection results.

3.4.1 Keypoint Detection

In data collected using ordinary cameras, keypoints [Bay et al., 2006; Lowe, 2004] invariant to rotations, affine changes, and illumination changes are preferable. In the case of using data collected by infrared cameras, keypoint detection intends to provide hypotheses for target objects, i.e. emergency telephone indicators. Thus intensity is of great importance. This method employs a simple yet useful method to detect keypoints.

(a) (b)

Figure 3.2: Keypoint detection.

Firstly, points are uniformly sampled with width step W, and height step H. In this manner the magnitude of instances is reduced by at lease one order. Then the points are considered as keypoints if they pass the test which verifies them by setting intensity thresholds .

Here Gaussian distribution is assumed for the intensities of the points.

let$\{\mathbf{x}\}$ denote all the sampled points, $I_{\mathbf{x}}$ the intensity of each point, and $l_{\mathbf{x}}$ the label. If the point is considered as belonging to target objects, $l_{\mathbf{x}} = 1$, otherwise, $l_{\mathbf{x}} = 0$. By setting lower threshold, $I_{\mathbf{x}}^{th1}$, and higher threshold, $I_{\mathbf{x}}^{th2}$, the probability that points belongs to target objects based on their falling into this interval is given by,

$$P(l_{\mathbf{x}} = 1 | I_{\mathbf{x}}^{th1} \leq I_x \leq I_{\mathbf{x}}^{th2}) = \frac{P(l_{\mathbf{x}} = 1, I_{\mathbf{x}}^{th1} \leq I_x \leq I_{\mathbf{x}}^{th2})}{P(I_{\mathbf{x}}^{th1} \leq I_x \leq I_{\mathbf{x}}^{th2})} \ . \tag{3.1}$$

At this step, that as few points belonging to the target objects as possible are excluded is also considered. The probability of ont point falling into the defined interval based on its belonging to target objects is given by,

$$P(I_{\mathbf{x}}^{th1} \leq I_x \leq I_{\mathbf{x}}^{th2} | l_{\mathbf{x}} = 1) = \frac{P(l_{\mathbf{x}} = 1, I_{\mathbf{x}}^{th1} \leq I_x \leq I_{\mathbf{x}}^{th2})}{P(l_{\mathbf{x}} = 1)} \ . \tag{3.2}$$

And points of which the intensities fall in the pre-set thresholds are detected as keypoints.

3.4.2 Keypoint Verification

As shown in Figure 3.5(b), the detected keypoints don't just belong to target objects, but also belong to background. And for training, keypoints belonging to target objects are considered as positive ones, otherwise negative.

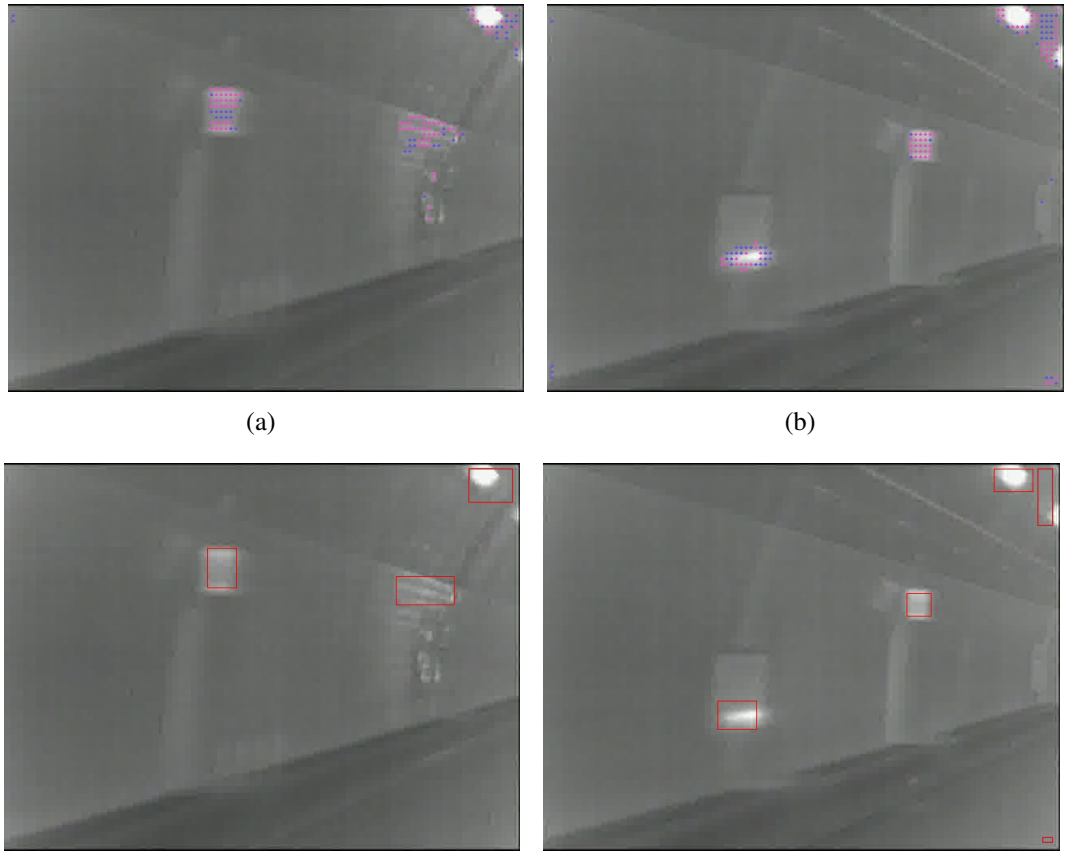
To verify the keypoints, the appearance of the sub-image around each keypoint is used. Intensity histograms are used to describe the appearance. Because the noisy keypoints come from several sources, i.e., way of the tunnel, ordinary lights, and other vehicles in the case of emergency telephone indicator detection. Thus robust linear classifiers are not suitable for the verification. Here, a general model in the form of a simple mixture is used. The *k*-means method is used to cluster the intensity histograms, $\{A_{\mathbf{x}}, l_{\mathbf{x}} = 1\}$, of the positive keypoints, and, $\{A_{\mathbf{x}}, l_{\mathbf{x}} = 0\}$, of the negative keypoints.

Let $\{C_1^i, i = 1, 2, ..., n_1\}$ denote the intensity histogram centers of the positive keypoints, and $\{C_0^i, i = 1, 2, ..., n_2\}$ the negative. For each C_1, the average Euclidean distance between $\{C_0^i, i = 1, 2, ..., n_2\}$ is calculated as,

$$Eu(C_1^i) = \frac{1}{n_2} \sum_{j=1}^{n_2} Euclid(C_1^i, C_0^j) \ . \tag{3.3}$$

Here, $Euclid(\cdot)$ calculates the Euclidean distance, and $Eu(\cdot)$ is a evaluation function of the positive feature centers. The positive feature centers are ranked by $Eu(\cdot)$, and the positive feature centers with largest $Eu(\cdot)$ are chosen and used for verification.

For verification, the intensity histogram of each keypoint's surrounding sub-image is extracted. Then the Euclidean distance between the extracted intensity histogram and its nearest positive feature center is calculated. If this distance exceeds a threshold, $D_{A_{\mathbf{x}}}^{th}$, it is considered as negative, else it is considered as positive. Here, for simplicity, unlike [Stauffer & Grimson, 1999], the same threshold is used for all components of the mixture.



(a) (b)

(c) (d)

Figure 3.3: Keypoint verification and clustering. Red circles mark keypoints which pass the verification, while blue marks failed ones. Rectangles mark keypoint clustering results.

3.4.3 Keypoint Clustering

After the keypoint verification step, on some frames, the result is pretty good, while on other frames, appearance of the keypoints is not enough for decisions of whether the keypoints belonging to target objects or not. Here generation of keypoint trajectories is not feasible, since nearby keypoints are similar in appearance and the time complexity of associating such a large number of keypoints along time dimension is high. So the keypoints are clustered, and then data association in time dimension only need to deal with a small number of keypoint clusters.

To cluster the keypoints, a minimum spanning tree (mst) is built using the pairwise Euclidean distance between two keypoints. And the mst is split by cutting edges larger a threshold. This results in a grouping results of the keypoints, denoted by, $\gamma = \{\mathbf{g}\}$.

3.4.4 Keypoint Cluster Verification by Appearance

For each keypoint cluster, the smallest bounding rectangle is considered as detection hypothesis, as shown in Figure 3.8(c) and Figure 3.8(d).

In the case of emergency telephone indicator detection, there are two main sources of noise. Ordinary lights are the first, and other vehicles with their shadows are the second. The global appearance of ordinary lights is different from that of the emergency telephone indicators. As ordinary lights get further from the infrared camera, the intensity of its corresponding sub-image in the collected data gets lower. At a certain distance, the intensity of the ordinary lights is almost the same with emergency telephone indicators'. And for ordinary lights of which the intensity is higher than the emergency telephone indicators', the transition regions from them to tunnel walls will have similar intensity with the emergency telephone indicators. This means though locally the emergency telephone indicators share the same appearance with ordinary lights, they can still be distinguished globally by appearance. As for other vehicles and their shadows, the intensity range of them is very close to the emergency telephone indicators', and they can hardly be distinguished just by appearance.

At this step, the keypoint clusters are verified by their appearance, aiming at filtering out keypoint clusters which do not share the same appearance model with the target objects. An Adaboost machine is trained using intensity histograms of the target objects and noisy objects. In the case of emergency telephone indicator detection,

the appearance of other vehicles is close to the emergency telephone indicators', and they are not used for training the machine. For training of the machine, labeled 32-dimensional intensity histograms are firstly normalized. Then each weak classifier of the machine makes decision on one dimension of the intensity histograms. After this step, each keypoint cluster is either labeled as positive or negative.

In this step, to emphasize the Adaboost machine's performance on the positive training examples, the initial weights of the positive training examples are set to be much larger than the weights of the negative training examples. Since in practice, whether each keypoint cluster is a target object or not is decided by both appearance and motion information. And the difficulties of exclude noisy objects can be left to the later steps.

Figure 3.4: Keypoint cluster verification by appearance. Red rectangles: positive detection hypotheses, and green: negative detection hypotheses.

3.4.5 Keypoint Cluster Tracking

Not all noisy detection hypotheses can be excluded by using appearance, as shown in Figure 3.4. To distinguish keypoint clusters belonging to other vehicles and their shadows, the keypoint clusters are tracked through frames to generate trajectories.

In this case, the problem of keypoint cluster tracking is relatively simple, since no occlusion occurs. To keep the method on-line and maintain efficiency, a pool of trajectories are kept, $\tau = \{T_{\mathbf{g}}^i, i = 1, 2, ..., n\}$, and new detection hypotheses act as detection responses, $\nu = \{n_{\mathbf{g}}^i, i = 1, 2, ...m\}$, in tracking. The problem of tracking is modeled as finding best data association hypothesis, H^*, between the trajectory set and detection response set as,

$$\begin{aligned} H^* &= \arg\max_{H \in \eta}(P(H|\tau, \nu)) \\ &= \arg\max_{H \in \eta}(\prod_{(T_{\mathbf{g}}^i, n_{\mathbf{g}}^j) \in H} P_{link}(n_{\mathbf{g}}^j | T_{\mathbf{g}}^i)) \ . \end{aligned} \tag{3.4}$$

Let $u_{ij} = 1$ or 0 indicates $n_{\mathbf{g}}^j$ is linked to $T_{\mathbf{g}}^i$ or not, and assuming each trajectory can link once and each detection response can only be linked once, the problem can be modeled as,

$$\begin{gathered} \arg\max_{u_{ij}} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{ij} \ln P_{link}(n_{\mathbf{g}}^j | T_{\mathbf{g}}^i) \\ s.t. : \ u_{ij} = 0 \text{ or } u_{ij} = 1, \forall \ i, \forall \ j; \\ \sum_{i=1}^{n} u_{ij} \leq 1 \ ; \sum_{j=1}^{m} u_{ij} \leq 1 \ . \end{gathered}$$

Here, $P_{link}(n_{\mathbf{g}}^j | T_{\mathbf{g}}^i)$ is defined by the appearance difference, the scale difference, and the time gap between the last detection response contained in $T_{\mathbf{g}}^i$ and $n_{\mathbf{g}}^j$. While Hungarian algorithm [Kuhn, 1955] gives near-optimal solution, the method follows a very simple manner for the solution by finding the best matched pairs and excluding them until no matching pairs can be found.

3.4.6 Keypoint Cluster Verification by Motion

As shown in Figure 3.12, the trajectories from keypoint clusters belonging to target objects are different from other objects'. In this step, the temporal information encoded in the trajectories are used to further verify the keypoint clusters. In the case of emergence telephone indicator detection, a linear model is used to fit each trajectory, and the significance of the fitting is the criteria for decisions.

Let $(x^i_{\mathbf{g}}, y^i_{\mathbf{g}})$ denote the coordinate of the ith element belonging to a trajectory. The linear assumption is that $y^i_{\mathbf{g}} = a_0 + a_1 x^i_{\mathbf{g}}$. The significance of the fitting is defined as,

$$r = \left| \frac{\sum_i \left(x^i_{\mathbf{g}} - \bar{x}_{\mathbf{g}}\right)\left(y^i_{\mathbf{g}} - \bar{y}_{\mathbf{g}}\right)}{\left[\sum_i \left(x^i_{\mathbf{g}} - \bar{x}_{\mathbf{g}}\right)^2 \cdot \sum_i \left(y^i_{\mathbf{g}} - \bar{y}_{\mathbf{g}}\right)^2\right]^{1/2}} \right| . \tag{3.5}$$

And the value of r is used to decide the trajectories of the keypoint clusters as belonging to emergency telephone indicators or not.

3.4.7 Pipeline Summary

	Appearance /Motion	Online /Offline	Discriminative /Generative
Keypoint Detection	Appearance	Offline	Generative
Keypoint Verification	Appearance	Offline	Generative
Keypoint Clustering	Appearance	Online	
KC Verification by Appearance	Appearance	Offline	Discriminative
KC Tracking	Motion	Online	
KC Verification by Motion	Motion	Online	Discriminative

Table 3.1: Summary of all steps in the pipeline. Here, KC is short for keypoint cluster.

Before proposing the decision step for detection, Table 3.1 summaries the steps in the pipeline from the type and style of the information source, and whether each step belongs to discriminative or generative methods. Generally speaking, motion information and discriminative methods are at later steps. This is because the computational cost is high, while discriminative methods need more information to guarantee performance. Online information are also mainly used in later steps, this is because without

information provided by model trained offline, there is no instances to generate online information. Also the computational cost of online information is higher on the run.

3.4.8 Object Detection

For each keypoint cluster on the current frame, there exists label given by the Adaboost machine according to its appearance, and the significance of fitting its trajectory as a straight line. For each keypoint cluster, it is considered as an target object if and only if its label given by the Adaboost machine is positive, its trajectory is longer than l^{th}, and the significance of fitting its trajectory into a straight line is larger than r^{th}.

Each trajectory not only connects the detection responses, but also connects the decisions for each detection responses made by their appearance and motion patterns. The target objects and noisy objects actually appear in successive frames, and even if we make a wrong decision on one frame, we can expect to recover from this mistake based on the results on other frames. The final results is based on the trajectories of decisions. When one trajectory ends, if most of the decisions it connects are positive, then this trajectory is considered as positive.

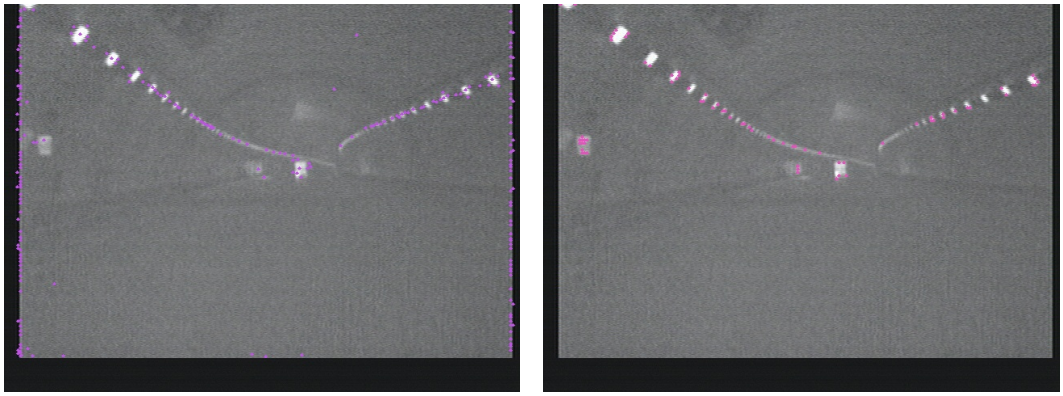
3.5 Experimental Results

In this section, this method is tested on detection performance and efficiency. There are two experiments carried on, and results from both are reported.

3.5.1 First Experiment

Data

For the first experiment, our experimental vehicle on top of which installed infrared cameras take several tours of the Awagatake tunnel. About 4,000 frames are collected for each tour. The frame size is 640×480, the intensity range is [0,255], and the frame rate is 30 frames per second of the camera, and 15 frames per second of the collection program.



(a) (b)

Figure 3.5: Keypoint detection. In (a), keypoint detection method in [Lowe, 2004] is used, and in (b), the keypoints are detected using the proposed method. And the proposed keypoint detection is more suitable for detecting keypoints belonging to emergency telephone indicators.

Implementation Settings

All models are trained using data from one tour, while evaluated on data from another tour. Firstly, all emergency telephone indicators are marked in the form of rectangles on all frames from the training tour.

To set intensity thresholds for keypoint detection, Gaussian distribution is assumed for the points belonging to target objects. Width step W set to 3, and H set to 4. Keypoints in the marked rectangle are sampled from the frames of the training tour, and used to estimate $I_{\mathbf{x}}^{th1}$ and $I_{\mathbf{x}}^{th2}$. Following the 3σ principle, $I_{\mathbf{x}}^{th1}$ is set to 160 and $I_{\mathbf{x}}^{th2}$, 190. In Figure 3.5, we also compare the keypoint detection results with the results using SIFT.

The keypoints used to set intensity thresholds, together with some keypoints randomly sampled from training data, which are out of the marked rectangles are used to train the mixture model, which is then used in the verification of the keypoints. Note this model and the training is not very accurate, since more accurate marking means more manual efforts. Also, future steps can filter out the false alarms produced by this step. About 30,000 intensity histograms of the positive keypoints are sampled, and about 3,000,000 of the negative. When using of k-means for clustering the positive intensity histograms, k is set to 40, and k set to 400 for negative. The k values over-

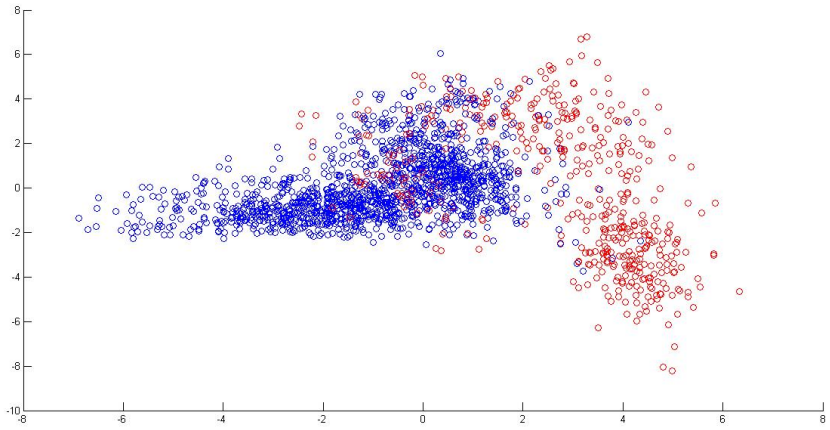


Figure 3.6: Dimension reduction of intensity histograms manually marked as positive and negative, which have 2 dimensions by principle component analysis. Blue circles: positive, and red: negative.

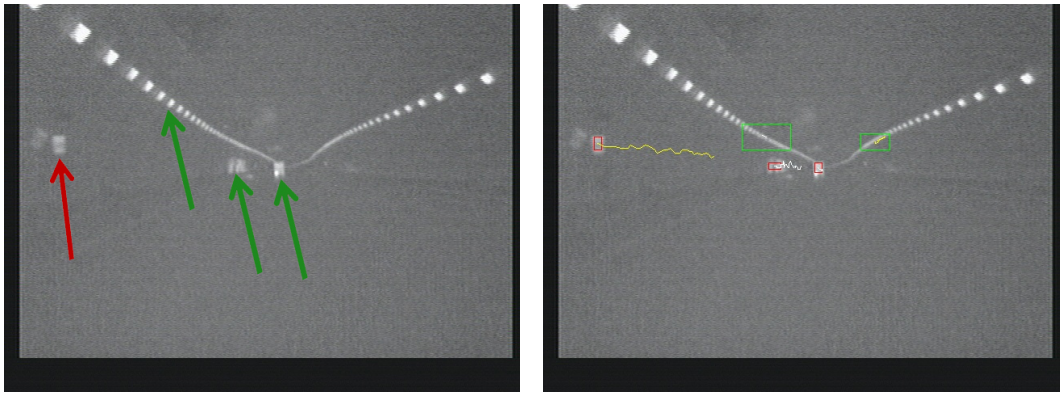
segment both feature sets. The threshold to verify keypoints, $D^{th}_{A_{\mathbf{x}}}$ is set to 0.14 for the normalized histograms.

For keypoint clustering, the threshold to split the mst is set to 10, which is half the largest height of the emergency lights sensed by the camera.

The Adaboost machine to distinguish other vehicles and their shadows is trained by intensity histograms of positive keypoint clusters and negative keypoint clusters. We manually mark 466 positive and 1,421 negative keypoint clusters. And in Figure 3.6, show the image features of positive and negative training examples for the Adaboost machine, which are reduced to two dimensional by PCA. The trained Adaboost machine is tested on the training dataset, and the classification rate is 85%.

During keypoint cluster tracking, whether a detection response can be linked to a trajectory or not is constrained by position and scale changes. Here scale change limit is set to 4. When the trajectories are fitted as lines, the linear model is also used in associating new detection responses.

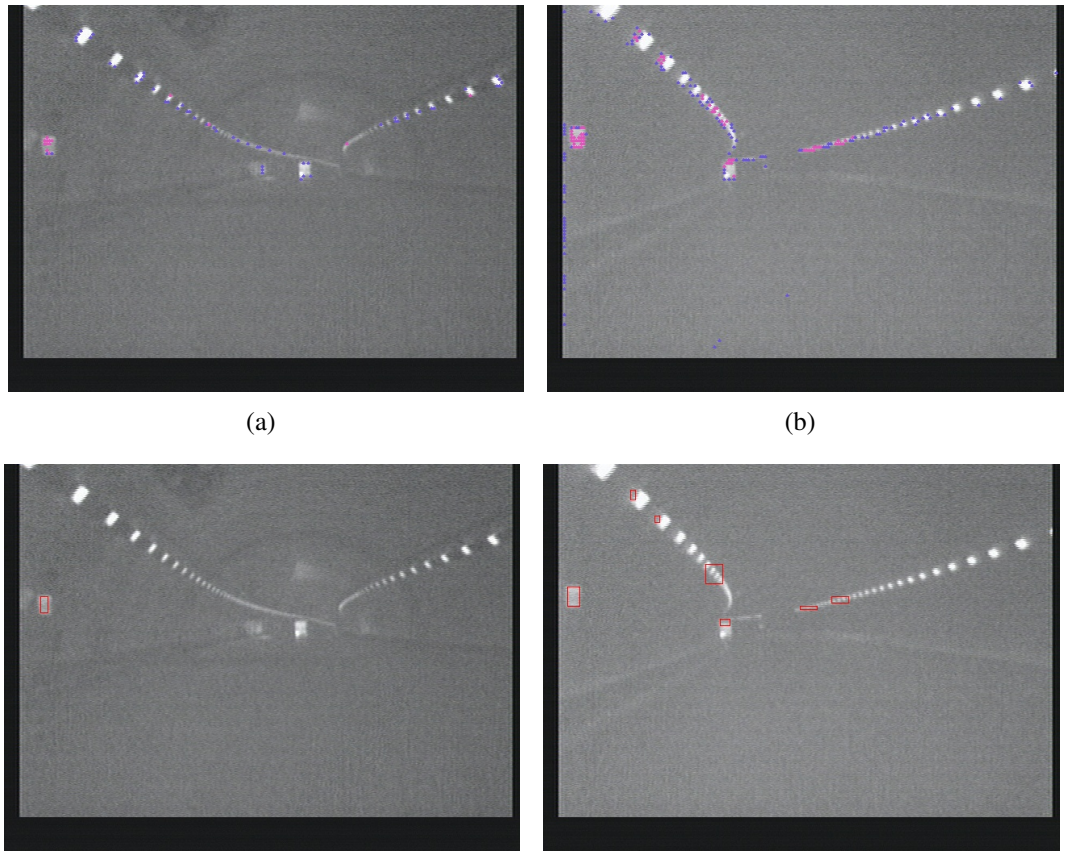
In this experiment, l^{th} is set to 5, r^{th} is set to 0.7, and R^{th} is set to 70%. As the figures in the section for pipeline are all from the second experiment, here the results of each step of the data from the first experiment are also given. In Figure 3.7, the original data and sample detection results are given. In Figure 3.8, the results from



(a) (b)

Figure 3.7: Original data and detection results. Red arrow points to the target object in 3.7(a).

keypoint verification and clustering are given. In Figure 3.9, the results from keypoint cluster verification by appearance information are given. In Figure 3.10, the results from keypoint cluster tracking are given. And at last, in Figure 3.11, the final detection results are given.



(a) (b)

(c) (d)

Figure 3.8: Keypoint verification and clustering.

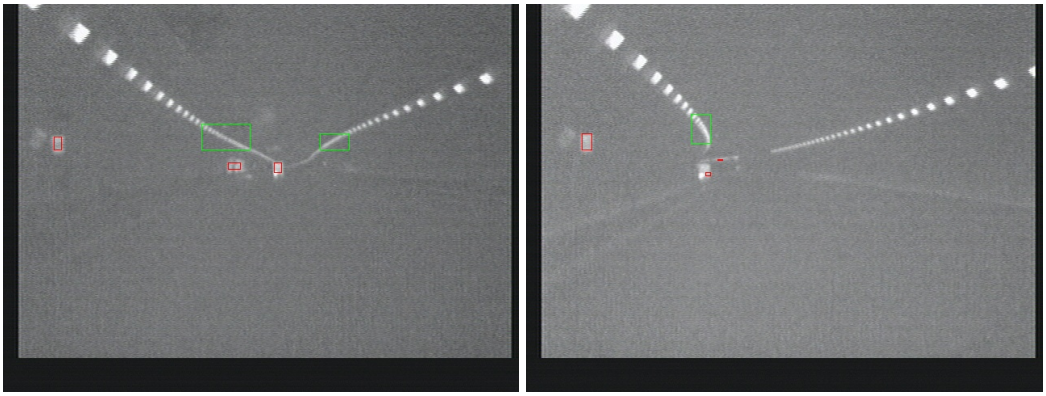


Figure 3.9: Keypoint cluster verification by appearance. Red rectangles: positive detection hypotheses, and green: negative detection hypotheses.

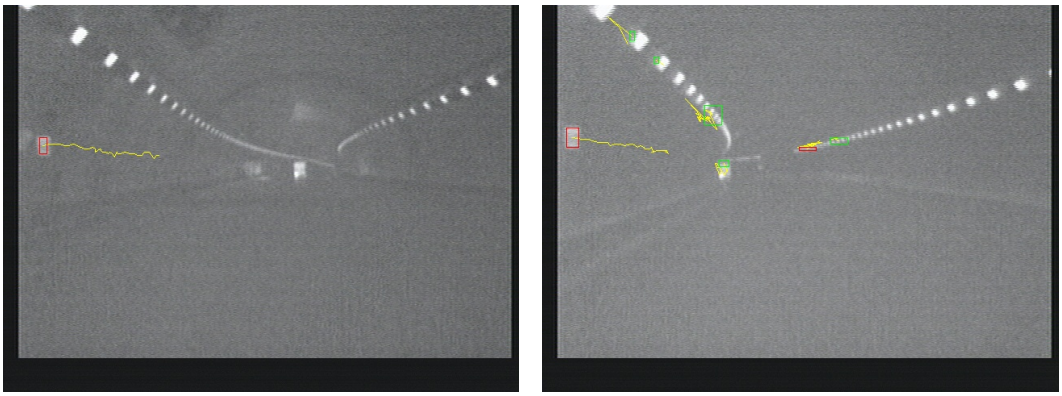


Figure 3.10: Keypoint cluster tracking.

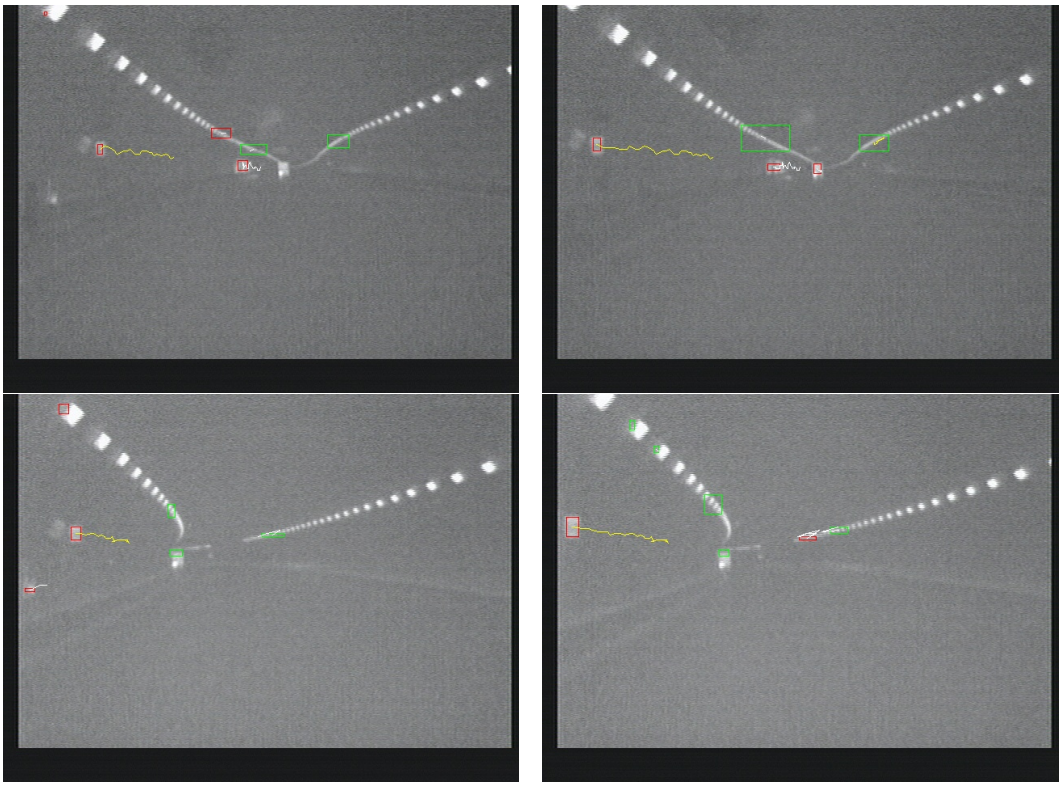


Figure 3.11: Detection results.

Detection Results

On a laptop with Intel Core2 Duo 2.8GHz processors, the method deals with real data at a frame rate of 34 frames per second, and this fulfills real-time requirements.

The detection rate and false alarm rate is evaluated on 250 frames, as shown in Table 3.2.

Total number	113
Correctly labeled	102
Miss detections	11
False alarms	21
Detection rate	90%
False alarm rate	19%

Table 3.2: Detection rate and false alarm rate.

3.5.2 Second Experiment

Data The results of the first experiment is not satisfactory, and then the second experiment is carried out. A better far infrared camera is used, and also the zoom of the camera is adjusted for better images. Then with the new camera mounted on top, the experimental vehicle took several tours of the Awagatake tunnel. About 7,000 frames are collected for each tour. The frame size is 640×480, the intensity range is [0,255], and the frame rate is 30 frames per second of the camera, also of the data collection program which is provided by the camera maker.

Implementation Settings The same with experiment one, all models are trained using data from one tour, and evaluated on data on another tour.

About intensity thresholds for keypoint detection, $I_{\mathbf{x}}^{th1}$ is set to 160 and $I_{\mathbf{x}}^{th2}$, 190. According to the sensed emergency telephone indicator, width step W set to 3, and H set to 4.

Instead of training a new mixture model for keypoint verification, the older one from experiment one is used, since the performance of this step is not critical.

For keypoint clustering, the threshold to split the mst is set to 40, which is half the largest height of the emergency telephone indicators sensed with new camera and experimental settings.

The Adaboost machine to distinguish other vehicles and their shadows is trained by intensity histograms of positive keypoint clusters and negative keypoint clusters. If the Adaboost machine is trained by averagely weighted training examples as in experiment one, its correct rate on the training examples is overall 84%. When trained using bias weighted training examples, which means the initial weight of the positive training examples are here 7 times as large as those of the positive training examples, its correct rate on the positive training examples is 94%, and 77% on the negative training examples.

The main difference between experiment one, and experiment two in the pipeline lays here. The same weights are assigned to positive and negative training examples during training the Adaboost machine, while biased weights are assigned here. This will results in an Adaboost machine which gives better performance on target objects, and perform worse on noisy objects. The produced false alarms can be later filtered out by more powerful step which uses motion information.

During keypoint cluster tracking, whether a detection response can be linked to a trajectory or not is constrained by position and scale changes. Here scale change limit is set to 4. When the trajectories are fitted as lines, the linear model is also used in associating new detection responses.

Detection Results

On an ordinary desktop computer with Intel Core2 Quad 2.6GHz processors, the method deals with real data at a frame rate of 41 frames per second, and this fulfills real-time requirements.

The detection rate and false alarm rate are evaluated on the keypoint clusters, as shown in Table 3.3. More detection results are shown in Figure 3.12.

Total number	472
Correctly labeled	468
Miss detections	4
False alarms	22
Detection rate	99.2%
False alarm rate	4.4%

Table 3.3: Detection rate and false alarm rate.

The detection rate and false alarm rate of the first experiment [Wang et al., 2010] are 90% and 19%, while evaluated on a much smaller dataset. Here the results are

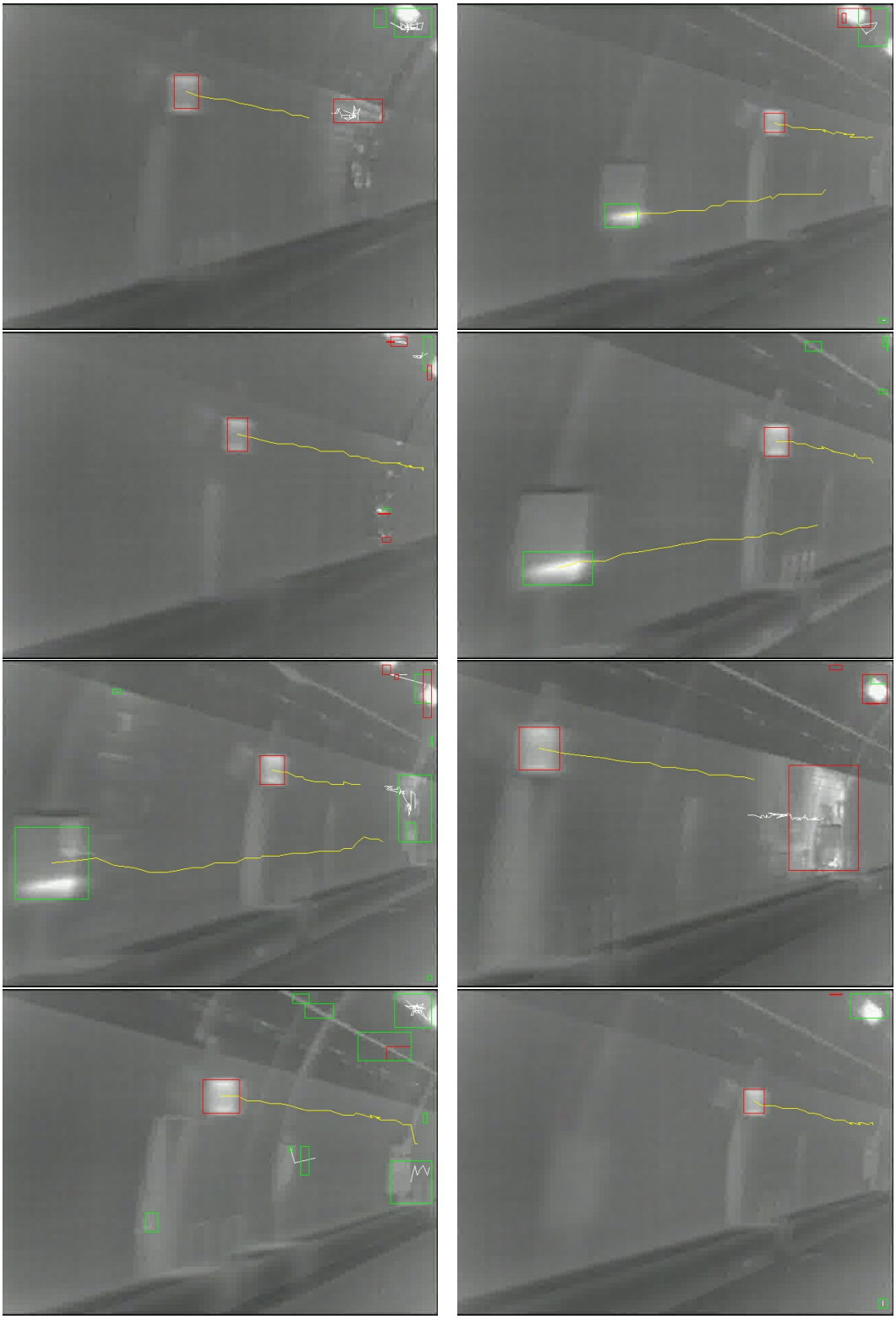


Figure 3.12: Detection results.

better than the first experiment, because the sensed images are much clearer, and also because this more effective training of the Adaboost machine. And in the second experiment, the efficiency has its root in better computer.

The results on the trajectories of decisions are also evaluated. When one trajectory ends, if its length is larger than 15, and over 80% of the decisions it connects are positive, it is considered as positive. The method correctly detects all the 22 emergency telephone indicators with no false alarms. The detection rate is 100%, and the false alarm rate is 0%, as shown in Table 3.4.

total number	22
correctly labeled	22
miss detections	0
false alarms	0
detection rate	100%
false alarm rate	0%

Table 3.4: Final detection rate and false alarm rate.

3.6 Experimental results on data collected by ordinary cameras

The method is extended at the first step to deal with data collected by ordinary cameras. The task is to detect pedestrians and bicycle riders. SURF [Bay et al., 2008] is used as keypoint detector and descriptor.

A new keypoint verification step is proposed. Keypoints from training examples are clustered using k-means as a mixture model. Gaussian distribution is assumed for each cluster, and the variances are also estimated, which will be used as criteria for keypoint verification. Different ks are used to generate the mixture model, and the performance is evaluated in Figure 3.14. An ensemble model is proposed, which never performs worst. All results of k-means with different parameters are summarised to produce the results of ensemble model.

As shown in Figure 3.13, keypoint detection step is performing good in detect keypoints belonging to the target objects. The performance of keypoint verification is not good, as shown in Figure 3.14. And this leads to infeasible later steps.

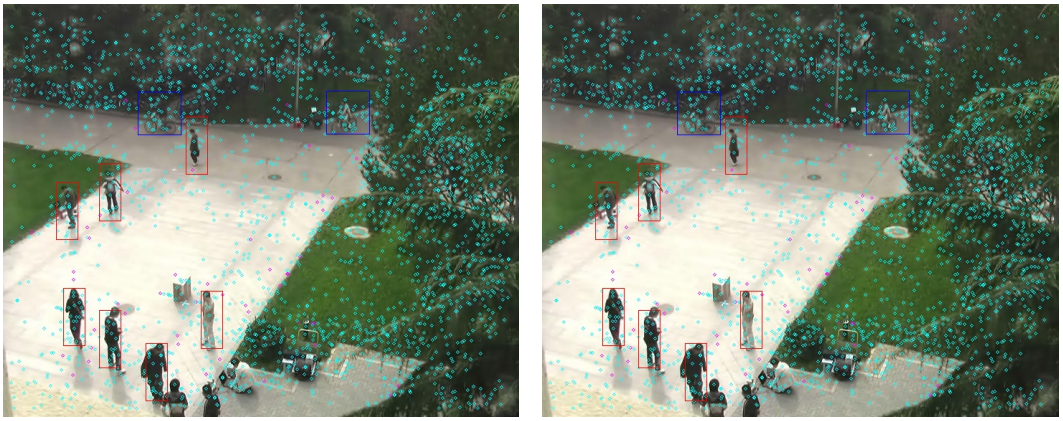


Figure 3.13: Results of keypoint verification. Rectangles mark the manually marked ground-truth boxes. Pink points mark the points labeled as negative in keypoint verification step, others mark keypoints of the positive.

The failure of this method in complex scene is the lacking of descriptive power in its model. Beside appearance information, the positional information is also important when talking about complex scenes.

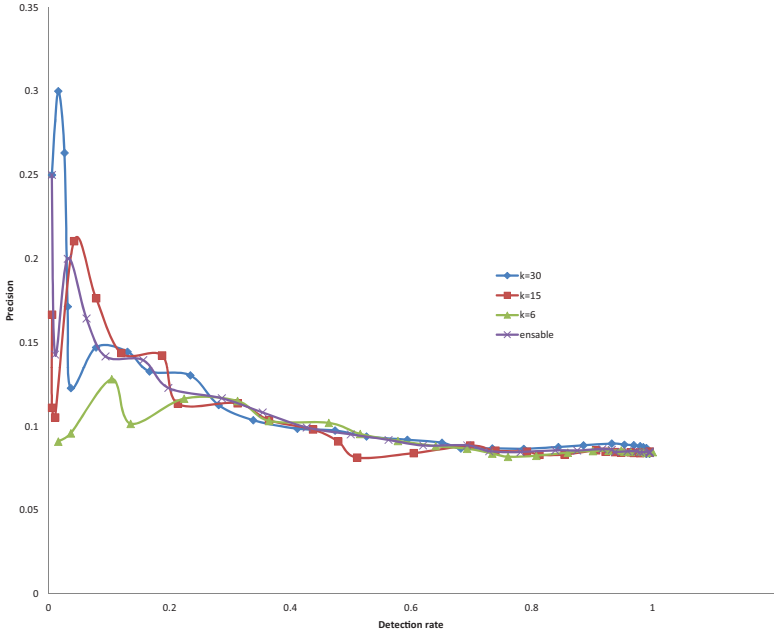


Figure 3.14: Evaluation of keypoint verification. Keypoint verification performance of k-means mixture models with different parameters. Here, k is the main parameter, while the model of ensemble is a mixture of all other k-means methods.

3.7 Chapter Conclusion

This chapter proposes an object detection method, which performs well in simple scenarios by combining appearance and motion information in a very efficient way. The method makes use of appearance and motion information of the target objects in a hierarchical manner. With careful optimization of detection pipeline, the method gives promising results in real time.

Also one main idea of the method is not to consider objects as something only with specific appearance patterns, but as something with both specific appearance and motion patterns. Another reason for the performance during the second experiment, is that make dangerous decisions later, when more information are available. While using both appearance and motion information results in a detection rate of about 99%, combing all the information connected by a trajectory results in a detection rate of 100%.

Though its performance under complex scenarios is not promising, it can still be used as a unit in positioning systems in tunnel environment.

Chapter 4

Grouping of Object Parts by Motion for Detection

4.1 Introduction

The reason why the method proposed in the Chapter 3 fails to work well in complex scenario, while performs well in detecting simple objects is that its model is too simple. Objects, especially pedestrians are actually structured, and different part will have different appearance.

Effective video-based detection methods are of great importance, and this chapter proposes a method to localize and label objects in complex scenarios.

The method is able to detect pedestrians and bicycle riders in complex scene. The method is inspired by the common fate principle, which is a mechanism of visual perception in human beings, and which states tokens moving or functioning in a similar manner tend to be perceived as one unit. This method embeds the principle in an Implicit Shape Model (ISM). In this method, keypoint-based object parts are firstly detected and then grouped by their motion patterns. Based on the grouping results, when the object parts vote for object centers and labels, each of the votes belonging to the same object part is assigned a weight according to its consistence with the votes of other object parts in the same motion group. Afterwards, the peaks which correspond to detection hypotheses on the Hough image formed by summing up all weighted votes become easier to find. Thus this method performs better in both position and label

estimations. Experiments show the effectiveness of this method in terms of detection accuracy.

Most state-of-the-art visual detection methods fall into two main categories: sliding-window methods and Hough transform based methods. The methods [Lampert et al., 2008; Yeh et al., 2009] based on a sliding window schema perform detection in a typical machinery way. In these method, decisions of whether a target object exists or not are made for part of or all the sub-images in a test image. Beside the attractive performance and the extendibility of combining various kernels, these methods are favorable also because they consider each object as a whole during detection. However, they share limited aspects with visual perception in human beings, and the efficiency heavily relies on the size of the test images.

The other methods [Felzenszwalb & Huttenlocher, 2005; Fergus et al., 2003; Leibe et al., 2008; Ohba & Ikeuchi, 1997] detect objects based on the generalized Hough transform [Ballard, 1981]. Object parts are detected, and the object parts provides confidence of locations being potential objects' centers. Locations of objects are decided according to the converged confidence. They are favorable for the robustness to partial deformation and easiness of training. To human beings, this kind of methods seems to be more natural. And in this work, we combine a mechanism of visual perception in human with the ISM [Leibe et al., 2008] to demonstrate this natural property.

A typical Hough transform based method contains two steps: training and detection. During training, a codebook of object parts is built from a set of well annotated images. Each code in the codebook contains information about the appearance of the object part, the relative position to the object center, and the class label. Each object part's appearance is given in the form keypoint descriptors [Leibe et al., 2008], image patches [Gall & Lempitsky, 2009; Okada, 2009], or image regions [Gu et al., 2009]. Each code not only encode one object part's appearance, but also its offset to the object center and the class label. while during detection, on each test image, object parts are detected. Then every object part is matched against the codebook, and activates several nearest codes in appearance. The offset and class label encoded in each activated code will act as a vote. All the votes from the object parts are added up to form a Hough image. The peaks of the Hough image are considered as detection hypotheses with the height of each peak as the confidence for the corresponding hypothesis.

Two challenging issues for detection methods are how to separate near objects

and how to separate similar different-class objects. The target objects, in some ITS applications, are pedestrians, bicycle riders, and automobiles. In the schema of sliding window, usually non-maximum suppression is needed for post-processing, and a mechanism in [Lampert et al., 2008] works by excluding from the feature pool the features which belong to each successive found detection response. In Hough transform based methods, a similar mechanism is also employed in [Barinova et al., 2010], however, this effort is after the forming of Hough image. During the forming of a Hough image, two kinds of votes make detection challenging: (1) votes casted by object parts from near objects make the peaks corresponding to different objects mixed up, and (2) votes casted by similar different-class object parts lead to tough decisions on the class label of the peaks. See Figure 4.2(d). Before the forming of Hough images, problems also arise from the pollution of training images' background part to the codebook. During training a very clean codebook can be built with foreground marked, which needs manual efforts. Otherwise, large amount of training examples are needed for the effectiveness of the codebook, and this harms efficiency.

In videos, motion information is also available by simple tracking of object parts. Thus this chapter proposes a method for detection which utilizes both appearance and motion information. The method is based on the common fate principle [Wertheimer, 1938]. The principle is one of the visual perception principles as theorized by gestalt psychologists, and it states for human beings, tokens moving coherently are perceptually grouped. This provides an intuition to group the object parts by their motion patterns, and let them vote afterwards. In this work, the object parts are represented using keypoint descriptors, which are tracked to generate trajectories. The object parts are grouped by the pairwise similarities of their corresponding trajectories. Having the assumption that object parts in the same motion group probably belong to the same object, for each object part, we assign higher weights for the votes of this object parts which are more "agreeable" within the motion group. This results in votes corresponding to true detection responses are more likely to be assigned higher weights. And on a Hough image formed by summing up these weighted votes, the peaks are easier to find as shown in Figure 4.1(d).

By the combination of motion analysis results with the Hough transform framework through assigning different weights to each object part's votes, the proposed method has several appealing properties:

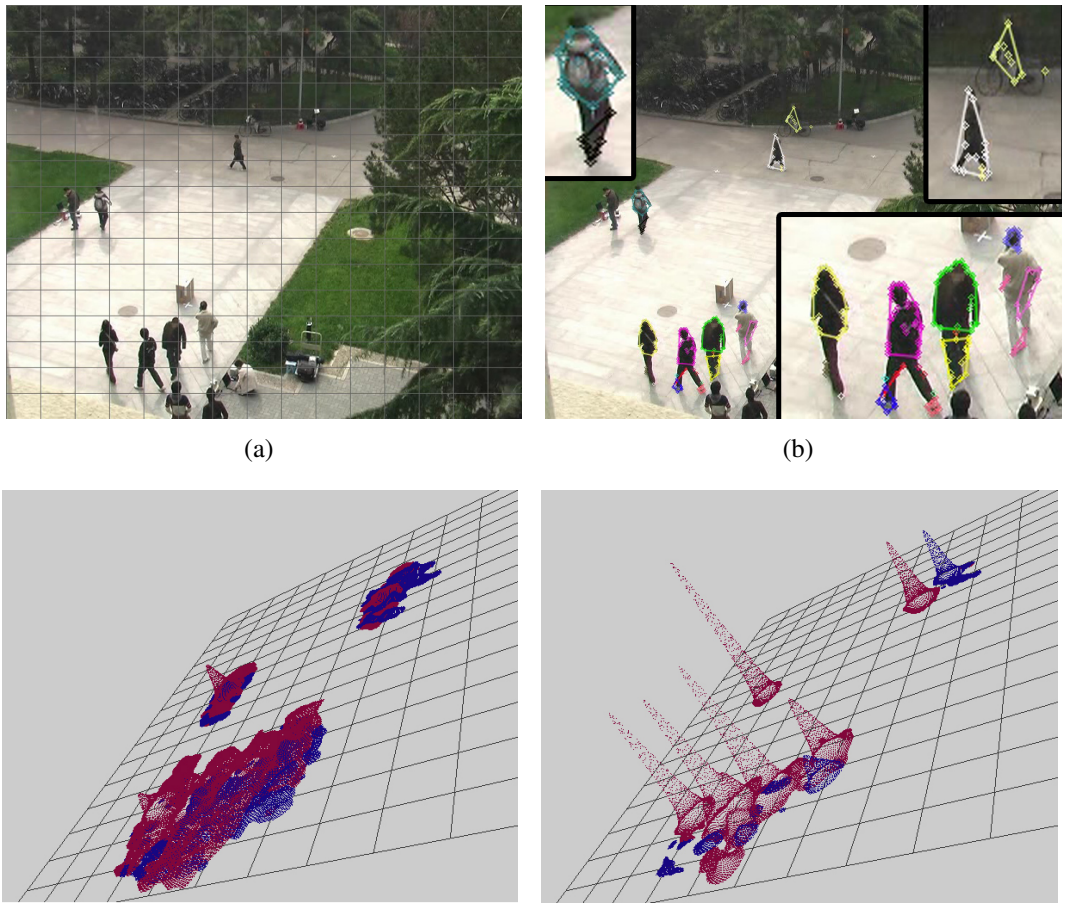


Figure 4.1: Merit of the proposed method. (a) Original image. (b) Motion grouping results. Some parts are enlarged to show details. (c) Original Hough image. (d) Hough image formed using this method. The grids in (c) and (d) correspond to the grids in(a).

- The method's ability to estimate object position and label of multiple objects from different classes. The existence of three types of objects makes the task challenging: near objects, similar different-class objects, and multi-pose same-class objects.
- Its ability to use a codebook trained by images with cluttered backgrounds.
- The framework to combine grouping results of object parts is very general, and has a good expandability.

The remaining chapter is organized as follows. Section 4.2 reviews related work, and section 4.3. Section 4.4 gives formalism of the common fate Hough transform. Section 4.5 describes inference on the formed Hough images. Section 4.6 gives experimental results, and section 4.7 summarises.

4.2 Related Work

This method is most related to object detection methods [Barinova et al., 2010; Leibe et al., 2007, 2008; Leibe & Schiele, 2003; Maji & Malik, 2009; Mikolajczyk et al., 2006] based on the Hough transform framework. Recently, such methods make a lot of progress. The ISM [Leibe et al., 2008; Leibe & Schiele, 2003] is extended by notifying correspondences between the object parts and the hypotheses [Barinova et al., 2010] for the detection of multiple near objects. While in [Gall & Lempitsky, 2009; Maji & Malik, 2009; Okada, 2009] the Hough transform is placed in a discriminative framework for object detection in a way that the codes are assigned different weights by the co-occurrence frequency of their appearance and offset to the object center.

Two Hough transform methods consider the grouping of object parts [Ommer & Malik, 2009; Yarlagadda et al., 2010]. The method in [Ommer & Malik, 2009] deals with scale change. Instead of estimating the scale by local features trained from different scaled examples, the votes are considered as voting lines. By considering the difference of the voted centers, local features are firstly grouped and then vote more consistently for the object center. In [Yarlagadda et al., 2010], the grouping of object parts, the correspondence between object parts and object, and the decisions on detection hypotheses are optimized in the same energy function. For this method, the

problem is that the grouping results don't have meaning or correspond to and real entities.

The work is also related to keypoint clustering methods. In [Estrada et al., 2009], SIFT [Lowe, 2004] is used as key point, color histogram as appearance feature, region covariance [Tuzel et al., 2006] as texture feature, and (x, y) coordinate as spatial feature. Then for one pair key points, a pairwise similarity is calculated based on the generated features. Each point is projected into a new space capturing the pairwise distance by a spectral embedding process. The final result is given by a simple k-means clustering in the new projected spaces of the key points. This is a typical appearance based key point clustering. In [Brostow & Cipolla, 2006], Harris corner [Tomasi & Kanade, 1991] is extracted as key point and motion information is gained by using KLT tracker [Lucas & Kanade, 1981] to trace out the extracted key points on a serial of frames. Two pairwise motion similarity measures are defined on the generated trajectory set, both of which relate to the (x_t, y_t) coordinate, where t is the frame index. One is defined by the largest Euclid distance between every point pair. The other captures the changing quantity between two trajectories by calculating the variance of the Euclid distance among all pairwise point distances. The clustering procedure is made up by making pairwise decisions on whether to merge to smaller clusters in a Minimal Description Language manner.

This work also share features with methods try to couple detection with other problem, and then solve them together. In [Leibe et al., 2007], a very general *Minimum Description Length*(MDL) framework for coupling tracking and detection is given. And [Zhang et al., 2008] re-detects detection responses missed in the detection procedure from tracking results. In [Gould et al., 2009], region semantic label associated with geometry information is estimated in the same energy framework whose parameters are learned together. Besides semantic and geometric information, other information is also estimated in the same framework. The advantage of the method is given by that the learned parameters well encoded the relationship between different information. In [Isukapalli & Greiner, 2003], a interpretation policy is used to specify when to apply which imaging operator, to which portion of the image during every stage of interpretation. The interpretation policy is defined using a dynamic programming schema with considering the cost and the gained information of the operator.

The work is also related to object detection methods by trajectories [Brostow &

Cipolla, 2006; Brox & Malik, 2010], methods weighting features [Yang et al., 2009], methods dealing with codebook noise [Mohottala et al., 2009], methods proposing features combining temporal information [Kläser et al., 2008], and methods which integrate temporal information [Wojek et al., 2010]. Also the mechanisms of visual cortex also supports the proposed method [Sincich & Horton, 2005].

4.3 Application Background

In ITS areas, detection methods using cameras can be used for navigation, safe driving, surveillance, and sustaining results from other sensors. In traditional ITS applications, vehicles are main targets. Currently pedestrians are also considered as important subjects of ITS applications, and bicycles also become very popular for environmental and economical reasons. In Japan, the number of traffic accidents among bicycles and pedestrians is very large. Thus the method proposed in this chapter can be used for detecting freely moving bicycle riders and pedestrians from the data collected by a camera which keeps them under surveillance from the top. These situations can be observed in parks, university campuses, station squares, tourist spots, etc. Here, the techniques from the area of computer vision for detection under surveillance scenarios is focused on.

4.4 Common Fate Hough Transform

Probabilistic standpoints are very appealing, because of inference easiness. However, as pointed in [Lehmann et al., 2011], placing the ISM in a probabilistic framework is not satisfactory. Especially, describing weights of the votes as priors does not make sense. Hough transform can be simply considered as transformation from a set of object parts, $\{\mathbf{e}\}$, to a confidence space of object hypotheses, $C(\mathbf{x}, l)$. And $\mathbf{x}$ is the coordinate of the object center, while l the label. Terms described as priors of the votes in the ISM are actually weights, and the likelihood terms are actually blurring functions to convert discrete votes into continuous space. Then this section describes how a Hough image for estimation of object centers and labels is formed from object parts observed on an image.

Let $\mathbf{e}$ denote an object part observed on the current image. The appearance of $\mathbf{e}$ is matched against the codebook, and $\mathbf{e}$ activates N best matched codes from the trained codebook. Each code contains the appearance, its offset to the object center, and the class label. According to the N matched codes, $\mathbf{e}$ casts N votes. Each vote $V_{\mathbf{e}}$ is about the object center that generates $\mathbf{e}$. The position of the object center casted by a vote, V, is denoted by $\mathbf{x}_V$, while the class label by l_V. Based on the N votes of $\mathbf{e}$, the confidence that a position $\tilde{\mathbf{x}}$ is the center of an object with class label $\tilde{l}$ is given by,

$$C(\tilde{\mathbf{x}}, \tilde{l}; \mathbf{e}) = \sum_{i=1}^{N} B(\tilde{\mathbf{x}}, \tilde{l}; V_{\mathbf{e}}^i) w(V_{\mathbf{e}}^i) \; . \tag{4.1}$$

Here $B(\tilde{\mathbf{x}}, \tilde{l}; V_{\mathbf{e}}^i)$ is the blurring function. And $w(V_{\mathbf{e}}^i)$ is the weight of $V_{\mathbf{e}}^i$.

The idea of the proposed method is that, the weight term, $w(V_{\mathbf{e}}^i)$, is defined by the motion grouping results of all the object parts.

The blurring function is defined as,

$$B(\tilde{\mathbf{x}}, \tilde{l}; V) = \begin{cases} 0 & \text{if } l_V \neq \tilde{l} \text{ or } |\tilde{\mathbf{x}} - \mathbf{x}_V| > d \\ G(\tilde{\mathbf{x}}; \mathbf{x}_V, \sigma) & \text{otherwise} \end{cases} \; . \tag{4.2}$$

Here $G(\tilde{\mathbf{x}}; \mathbf{x}_V, \sigma)$ is a Gaussian function that fixes the spatial gap between $\tilde{\mathbf{x}}$ and $\mathbf{x}_{\mathbf{V}}$.

Let M be the total number of object parts on the image, then by summing up over all the object parts, the confidence of $\tilde{\mathbf{x}}$ being the center of a $\tilde{l}$-class object is given by,

$$\begin{aligned} C(\tilde{\mathbf{x}}, \tilde{l}) &= \sum_{j=1}^{M} C(\tilde{\mathbf{x}}, \tilde{l}; \mathbf{e}_j) w(\mathbf{e}_j) \\ &= \sum_{j=1}^{M} \sum_{i=1}^{N} B(\tilde{\mathbf{x}}, \tilde{l}; V_{\mathbf{e}_j}^i) w(V_{\mathbf{e}_j}^i) w(\mathbf{e}_j) \; . \end{aligned} \tag{4.3}$$

Here, a uniform weight is assumed for each object part, and $w(\mathbf{e}_j) = \frac{1}{M}$. Then by considering $C(\tilde{\mathbf{x}}, \tilde{l})$ as the evaluation score of the Hough space $(\tilde{\mathbf{x}}, \tilde{l})$, the task of estimating object centers and labels converts to finding and then validating the local maxima of the Hough image.

4.4.1 Common Fate Weights

To meet the challenges of separating near objects, separating similar different-class objects, and using a noisy codebook, different weights are assigned to the votes of each object part by considering the motion grouping results of the object parts. In this sub-section, when given some grouping results, how the results are combined into a Hough transform framework is introduced.

Let $\gamma = \{\mathbf{g}\}$ denote the grouping results, where $\mathbf{g}$ is a group of object parts, and assume $\mathbf{e}_m \in \mathbf{g}$ and $\mathbf{e}_n \in \mathbf{g}$. Those votes of $\mathbf{e_m}$ which are more "agreeable" by the votes of the other objects in $\mathbf{g}$ are assigned larger weights.

Towards this end, the relationship between the votes of $\mathbf{e_m}$ and the votes of $\mathbf{e_n}$ needs to be given in advance. This relationship is named support. The support from $V_{\mathbf{e}_n}$ to $V_{\mathbf{e}_m}$ is defined by that based on $V_{\mathbf{e}_n}$, the confidence $V_{\mathbf{e}_m}$'s voted center is correct, as,

$$S(V_{\mathbf{e}_n} \rightarrow V_{\mathbf{e}_m}) = B(\mathbf{x}_{V_{\mathbf{e}_m}}, l_{V_{\mathbf{e}_m}}; V_{\mathbf{e}_n}) \, , n \neq m \, .$$

Here $B(\mathbf{x}_{V_{\mathbf{e}_m}}, l_{V_{\mathbf{e}_m}}; V_{\mathbf{e}_n})$ is defined in (4.2). This measures the coherence of the two votes from different object parts.

Then, the support from $\mathbf{e}_n$ to $V_{\mathbf{e}_m}$ is defined by that based on $\mathbf{e}_n$, the confidence that $V_{\mathbf{e}_m}$'s voted center is correct, as,

$$\begin{aligned} S(\mathbf{e}_n \rightarrow V_{\mathbf{e}_m}) &= C(\mathbf{x}_{V_{\mathbf{e}_m}}, l_{V_{\mathbf{e}_m}}; \mathbf{e}_n) \\ &= \sum_{i=1}^{N} S(V^i_{\mathbf{e}_n} \rightarrow V_{\mathbf{e}_m}) w(V^i_{\mathbf{e}_n}) \, , n \neq m \, . \end{aligned}$$

And the support from $\mathbf{g}$ to $V_{\mathbf{e}_m}$ is defined by the confidence that $V_{\mathbf{e}_m}$'s voted center is correct based on the votes of all the other object parts but its belonging object part in $\mathbf{g}$, as,

$$\begin{aligned} S(\mathbf{g} \rightarrow V_{\mathbf{e}_m}) &= \sum_{\mathbf{e}_i \in \mathbf{g} - \{\mathbf{e}_m\}} C(\mathbf{x}_{V_{\mathbf{e}_i}}, l_{V_{\mathbf{e}_m}}; \mathbf{e}_i) w(\mathbf{e}_i) \\ &= \frac{1}{M} \sum_{\mathbf{e}_i \in \mathbf{g} - \{\mathbf{e}_m\}} S(\mathbf{e}_i \rightarrow V_{\mathbf{e}_m}) \, . \end{aligned}$$

By assuming all object parts in the same motion group are from the same object, which means motion grouping gives good results, the estimations for center position

and class label given by every object part shall be consistent with that given by the motion group. Thus for a particular vote of $\mathbf{e}_m$, i.e., $\tilde{V}_{\mathbf{e}_m}$, a weight is assigned to it by considering its consistence with $\mathbf{g}$ and the consistence of $\mathbf{e}_m$'s other votes with $\mathbf{g}$, as:

$$
\begin{aligned}
w(\tilde{V}_{\mathbf{e}_m}) &= \frac{S(\mathbf{g} \to \tilde{V}_{\mathbf{e}_m}) + \frac{\Delta}{N}}{\sum_{i=1}^{N} S(\mathbf{g} \to V^i_{\mathbf{e}_m}) + \Delta} \\
&= \frac{\sum_{\mathbf{e}_j \in \mathbf{g}-\{\mathbf{e}_m\}} \sum_{k=1}^{N} S(V^k_{\mathbf{e}_j} \to \tilde{V}_{\mathbf{e}_m}) w(V^k_{\mathbf{e}_j}) + \frac{M\Delta}{N}}{\sum_{i=1}^{N} \sum_{\mathbf{e}_j \in \mathbf{g}-\{\mathbf{e}_m\}} \sum_{k=1}^{N} S(V^k_{\mathbf{e}_j} \to V^i_{\mathbf{e}_m}) w(V^k_{\mathbf{e}_j}) + M\Delta} .
\end{aligned}
\tag{4.4}
$$

Here, Δ is a small constant for preventing zeros. Notice, $w(\tilde{V}_{\mathbf{e}_m})$ is defined using $w(V^k_{\mathbf{e}_j})$, the weights of the votes of the other object parts in $\mathbf{g}$. In order to give $w(\tilde{V}_{\mathbf{e}_m})$, uniform weights are firstly assigned to the votes of each object part in $\mathbf{g}$, i.e., $w(V^k_{\mathbf{e}_j}) = \frac{1}{N}$. Then new weights are calculated based on the uniformly assigned weights. The weights of votes to form the Hough image are weights converged in iterations.

The grouping result $\gamma = \{\mathbf{g}\}$, can be replaced by grouping results based on other information, while this method utilizes motion to group the voting elements. The manner of extending the Hough transform is very general, and the extended Hough transform with motion grouping results is called the common fate Hough transform. The votes given by the best matched codes and the votes with higher defined weights are shown in Figure 4.2.

(a) (b)

(c) (d)

Figure 4.2: Effect of the proposed weight. (a) Motion groups, different colors mark different motion groups. (b) Voted centers given by the 7 best matched codes. (c) Voted centers with the highest defined weights. (d) Voted centers with weights higher than a threshold.

4.4.2 Motion Grouping

In this subsection how to group the object parts by their motion patterns is introduced. Basically, the object parts are tracked, and clustered by their motion patterns. The object parts are tracked through frames before and after the current frame to generate trajectories. Then the object parts are grouped by their corresponding trajectories' pairwise motion similarity.

The object parts in this method are in the form of keypoint descriptors. The Harris Corner [Harris & Stephens, 1988] feature is chosen, for robustness, to represent each object part, while for appearance, the region covariance [Tuzel et al., 2006] feature of the image patch around each keypoint is used. The image feature is chosen because of its flexibility to combine multiple channels of information, and also for its capability of handling scale changes in a certain range. For each object part, a trajectory is generated by tracking its corresponding Harris Corner by the KLT tracker [Tomasi & Kanade, 1991]. To group the trajectories, two pairwise similarities are defined.

Let $T_{\mathbf{e}_m}$ and $T_{\mathbf{e}_n}$ denote two trajectories corresponding to $\mathbf{e}_m$ and $\mathbf{e}_n$. The first similarity between two trajectories is defined as,

$$D_1(T_{\mathbf{e}_m}, T_{\mathbf{e}_n}) = \max_{i=1\ldots L}(|\mathbf{x}^i_{T_{\mathbf{e}_m}} - \mathbf{x}^i_{T_{\mathbf{e}_n}}|) \,.$$

Here, i is the frame index, and L is the number of frames which are crossed by both trajectories.

To define the second similarity, the ith directional vector of T is firstly defined as, $\mathbf{d}^i_T = \mathbf{x}^{i+3}_T - \mathbf{x}^i_T$. Let $\mathbf{a}_i = \mathbf{d}^i_{T_{\mathbf{e}_m}}$, $\mathbf{b}_i = \mathbf{d}^i_{T_{\mathbf{e}_n}}$, $a_i = \frac{\mathbf{a}_i \cdot \mathbf{b}_i}{\mathbf{a}_i \cdot \mathbf{a}_i}$, and $b_i = \frac{\mathbf{a}_i \cdot \mathbf{b}_i}{\mathbf{b}_i \cdot \mathbf{b}_i}$. Then the second similarity is defined as,

$$D_2(T_{\mathbf{e}_m}, T_{\mathbf{e}_n}) = \max_{i=1\ldots L-3}(\max(|\mathbf{a}_i - a_i\mathbf{a}_i|, |\mathbf{b}_i - b_i\mathbf{b}_i|)) \,.$$

Before grouping the trajectories, the static points are excluded. The defined D_1 is calculated for all pairs of trajectories, and a minimal spanning tree is then built using the calculated distances. The built mst is split by cutting edges larger than a threshold, D^1_{th}, and this gives a grouping result of the trajectories. For each element in the clustering result, D_2 is used in the same procedure to generate even smaller clusters. This hierarchical procedure ensures that trajectories in the same group have

both small D_1 and D_2. Each trajectory corresponds to an object part, and the grouping results of the trajectories correspond to grouping results of the object parts.

4.4.3 Codebook

For training, Harris corners are extracted from the training images with the object center and the class label annotated. In this method, region covariance is chosen to represent the appearance, which is defined as,

$$\mathbf{r} = \frac{1}{K-1}\sum_{i=1}^{K}(\mathbf{z}_i - \mu)(\mathbf{z}_i - \mu)^T \ .$$

Here, K is the number of pixels in the region, and $\mathbf{z}_i$ is a 7-dimensional vector regarding the (x, y) coordinate of the pixel, while μ is the mean of $\mathbf{z}_i$. And $\mathbf{z}(x, y)$ contains the RGB color of the pixel and the intensity gradients of the pixel, as: $r(x, y)$, $g(x, y)$, $b(x, y)$, $|\frac{\partial I(x,y)}{\partial x}|$, $|\frac{\partial I(x,y)}{\partial y}|$, $|\frac{\partial^2 I(x,y)}{\partial x^2}|$, and $|\frac{\partial^2 I(x,y)}{\partial y^2}|$.

The appearance similarity between $\mathbf{r}_m$ and $\mathbf{r}_n$ is given by,

$$\rho(\mathbf{r}_m, \mathbf{r}_n) = \sqrt{\sum_{i=1}^{7}\ln^2\lambda_i} \ .$$

Here, λ_i is the generalized eigenvalue by solving the generalized eigenvalue problem, $\lambda_i \mathbf{r}_m \mathbf{u}_i = \mathbf{r}_n \mathbf{u}_i, \mathbf{u}_i \neq \mathbf{0}$, with $\mathbf{u}_i$ the eigenvector.

A square image patch around each keypoint is used to represent the appearance of an object part. Six region co- variances are generated for each image patch by using the pixels of the top-left, the top-right, the bottom-left, the bottom-right, the central, and all of the image patch. Then besides the offset and the class label, a code contains six region covariances. All codes from all training images constitute the codebook. When an object part is matched against the codebook, the similarity between the image patch of the object part and a code is defined by the smallest similarity of the corresponding region covariance.

4.5 Detection

After forming the Hough image, the detection hypotheses are validated. Let $\mathbf{h} = \{H\}$ be the points in the Hough space which are evaluated by $C(\mathbf{x}_H, l_H)$ and have $C(\mathbf{x}_H, l_H) > 0$. Inspired by [Barinova et al., 2010], the hypotheses are validated by an optimizing procedure. Let O be the number of the points in $\mathbf{h}$. Let $u_i = 1$ or 0 indicate H_i being a true object center or not. The problem is:

$$\arg\max_{u_i} \prod_{i=1}^{O} C^{u_i}(H_i) \Longleftrightarrow \arg\max_{u_i} \sum_{i=1}^{O} u_i \ln(C(H_i)) \ .$$

Let $v_{ij} = 1$ or 0 indicate $\mathbf{e}_j$ belongs to H_i or not, then

$$\begin{aligned} C(H_i) &= \sum_{j=1}^{M} C(\mathbf{x}_{H_i}, l_{H_i}; \mathbf{e}_j) w(\mathbf{e}_j) \\ &= \frac{1}{M} \sum_{j=1}^{M} v_{ij} C(\mathbf{x}_{H_i}, l_{H_i}; \mathbf{e}_j) \ , \end{aligned}$$

and by assuming one object part belongs to and only belongs to one hypothesis, the problem is,

$$\begin{aligned} &\arg\max_{u_i, v_{ij}} \sum_{i=1}^{O} u_i \ln(\sum_{j=1}^{M} v_{ij} C(\mathbf{x}_{H_i}, l_{H_i}; \mathbf{e}_j)) \\ s.t. :\ & u_i = 0 \text{ or } u_i = 1, \forall\ i; \\ & v_{ij} = 0 \text{ or } v_{ij} = 1, \forall\ i, \forall\ j; \\ & \sum_{i=1}^{O} v_{ij} = 1, \forall\ j; \\ & \sum_{j=1}^{M} v_{ij} \leq u_i, \forall\ i \ . \end{aligned}$$

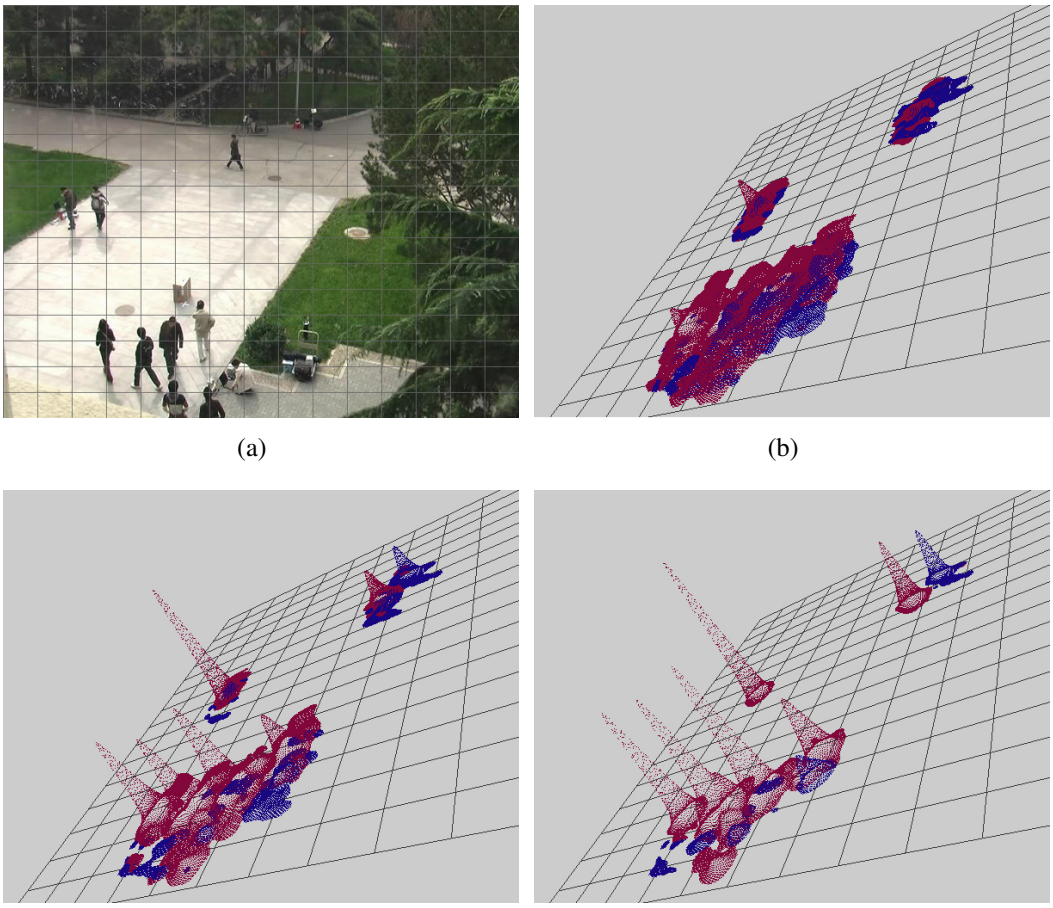
Following [Barinova et al., 2010], the optimal result for the problem is given by greedy maximization. As described in Algorithm 4.1, the largest local maximum of all the local maxima is chosen to be the center of a true object and then the object parts belonging to the chosen object center are excluded from the object part set. A new Hough image where new objects are found is formed using the remaining object

parts. And this procedure ends when the object part set is empty or the confidence of the chosen object is lower than a threshold.

Algorithm 4.1 Greedy Maximization

Let ε be the set of object parts, C_{th} be the low confidence threshold to accept detection responses, and $\hat{\mathbf{h}}$ be the local maxima of $\mathbf{h}$

while $\varepsilon \neq \emptyset$ **do**
  Form $\mathbf{h}$ with ε
  Generate $\hat{\mathbf{h}}$ and select $H_i \in \hat{\mathbf{h}}$ with the largest $C(\mathbf{x}_{H_i}, l_{H_i})$
  if $C(\mathbf{x}_{H_i}, l_{H_i}) >= C_{th}$ **then**
    for $\mathbf{e}_j \in \varepsilon$ **do**
      if $\forall H' \in \hat{\mathbf{h}}, C(\mathbf{x}_{H_i}, l_{H_i}|\mathbf{e}_j) >= C(\mathbf{x}_{H'}, l_{H'}|\mathbf{e}_j)$ **then**
        $\varepsilon \leftarrow \varepsilon - \{\mathbf{e}_j\}$
      end if
    end for
  else
    $\varepsilon \leftarrow \emptyset$
  end if
end while
return $\{H_i\}$



(a) (b)

(c) (d)

Figure 4.3: Example Hough images. Grids in (b), (c), and (d) correspond to the grids on the image coordinate. Red indicates pedestrians, while blue indicates bicycle riders. Hough image of (b) is formed by votes with uniform priors, Hough image of (c) is formed by votes with priors after 5 iterations, and Hough image in (d) is formed with converged priors.

4.6 Experimental Results

In experiments, improvement of the method is verified in terms of detection accuracy. The method is tested on the P-campus dataset with [Barinova et al., 2010] as benchmark, and then tested on a dataset of several animals.

4.6.1 Campus-scene Detection

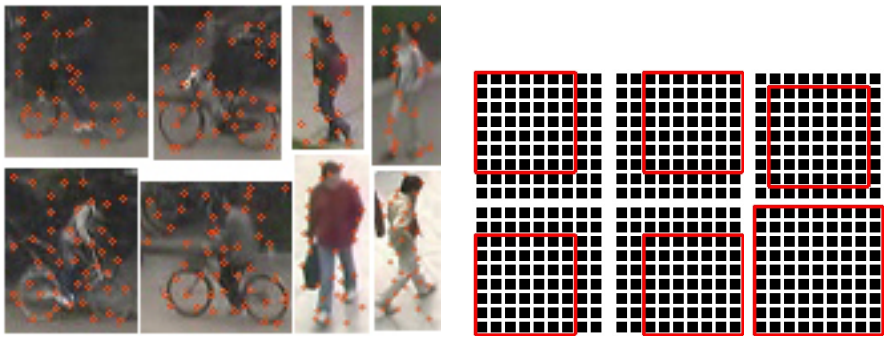
Dataset The P-campus dataset contains two primary classes of foreground objects: pedestrians and bicycle riders. The frame size is 720×576. Among all the 401 continuous frames, 633 different-class ground truth bounding boxes are annotated on 79 frames. In this dataset, pedestrians and bicycle riders have in common the upper human body, and pedestrians appear in front, back, and side views.

Implementation Settings For training, 52 images of bicycle riders and 171 images of pedestrians are randomly selected. Harris corners are generated on the image, examples are given in Figure 4.4(a). For appearance, six region covariances are generated for each keypoint using the 9×9 image patch around it as shown in 4.4(b). The appearance, the offset to the image (object) center, and the label of the training image are encoded into a code, and the code is inserted into a codebook. The final codebook contains 5502 codes.

For motion grouping, each keypoint is tracked through 10 frames before and through 10 frames after the current frame. The similarity of two 21-point trajectories is defined using only the frames crossed by both trajectories. To set the two thresholds for motion grouping, D_1 and D_2 are measured for keypoint pairs of different objects. D_{th}^1 is set that it is larger than only 10% of the measured D_1s, and so is D_{th}^2. By doing so, keypoints belonging to different objects are not likely to be grouped together. So that in one motion group, the keypoints are very likely to belong to the same object, as shown in Figure 4.5.

To form the Hough image, 35 best matched codes are chosen from the codebook for each object part. In (4.3), d and σ need to be given. The precision-recall curves are based on σ, while d is set to be 10. Here σ is the most important parameter.

Comparisons For comparison, detection is done on the Hough images formed with and without motion grouping results. The same codebook and the same parameter



(a) (b)

Figure 4.4: (a) Training images. Note some keypoints fall on the background. (b) The manner how a 9×9 image patch is used to generate six region covariances, and red rectangles indicate the pixels used for each covariance.

Figure 4.5: Motion grouping results.

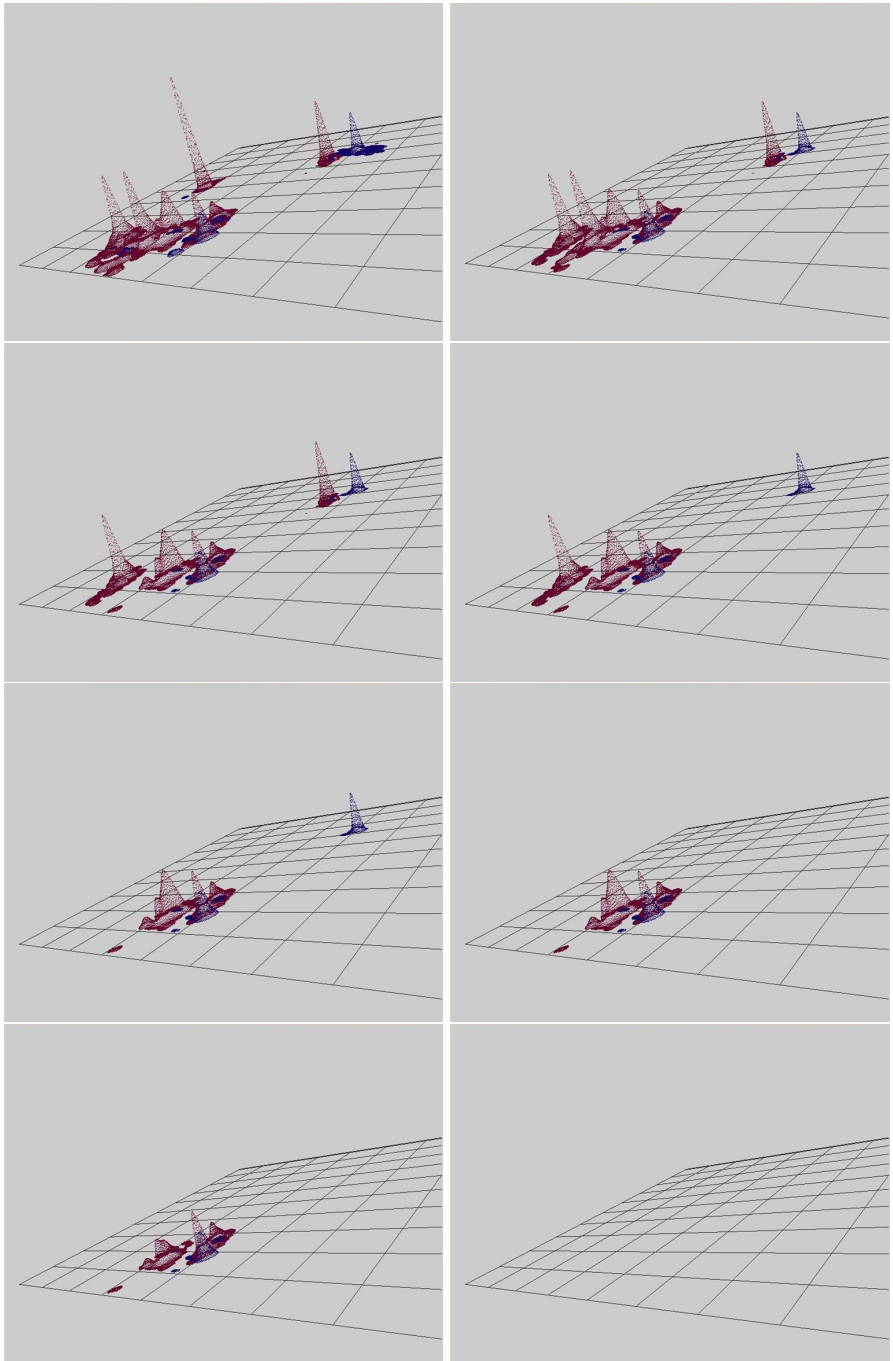
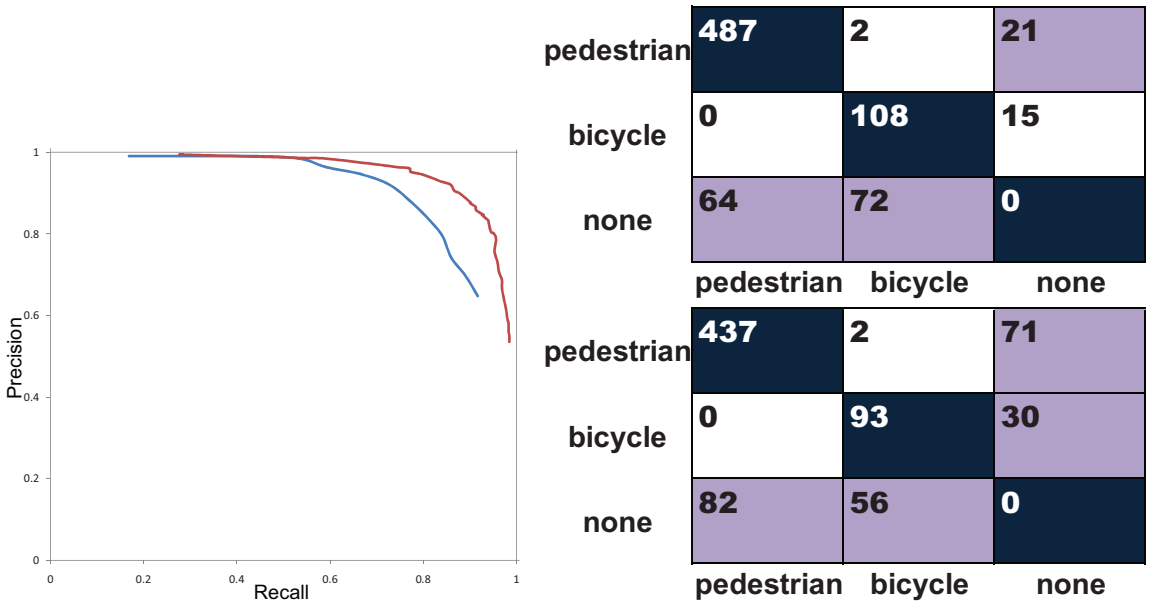


Figure 4.6: Inference procedure.



(a) (b)

Figure 4.7: (a) Precision-recall curves (red: the proposed method, blue: the benchmark method). (b) Confusion matrices (upper: the proposed method, down: the benchmark method).

settings are used for forming and searching over both Hough images. The votes of each object part are assigned uniform weights in the benchmark method, while weights defined in (4.4) are assigned in the proposed method.

The precision-recall curves are shown in Figure 4.7(a). An object is considered as correctly detected only if the distance from the ground truth to it is less than 10 pixels. In Figure 4.7(a), the correctly positioned but wrongly labeled objects are considered as true positives, aiming at verifying the positioning ability of the proposed method.

The confusion matrices are given in Figure 4.7(b). For clarity of the comparisons, the proposed method is compared with the benchmark method when they have nearly equal number of false alarms. To evaluate the labeling ability, a class of "none" to represent missed detections and false alarms is manually added. For example, in Figure 4.7(b), 487 pedestrian instances are correctly positioned and labeled by the proposed method, 2 are wrongly labeled to be bicycle riders, and 21 are miss-detected. More results are shown in Figure 4.8.

Figure 4.8: Results. Red rectangles and blue rectangles mark correctly detected pedestrians and bicycle riders. Yellow rectangles mark missed detections. White rectangles mark correctly detected but not correctly labeled objects. Green rectangles mark false alarms. Black rectangles mark static objects, which are beyond the verification for the method.

4.6.2 Wild-scene Detection

Dataset In order to show that this method can be used for general purposes, we test this method on complicated scenes, especially, complicated background. Even in these cases, this method works well, which shows robustness of this method. A mini dataset is built upon leopards and tigers of the family Felidae. Especially, the image feature used by this method belongs to the type of texture, and texture from different positions of the leopards are almost the same. The dataset contains 6 video clips of 9 leopards and 4 tigers. The frame size is 640×480. Both the animals are in the side view.

Implementation settings Most implementation settings are the same with the settings for campus object detection. For training, 5 leopards and 2 tigers are used. The size of the image patch around each keypoint is 27×27.

Comparisons In Figure 4.9, the motion grouping results and how the voted centers are affected are given. Since parts from different positions of the leopard are very similar, the true center of a leopard is difficult to find from the voted centers of the object parts. In Figure 4.10, example Hough images are given to show the merit of the proposed prior by the ability to detect leopards. In Figure 4.11, the detection results are given. The proposed method successfully localizes and labels all the leopards and tigers, while the benchmark method miss-detects three leopards.

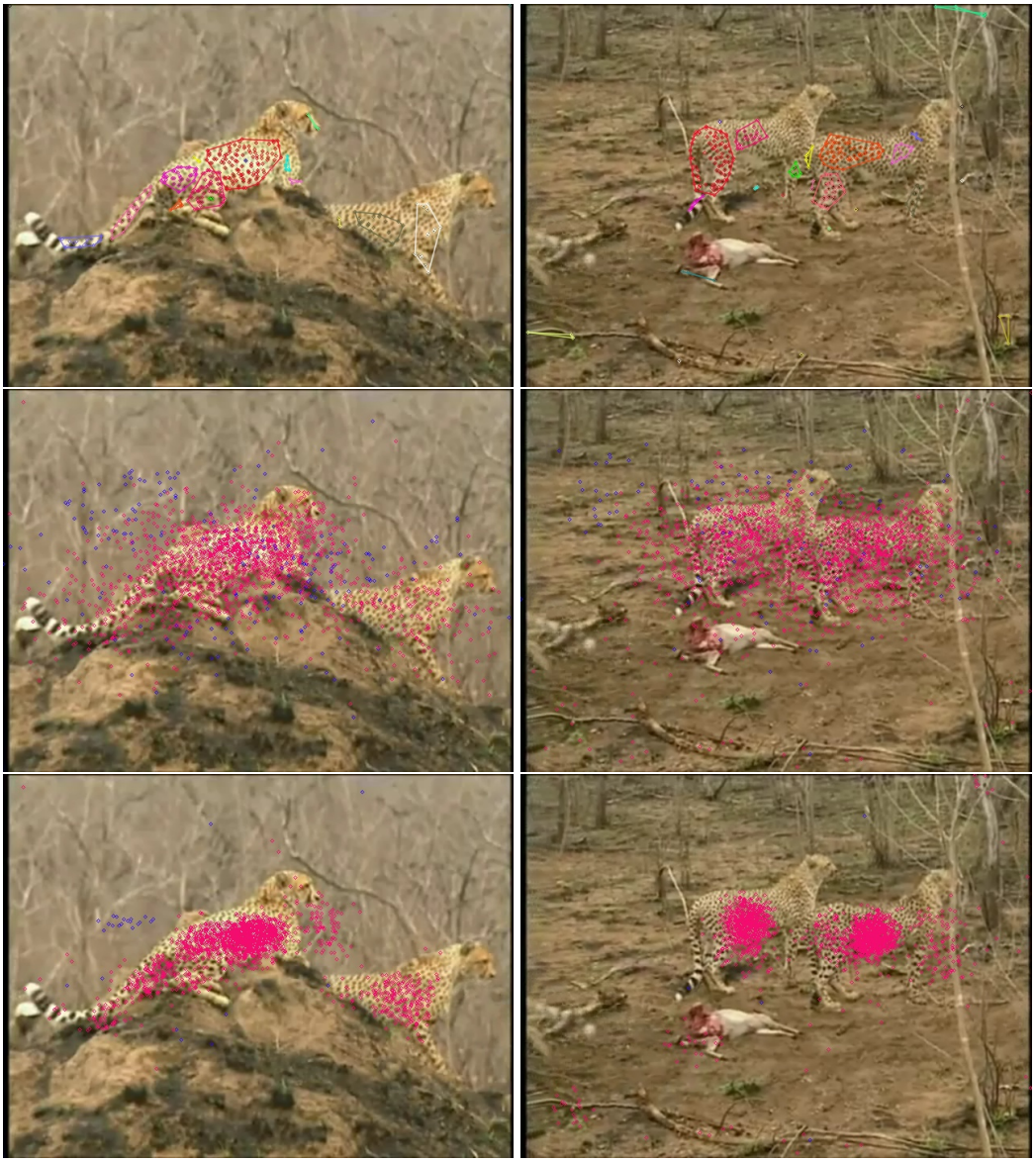


Figure 4.9: Effect of the proposed weight assignment. Red circles are voted center for leopards, while blue ones are voted centers for tigers. On the top are the motion grouping results. In the middle are the voted centers according to the best matched codes. On the bottom are the voted centers voted by votes with highest weights.

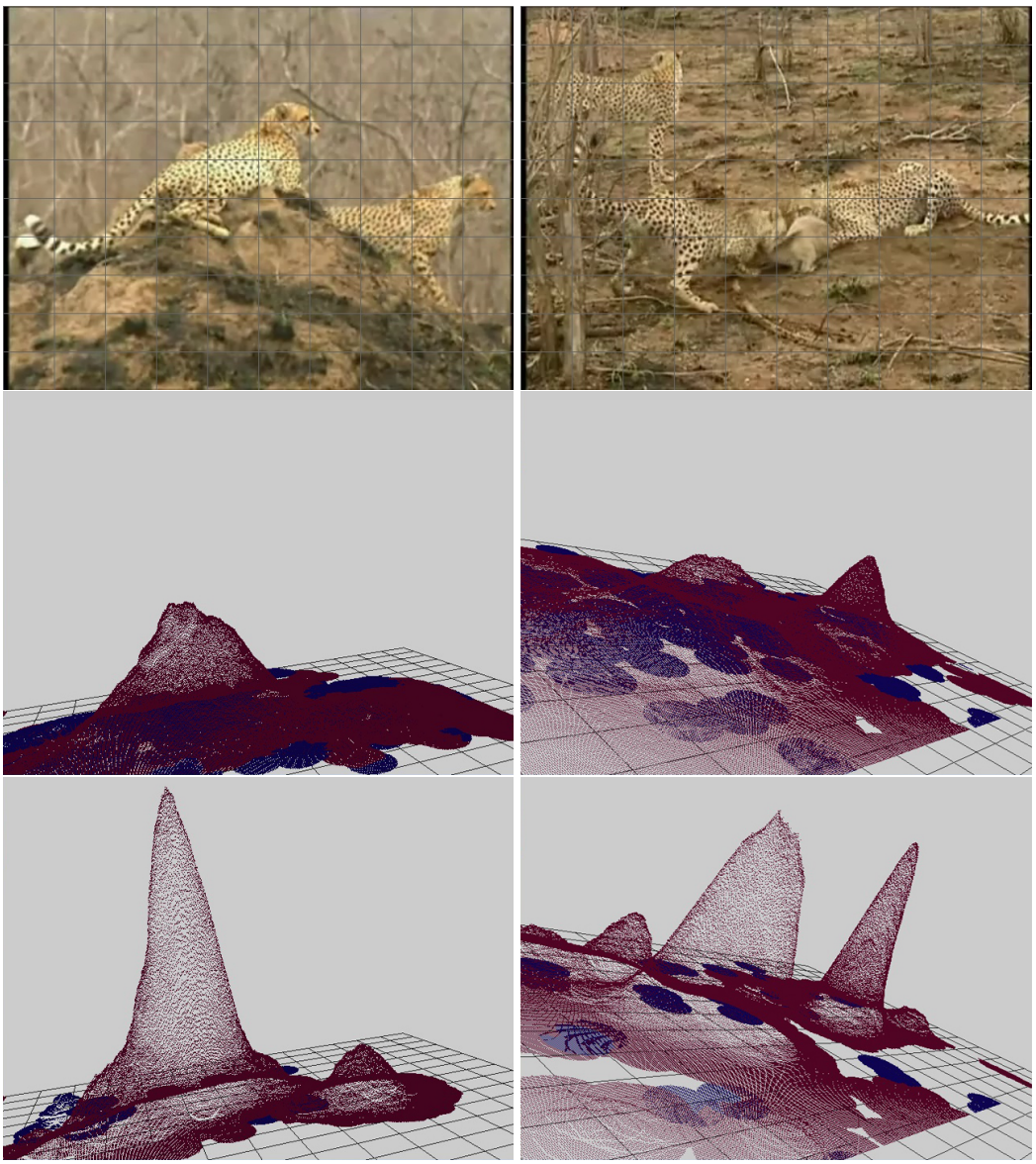


Figure 4.10: Example Hough images. On the top are the original images. In the middle are the Hough images formed by votes with uniform priors. On the bottom are the Hough images formed by votes with the proposed weights. Red indicates leopards, and blue indicates tigers. Note for the two leopards, there is no peak corresponding to the right one on the benchmark Hough image. For the three leopards, there is also no peak corresponding to the leopard in behind on the benchmark Hough image.

Figure 4.11: Results. Red crosses mark the centers for leopards and blue crosses mark the centers for tigers.

4.7 Chapter Conclusion

The computational ability of human beings is limited, while the ability to detect is far beyond machines. Thus, it is very possible that this detection ability benefits from multiple perceptual mechanisms. By using one of these mechanisms, we propose a detection method. By embedding motion grouping results into the voting schema of hough transform, the method is capable to distinguish near objects' positions, to distinguish similar objects' labels, and to maintain detection rate with a noisy codebook. The success of this method further demonstrate the advancement of perceptual mechanisms in human beings. And the success of this method will help with detection methods in ITS areas.

Chapter 5

Pyramid Match Score for Detection

5.1 Introduction

Bag-of-features [Jégou et al., 2010] schema can be considered as the watershed between traditional and modern detection methods. Instead of considering each target object as a collection of raw optical elements, i.e., pixels, the schema tries to consider each object as a set of semantic elements or so-called object parts which are usually some strong local image features. Then one visual object is said to be a target object if it possesses some certain local features of certain numbers, while does not contain other certain local features of certain numbers. While this is quite straightforward, the very precious information encoded in local features' relative positions is left over. Following some pioneering ideas [Ferrari et al., 2007; Lazebnik et al., 2006], this chapter proposes a detection method which combines spatial and visual information of local image features in a way pursuing both efficiency and effectiveness.

The results of Chapter 4 is promising on the two experimental datasets, however, the efficiency is not good due to the employment of Hough transform framework. Besides, inferring object status in a bottom-up manner fails to capture global information of each target object from the very beginning. And this is also why recently the detection results of Hough transform based methods need refinement by discriminative methods in order to be competitive. Still the way how to use spatial information of local features is very illuminate.

Just as said in [Lehmann et al., 2011], Hough transform based methods and sliding-

window methods are the two sides of the same coin. The method proposed in this chapter calculates confidence of a target object class for each sub-window in an image. Instead of considering each object as a collection of visual patterns (appearance of local features), the method considers each object as a set of visual-spatial patterns. One object is considered as a set of points. Each point is a digital vector, with the last two dimensions the relative $x-$ and $y-$ coordinates to object center, and SIFT after principle component analysis as the remaining dimensions. The training procedure is about collecting all such visual-spatial points into a point set, which acts as a super template. During detection, each sub-image is considered as a point set, and it is matched against the super template. The confidence is then the match score.

The key to this method is how to define a matric or match score for two point sets. Here pyramid matching procedure is employed, not only for efficiency, but also for combining visual and spatial information from local features in an effective manner. The visual-spatial space is divided from fine to coarse. Under a certain dividing parameter, points from the two matching point sets are considered as match if they fall into the same grid, and they are excluded from the respective point sets. The procedure continues till one point set is empty. Then the numbers of matched pairs under each dividing method is counted, and a weighted sum of all these numbers are considered as the match score for two point set, which will be referred to as Pyramid Match Score, or PMS for short. The weights under all dividing methods are learned during training, and also how to divide the visual-spatial space is of great importance.

Obviously, each object is considered as a whole during detection in this method.

The proposed method also has several appealing properties, which include but not limited to:

- Feasibility of sequential/batch training, which will lead to easy deployment in distributed system.
- Space complexity of the model is $O(1)$. Since the capacity of the point set acting as the super template is finite, and the size of the model is limited by the capacity.
- Detection time complexity not related to the size of training examples.

This chapter is organized as follows. Section 5.2 reviews most related work. Section 5.3 propose the training and detecting procedure. Section 5.4 gives experimental

results. Section 5.5 concludes this chapter.

5.2 Related Work

Detection methods still mainly follow the sliding-window schema or share similar structures with Hough transforms. While the focus of the later is to infer about object status by use each online feature as query against a well-trained codebook. These methods fail to consider target objects as a whole at the beginning. The problem of sliding-window is that it often ignores positional information when also following bag of features [Jégou et al., 2010]. In the method of [Lazebnik et al., 2006], positional information is considered in the kernel function. Here a kernel function is usually used in classifiers, which are usually support vector machines, as introduced in [Shawe-Taylor & Cristianini, 2004]. The assumption behind [Lazebnik et al., 2006] is that two images are considered as similar if they possess similar object parts at similar relative positions. Despite of the good theory, its being embedded in support vector machines as kernel function limits the efficiency of this method.

The Bag-of-features [Jégou et al., 2010] schema successfully improves detection performance, while still there are information which are left behind in images. The positional information is not fully made use of, even the method of [Shawe-Taylor & Cristianini, 2004]. While [Fergus et al., 2003] provides a method to model the relationship between object parts, there are two many parameters to estimate in their model, which requires large amount of training data for acceptable performance.

In the method proposed in [Jiang & Yu, 2009], each object is modeled as a graph, when matching each object with another, constrains are made not only between the two objects, but also between different features of the same object. The relationship between elements of the same object is important. However, the inefficiency of this method prevents it from directly being used for object detection, while its performance on matching the same object under different views is promising.

The method in [Liu et al., 2011] instead of building some parametric or non-parametric model, directly maps the labels of similar images in the training images to the current image. In this manner, the descriptive ability of model can be left alone, which in return makes the method robust. However, this kind of methods heavily rely on the manually marked labels in the training dataset, while such labels are very ex-

pensive in human power and computation.

The method in [Fergus et al., 2003] considers both appearance model of object parts, and the relative distance changes between object parts. While giving promising results, there are too many parameters in the model, and training is troublesome when limited training images are available.

The successes of HOG [Dalal & Triggs, 2005] on pedestrians are also because of its ability to encode relative spatial and visual information from each divided cells. Still the flexibility is not enough, and this leads to deformable part model [Felzenszwalb et al., 2010; Ferrari et al., 2007], and its enhanced versions [Felzenszwalb et al., 2010, 2008]. The model will be referred as DPM for short. It is currently employed by most state-of-the-art methods considering appearance information of object parts together with the relative positional information between object parts. In methods following DPM, a root template is used to detect each object as a whole, and HOG feature is usually used. When an potential object is detected, all possible object parts are detected accordingly. Finally, the confidence of the object is given by the confidence of the root object, the sum of confidence of the object parts, and the cost to deploy the object parts within the root object. Latent SVM is employed in the methods for optimization while using representation of complex information. Some methods motivated by DPM try to improve DPM by providing better solution searching strategies [Pedersoli et al., 2011]. Two recent methods [Pirsiavash & Ramanan, 2012; Song et al., 2012] try to find sparse basis of object parts to reduce the number of parameters need estimating. For efficiency, the method in [Dean et al., 2013] replace the dot operator of DPM with start-of-the-art hashing method [Gionis et al., 1999]. There also exists hierarchical extension of DPM [Zhu et al., 2010], and multi-view extension of DPM [López-Sastre et al., 2011].

Advantages of DPM include that it only need to encode the object parts in positive training examples, and that some of its invariants can give promising results in real time. Still DPM heavily rely of the latent SVM, which is trained in an expectation-maximization manner, and this stops it from adopting new training examples. While nowadays, training examples often come sequentially. A very flexible model, which will evolve with training examples is preferred. These evolvements include object part number evolving, evolving of the appearance models of object parts, and evolving of the relative positions of object parts.

The pyramid match score method is most related to methods using [Grauman & Darrell, 2005] or [Shawe-Taylor & Cristianini, 2004] as kernel functions, methods employ Hough transforms, and also the methods proposing efficient solution space searching techniques [Lampert et al., 2009]. The method is also related to efforts trying to encode images [Olshausen & Field, 1996].

5.3 Pyramid Match Score

In this section, firstly the typical procedure of pyramid matching is reviewed, and how a match score between two point sets by using pyramid matching is defined. Then based on the defined matric, how from the training examples, a super template can be learnt and how the super template can be used for object detection in a test image is proposed. In the definition of the matching score, there are parameters very important, finally in this section, how these parameters are estimated is introduced in three sub-sections.

In the remaining content of this chapter, all is, js and ks are local symbols.

5.3.1 Pyramid Matching

The Pyramid Matching method is designed to find the best one-one match, as shown in Figure 5.2, between two point sets in a heuristic manner.

Given two point sets, $S_1 = \{u_1, u_2, ..., u_m\}, u_i \in R^d$ and $S_2 = \{v_1, v_2, ..., v_n\}, v_i \in R^d$, there exists a best one-one matching π^* that minimizes the sum of $L1$ -distances between matched pairs,

$$\pi^* = \arg\min_{\pi} \sum_{u_i \in S_1} ||u_i - v_{\pi(i)}||_1 \ .$$

Here $m \leq n$, and π maps each feature u_i in S_1 to a unique feature $v_{\pi(i)}$ in S_2. There is a 2D example in Figure 5.1

The best matching exists, and can be found by simple brute-force enumeration. In special cases, the Hungarian algorithm [Kuhn, 1955] is also applicable.

Sub-optimal solution can be found by heuristic methods. A very intuitionistic method is to find matched pairs of nearest distance, exclude corresponding points from

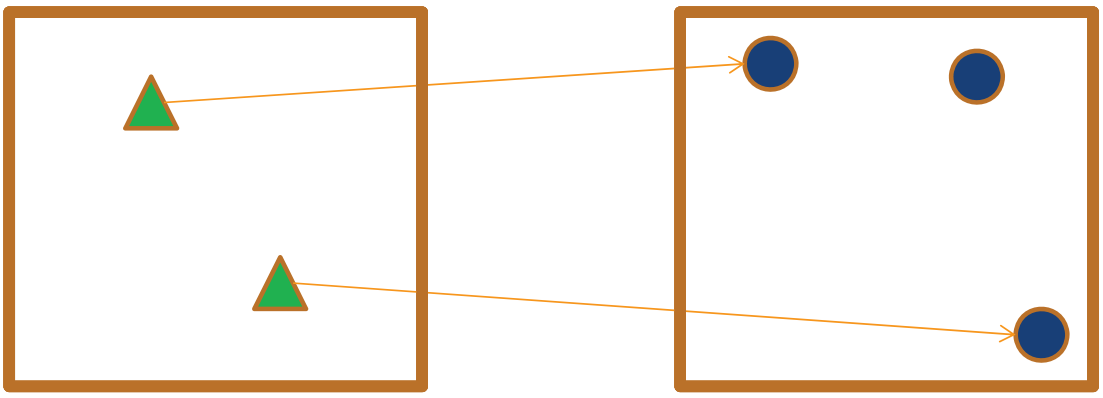


Figure 5.1: A best one-one match problem in 2D space. There are two points in the first point set, three in the second point set. The arrows show correspondence between the two point set.

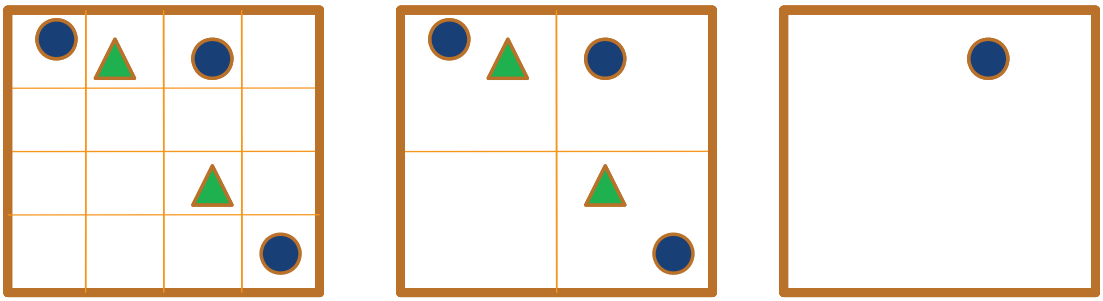


Figure 5.2: Pyramid matching procedure which takes the 2D one-one match problem in Figure 5.1 as an example. The pyramid matching method divides the 2D space from fine to coarse in the 2D space. Notice, the triangle points belong to point set one, and the circle points belong to point set two. In the left, each dimension is divided into 4, results in totally 16 grids, and no matched point pairs are found. In the middle, each dimension is divided into 2, results in totally 4 grids, and two pairs of points belonging to different point sets are found and excluded. In the right, since the matched points belonging to matched point pairs are excluded, then only one point from the second point set is left. So the number of pairs found under all dividing methods are, 0, 2, and 0. The pyramid match score is calculated as a weighted sum of these 0, 2, and 0.

both point sets, and repeat until no matched pair can be found.

The Pyramid Matching method is straightforward. Divide the point space from fine to coarse, find pairs of points from different point sets in the same grid under the current dividing parameter, exclude the matched pairs, and continue this procedure until the smaller point set is empty. The Pyramid Matching method is very efficient, and its time complexity is bounded by $O(dmL)$ [Grauman & Darrell, 2005]. Here d is the number of dimensions in each point set, m is size of the smaller point set, and L is number of dividing methods. In the example of Figure 5.2, L is 3.

In [Grauman & Darrell, 2005], pyramid matching helps to define kernel functions for SVMs. The meaning of pyramid matching is that, it changes how the way to define similarity between two objects. Originally two objects are considered as similar if they both contain certain number of certain object parts, while the idea of one-one match will only favor the objects parts which have corresponding counterparts.

Let $\gamma = \{\mathbf{g_1}, \mathbf{g_2}, ..., \mathbf{g}_L\}$ be an ordered set, which contains all the dividing methods from fine to coarse. Let $N(S_1, S_2; \mathbf{g}_i)$ be the number of matched pairs of points under dividing method $\mathbf{g}_i$. Then the pyramid match score between S_1 and S_2 on γ is defined by,

$$\mathrm{P}(S_1, S_2; \gamma) = \frac{\omega_1 \times N(S_1, S_2; \mathbf{g}_1) + \sum_{i=2}^{L} \omega_i \times (N(S_1, S_2; \mathbf{g}_i) - N(S_1, S_2; \mathbf{g}_{i-1}))}{m}. \tag{5.1}$$

To exactly follow the procedure as shown in Figure 5.2, the definition in 5.1 is rewritten as,

$$\mathrm{P}(S_1, S_2; \gamma) = \frac{\sum_{i=1}^{L} \omega_i \times N(S_1^{(i-1)}, S_2^{(i-1)}; \mathbf{g}_i)}{m}. \tag{5.2}$$

Here, S_1^i and S_2^i represent the point set after excluding the points which are found match after the ith round matching respectively from $S_1^{(i-1)}$ and $S_2^{(i-1)}$. Actually, $S_1^0 = S_1$, and $S_2^0 = S_2$.

The procedure is as follow, 1) given the original S_1 and S_2, find the point pairs which fall into the same grid in the space defined by $\mathbf{g}_1$, 2) exclude the matched points respectively from S_1 and S_2, will lead to S_1^1 and S_2^1, and 3) continue until $i = L$ or one

point set is empty.

Left behind is how to construct the dividing methods in γ and how to define the corresponding weight ω_i for each $\mathbf{g}_i \in \gamma$. And these also belong to the factors which distinguish the proposed method with [Grauman & Darrell, 2005].

5.3.2 Training and Detection

The Pyramid Match Score is a matric between two point sets. In [Grauman & Darrell, 2005], image features are considered as points, while in the proposed method, each point encodes both appearance and location information of each local feature. Each visual-spatial point is $d-$dimensional, and, the first $(d-2)$ dimensions are SIFT after PCA, while the last 2 dimensions are relative $x-$ and $y-$ coordinates after considering scale and width-height ratio changes.

Let $\mathbf{p}$ be a visual-spatial point in the point set of an image, I, and $F_{\mathbf{p}}$ be the image feature of $\mathbf{p}$, which is $(d-2)-$dimentional. Let $x_{\mathbf{p}}$ and $y_{\mathbf{p}}$ be the $x-$ and $y-$ coordinates of $\mathbf{p}$. Let w_I and h_I be the width and height of I. Then

$$\mathbf{p} = [F^1_{\mathbf{p}}, F^2_{\mathbf{p}}, ..., F^{d-2}_{\mathbf{p}}, \frac{x_{\mathbf{p}}}{w_I}, \frac{y_{\mathbf{p}}}{h_I}] \, .$$

Instead of following [Grauman & Darrell, 2005], PMS does not server as kernel functions for SVMs. And, a procedure similar to Hough transform is employed. Each training image is considered as a point set. From all the training images, the method generates a point set as a super template, $S_{\mathbf{T}}$, following Algorithm 5.1. This is just a procedure to collect all points from point sets generated from training images into one point set.

Algorithm 5.1 Template Generation

1: $S_{\mathbf{T}} \leftarrow \emptyset$
2: **for** $S_{I_{tr}} \in \{S_{I_{tr}}\}$ **do**
3: $\quad S_{\mathbf{T}} \leftarrow S_{\mathbf{T}} + S_{I_{tr}}$
4: **end for**
5: **return** $S_{\mathbf{T}}$

In Algorithm 5.1, each $S_{I_{tr}}$ in $\{S_{I_{tr}}\}$ is the point set generated from the corresponding training image I_{tr}, and the $+$ operator is defined on two sets.

Actually, $S_{\mathbf{T}}$ plays a role similar to a codebook as in methods based on Hough transform.

For detection, a most popular pipeline is employed, as in Algorithm 5.2. All possible hypotheses are generated, given by $\{\eta\}$. Each hypothesis, η is a rectangle in the image where target objects will be detected, and

$$\eta = [x_\eta, y_\eta, w_\eta, h_\eta].$$

So each η is defined by its starting (x, y) coordinate, its width, and its height. To generate $\{\eta\}$, the sliding window schema is followed, and it works by enumerating all possible rectangles by certain steps of sub-windows' positions and sizes. In Algorithm 5.2, Ω is the set of final detection results, P_{th} is a threshold to accept hypotheses as detections, I_{te} is a test image, and S_η is the point set generated by local features contained in η.

Algorithm 5.2 Detection Procedure

$\Omega \leftarrow \emptyset$, generate $\{\eta\}$ from I_{te}
for $\eta \in \{\eta\}$ **do**
  Calculate $\mathrm{P}(S_\eta, S_{\mathbf{T}}; \gamma)$
end for
Sort $\{\eta\}$ by $\mathrm{P}(S_\eta, S_{\mathbf{T}}; \gamma)$ in descending order
while $\mathrm{P}(S_{\eta_1}, S_{\mathbf{T}}; \gamma) >= \mathrm{P}_{th}$ **do**
  $\Omega \leftarrow \Omega + \eta_1$
  $\{\eta\} \leftarrow \{\eta\} - \eta_1$
  for $\eta \in \{\eta\}$ **do**
    for $\eta^{'} \in \Omega$ **do**
      for $\mathbf{p} \in \mathbf{S}_\eta$ **do**
        if $(\mathbf{p}^{(d-1)}, \mathbf{p}^d)$ is inside $\eta^{'}$ **then**
          $S_\eta \leftarrow S_\eta - \mathbf{p}$
        end if
      end for
    end for
    Calculate $\mathrm{P}(S_\eta, S_{\mathbf{T}}; \gamma)$
  end for
  Sort $\{\eta\}$ by $\mathrm{P}(S_\eta, S_{\mathbf{T}}; \gamma)$ in descending order
end while
return Ω

5.3.3 Dividing Visual-spatial Space

What is very important in Algorithm 5.2 is how to define the set of dividing methods, γ. In the method of [Grauman & Darrell, 2005], $\mathbf{g}_i$ means dividing each dimension of the point space into 2^i intervals. However, the space in [Grauman & Darrell, 2005] is a pure feature space, while the space here is a visual-spatial space. And also, in [Grauman & Darrell, 2005], the two point sets both belong to objects, while here one point set belong to the super template.

The space-dividing method proposed here divides the dimensions of visual features and spatial coordinates at different grid sizes. Let

$$\mathbf{g} = g(i, j), i, j \in N.$$

Here $g(i, j)$ is a function which defines how to divide the visual-spatial space. And i means each dimension belonging to visual channel is divided in to 2^i intervals, and j means each dimension belonging to spatial channel is divided into 2^j intervals. Note, that for a point, $\mathbf{p}$, the first $(d-2)$ dimensions belong to visual channel, while the remaining 2 dimensions belong to spatial channel. For example, if $d = 3$, then $g(2, 3)$ will divide the whole space into $(2^i)^{(d-2)} \times (2^j)^2 = 256$ grids.

In Figure 5.3, an example is given by considering visual information as one dimension, and spatial information the other dimension. Note, the total dimension of a point is actually d, while in the example it is 2.

About γ, not only its members, but also the order of its members is important. For 5.1 to work, a requirement must be fulfilled, that if $i < j$, $\mathbf{g}_i$ is finer than $\mathbf{g}_j$, which means if two points are decided as match under $\mathbf{g}_i$, they must be decided as match under $\mathbf{g}_j$. This is,

$$i < j, G(\mathbf{p}_{S_1}; \mathbf{g}_i) = G(\mathbf{p}_{S_2}; \mathbf{g}_i) \;\Rightarrow\; G(\mathbf{p}_{S_1}; \mathbf{g}_i) = G(\mathbf{p}_{S_2}; \mathbf{g}_i)\,. \tag{5.3}$$

If $\mathbf{g}_i$ is finer than $\mathbf{g}_j$, it is also written as $\mathbf{g}_i > \mathbf{g}_j$.

In 5.3, $G(\mathbf{p}; \mathbf{g})$ is a function to map $\mathbf{p}$ into a particular grid, given dividing method

$\mathbf{g}$. And $G(\mathbf{p}; \mathbf{g})$ on the kth dimension is defined by,

$$G^k(\mathbf{p}; g(i,j)) = \begin{cases} \lfloor \frac{2^i \times (\mathbf{p}^k_{max} - \mathbf{p}^k)}{\mathbf{p}^k_{max} - \mathbf{p}^k_{min}} \rfloor & \text{if } k \leq (d-2) \\ \lfloor \frac{2^j \times (\mathbf{p}^k_{max} - \mathbf{p}^k)}{\mathbf{p}^k_{max} - \mathbf{p}^k_{min}} \rfloor & \text{otherwise} \end{cases} .$$

Here, $\mathbf{p}^k_{max}$ and $\mathbf{p}^k_{min}$ are the maximum and minimum values on the kth dimension, which are determined by training dataset.

There is no such constrain which requires γ is descending ordered by the fineness level in 5.2. Thus, in the following paper, 5.2 will be used. In fact, when 5.3 is satisfied, 5.1 and 5.2 are the same.

Though 5.2 can be used to calculate a pyramid match score for two point sets, given any set, γ, still the dividing methods and the order of the dividing methods will affect performance. For a largest fineness level, $l_{max}, l_{max} \in N$, γ is defined in Algorithm 5.3.

Algorithm 5.3 Set of Dividing Methods Generation

$\gamma \leftarrow \emptyset$, $r \leftarrow 2 \times (l_{max} - 1)$
while $r \geq 0$ **do**
  if $r \geq l_{max} - 1$ **then**
    $i \leftarrow l_{max} - 1$
  else
    $i \leftarrow r$
  end if
  $j \leftarrow r - i$
  while $i \leq (l_{max} - 1)$ **and** $i \geq 0$ **and** $j \leq (l_{max} - 1)$ **and** $j \geq 0$ **do**
    $\gamma \leftarrow \gamma + g(i,j)$, $i \leftarrow i - 1$, $j \leftarrow r - i$
  end while
  $r \leftarrow r - 1$
end while
return γ

The size of γ, $L = l_{max} \times l_{max}$. For two dividing methods $\mathbf{g}_i, i \in 1, 2, ..., L$ and $\mathbf{g}_j, j \in 1, 2, ..., L$, if $\mathbf{g}_i > \mathbf{g}_j$, then $i < j$, which means if one dividing method is finer than the other, it will appear earlier in the set of dividing methods. There are also dividing methods, of which the fineness level cannot be compared, i.e., $g(1,2)$ and $g(2,1)$ as shown in Figure 5.3.

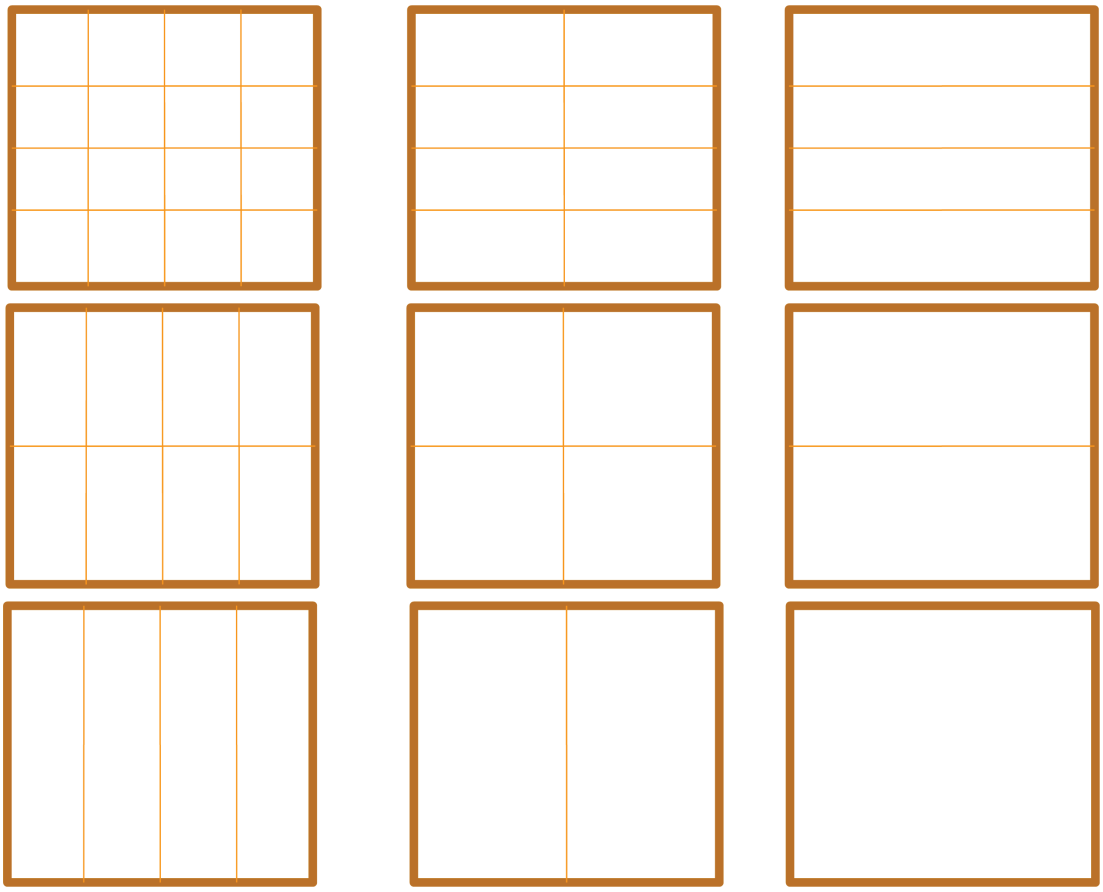


Figure 5.3: An example set of methods to divide the visual-spatial space. $x-$ and $y-$ coordinates represent visual and spatial information respectively. From left to right, the first line is $g(2,2)$, $g(1,2)$, and $g(0,2)$. The second line is $g(2,1)$, $g(1,1)$, and $g(0,1)$. And the third line is $g(2,0)$, $g(1,0)$, and $g(0,0)$. And γ is defined as an ordered set of all the dividing methods with different parameters, i.e., $\gamma = \{g(2,2), g(1,2), g(2,1), g(0,2), g(1,1), g(2,0), g(0,1), g(1,0), g(0,0)\}$.

5.3.4 Deciding Weights for Dividing Methods

After how to divide the visual-spatial space is decided, the remaining task is, for each dividing method $\mathbf{g}$, defining a corresponding weight. When talking about two points which are found in the same grid under $\mathbf{g} = g(i,j)$, there is an upper bound to their $L1$-distance, which is given by

$$D_{ub} = (d-2) \times \frac{1}{2^i} + 2 \times \frac{1}{2^j},$$

if unit length is assumed for all $(\mathbf{p}^k_{max} - \mathbf{p}^k_{min}), k \in \{1, 2, ..., d\}$. Since the first $(d-2)$ dimensions of each grid under $g(i,j)$ possess length of $\frac{1}{2^i}$, while the last 2 dimensions possess length of $\frac{1}{2^j}$.

Following [Grauman & Darrell, 2005], for two point from different point sets, if they are in the same grid under $g(i,j)$, which means $G(\mathbf{p}_{S_1}; g(i,j)) = G(\mathbf{p}_{S_2}; g(i,j))$, the visual difference between $\mathbf{p}_{S_1}$ and $\mathbf{p}_{S_2}$ is defined as $\frac{(d-2)}{2^i}$, and the spatial difference is defined as, $\frac{2}{2^j}$. A weight, ω, defined for a dividing method $g(i,j)$ shows the importance of two matched points, and measures how difficult it is to match under such dividing method.

$$\omega_{g(i,j)} = \sqrt{((d-2) \times 2^i) \times (2 \times 2^j)}. \tag{5.4}$$

As is seen in 5.4, the finer one grid is in $g(i,j)$, the larger a weight will be assigned for it. The weight is the confidence that the point set belong to a target object based on a point has corresponding evidence from the super template under the current $\mathbf{g}$.

5.3.5 Learning Weights for Dividing Methods

Besides directly assigning weights to all the dividing methods in a deterministic way, as in 5.4, a framework for learning weights is here proposed.

Often Gaussian kernels are used to measure differences between two features or two positions in Hough transform based methods following [Leibe et al., 2008]. For two visual-spatial points found match in the same grid under dividing method, $g(i,j)$, the visual difference is $\frac{(d-2)}{2^i}$, and the spatial difference is $\frac{2}{2^j}$. The total difference between two points found match in the same grid is modeled using a 2D Gaussian

kernel, as

$$\omega_{g(i,j)} = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp(-\frac{1}{1-\rho^2}(\frac{(\frac{(d-2)}{2^i})^2}{{\sigma_1}^2} + \frac{(\frac{2}{2^j})^2}{{\sigma_2}^2} - \frac{2\rho\frac{(d-2)}{2^i}\frac{2}{2^j}}{\sigma_1\sigma_2})). \quad (5.5)$$

In 5.5, ρ is the correlation between visual and spatial channel, while σ_1 and σ_2 are standard deviations for visual and spatial channel.

To make 5.5 clear, it is rewritten as,

$$\omega_{g(i,j)} = t\exp(-a(\frac{1}{2^i})^2 - b(\frac{1}{2^j})^2 + c(\frac{1}{2^i})(\frac{1}{2^j})). \quad (5.6)$$

Where,

$$t = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}},$$
$$a = \frac{(d-2)^2}{(1-\rho^2){\sigma_1}^2},$$
$$b = \frac{2^2}{(1-\rho^2){\sigma_2}^2}, \text{ and}$$
$$c = \frac{2(d-2)(2)\rho}{(1-\rho^2)\sigma_1\sigma_2}.$$

In 5.6, the larger, the visual difference, or the larger, the spatial difference, the smaller the corresponding weight. This is decided by the 2D Gaussian kernel, and also this is consistent with Hough transform based methods.

In Algorithm 5.1, the weights are not needed, and the super template, $S_{\mathbf{T}}$, can be generated firstly. Then the performance of Algorithm 5.2 will rely on the set of dividing method, γ, and each corresponding weight, ω.

In 5.6, to define weight for $g(i,j)$, besides i and j, still there are four parameters, t, a, b, and c, need to be given. Here, t is just a factor, it won't affect the results of Algorithm 5.2, which are the final detection results.

Here, a, b, and c have their meanings. When a is larger, visual information will play a more important role, and when b is larger, spatial information will play a more important role. What is more interesting is that, there is c, which will be responsible for modeling correlating visual-spatial information.

To estimate a, b, and c for a particular γ, all positive training images, $\{I_p\}$, and

negative training images, $\{I_n\}$, are used. For brevity, the pyramid match score against the super template, with defined γ, is rewritten according to 5.2,

$$\begin{aligned} \mathrm{P}(S_I, S_{\mathbf{T}}; \gamma) &= \frac{\sum\limits_{i,j} \omega_{g(i,j)} \times N(S_I^{(i-1)}, S_{\mathbf{T}}^{(i-1)}; g(i,j))}{m} \\ &= t \sum_{i,j} \exp(-a(\frac{1}{2^i})^2 - b(\frac{1}{2^j})^2 + c(\frac{1}{2^i})(\frac{1}{2^j})) \times \frac{N(S_I^{(i-1)}, S_{\mathbf{T}}^{(i-1)}; g(i,j))}{m} \end{aligned} \tag{5.7}$$

In 5.7, after summing up along i, and j at given γ and $S_{\mathbf{T}}$, it will be a function which changes according to I, a, b, and c. It is rewritten as $\mathrm{PMS}(I; a, b, c)$ for brevity.

The objective function is written as the gap between pyramid match scores of positive training images and negative training images under normalizing condition as,

$$\begin{aligned} &\arg\max_{a,b,c} \frac{\frac{\sum \mathrm{PMS}(I_p; a, b, c)}{|\{I_p\}|} - \frac{\sum \mathrm{PMS}(I_n; a, b, c)}{|\{I_n\}|}}{\frac{\sum \mathrm{PMS}(I_p; a, b, c) + \sum \mathrm{PMS}(I_n; a, b, c)}{|\{I_p\}| + |\{I_n\}|}} \\ &s.t.: \ a > 0; \\ &\qquad b > 0 \ . \end{aligned}$$

After all positive and negative training examples are given, the objective function will only contain a, b, and c. For such a function, brute-force solutions will be feasible.

So far, in Algorithm 5.1, how the template can be generated is defined, , so Algorithm 5.2 can be used for detection.

5.4 Experimental Results

In Figure 5.4, are some good results on UIUC cars. And new experiments will be added.

In Figure 5.5, evaluations are given.

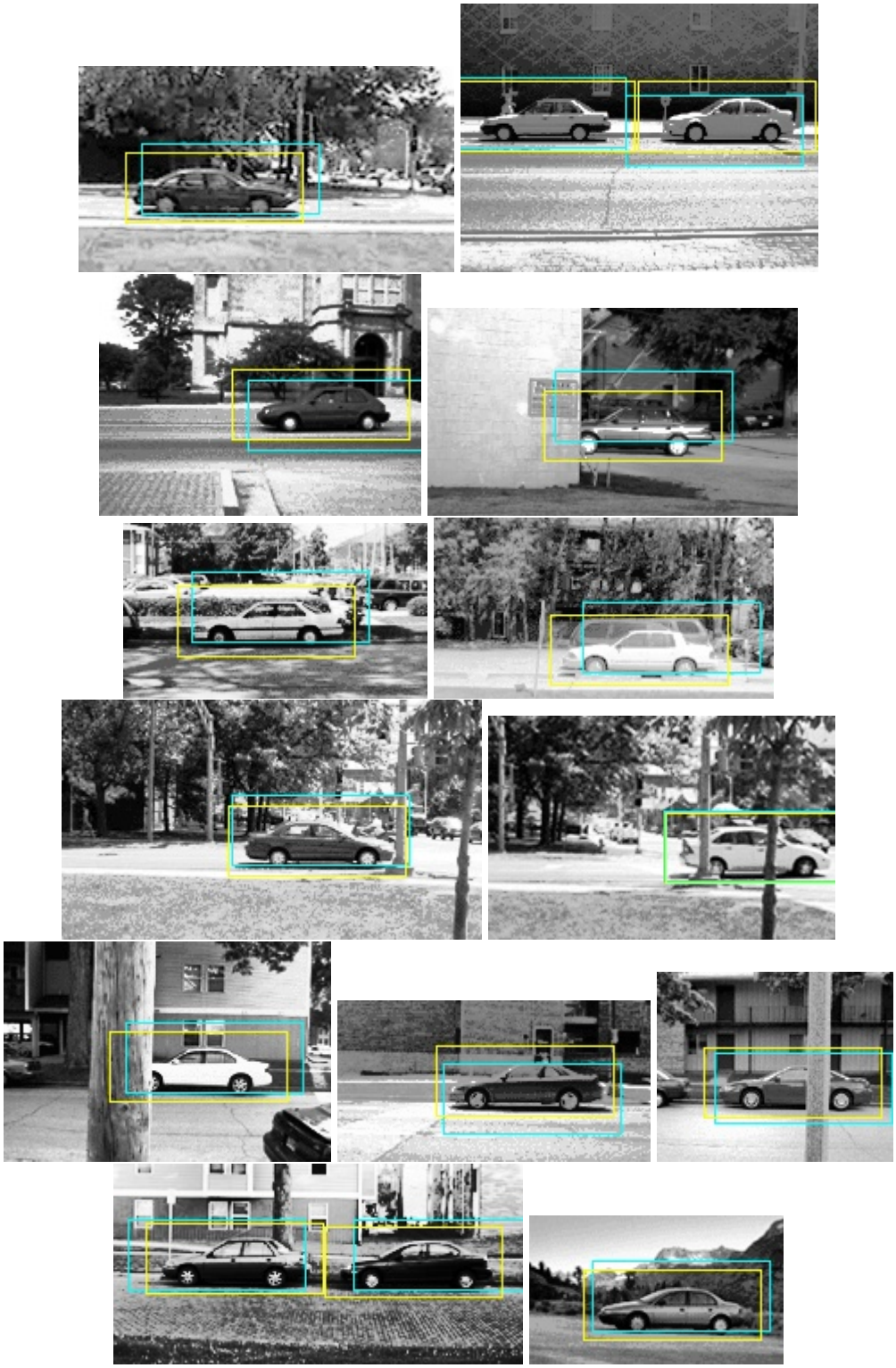


Figure 5.4: Detection results on UIUC cars [Agarwal et al., 2004]. Yellow color marks ground truths, while blue marks detections.

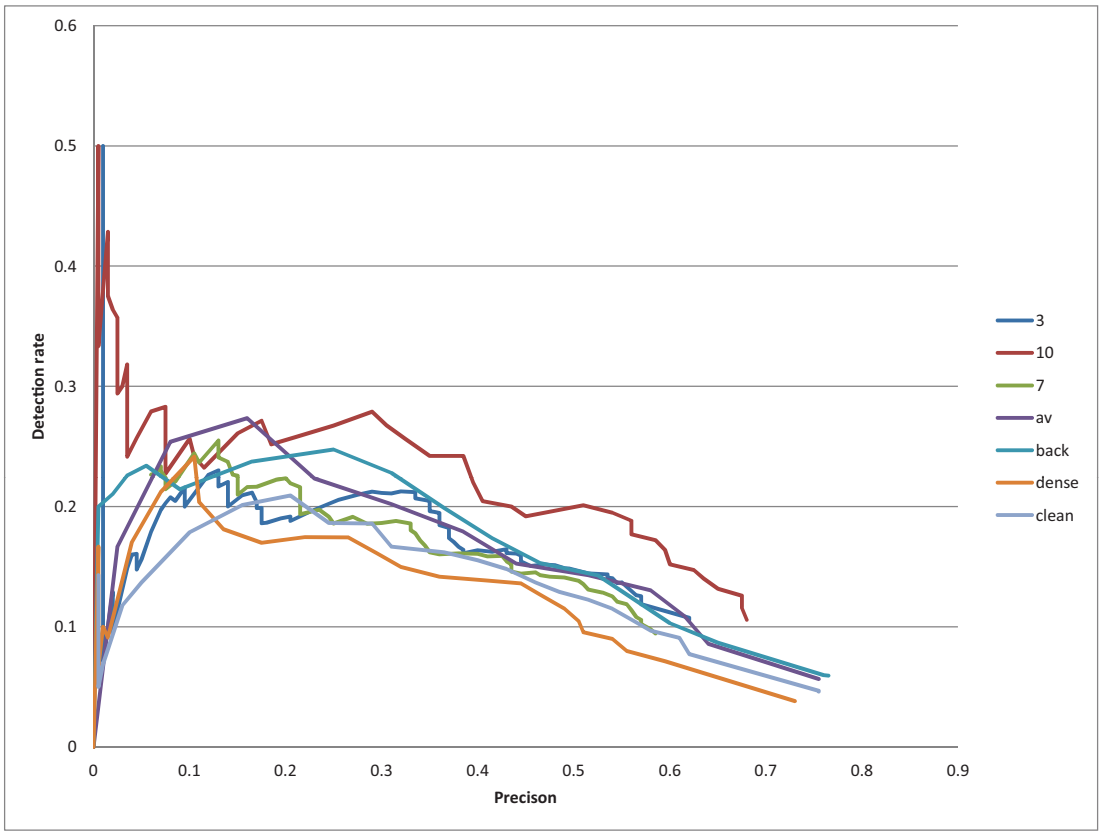


Figure 5.5: Results evaluation on UIUC cars with different implementation parameters.

5.5 Chapter Conclusion

There will be several experiments to be summarized for this chapter, and also several experiments to be added.

Chapter 6

Conclusion and outlook

In Chapter 2.

Visual object detection by computers is still and, in near future, continues to be a very open problem. There are several things we can expect which will improve the performance of the detection methods.

Bibliography

Agarwal, S., Awan, A., & Roth, D. (2004). Learning to detect objects in images via a sparse, part-based representation. *PAMI*, *26*(11), 1475–1490. 81

Alexe, B., Deselaers, T., & Ferrari, V. (2010). What is an object? In *CVPR*, (pp. 73–80). 11

Attias, H. (2000). A variational bayesian framework for graphical models. In *NISP*, (pp. 209–215). MIT Press. 9

Baidu. baidu image search. `http://stu.baidu.com`. [Online; accessed 22-May-2013]. 5

Ballard, D. (1981). Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, *13*(2), 111–122. 41

Barinova, O., Lempitsky, V., & Kohli, P. (2010). On detection of multiple object instances using hough transforms. In *CVPR*, (pp. 2233–2240). 1, 11, 15, 42, 44, 53, 56

Bay, H., Tuytelaars, T., & Van Gool, L. (2006). Surf: Speeded up robust features. In *ECCV*, (pp. 404–417). 8, 17

Bay, H., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (surf). *CVIU*, *110*(3), 346–359. 36

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *PAMI*, *24*(4), 509–522. 8

Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, *2*(1), 1–127. 2, 7, 9

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022. 9

Brostow, G. & Cipolla, R. (2006). Unsupervised bayesian detection of independent motion in crowds. In *CVPR*, (pp. I: 594–601). 16, 45

Brox, T. & Malik, J. (2010). Object segmentation by long term analysis of point trajectories. In *ECCV*, (pp. V: 282–295). 16, 46

Cadieu, C. F., Hong, H., Yamins, D., Pinto, N., Majaj, N. J., & DiCarlo, J. J. (2013). The neural representation benchmark and its evaluation on brain and machine. *CoRR*, *abs/1301.3530*. 6

Dalal, N. & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR*, (pp. 886–893). 1, 8, 11, 15, 69

Davidson, P., Hautamaki, J., & Collin, J. (2008). Using low-cost mems 3d accelerometer and one gyro to assist gps based car navigation system. In *International Conference on Integrated Navigation Systems*. 17

Dean, T., Ruzon, M., Segal, M., Shlens, J., Vijayanarasimhan, S., & Yagnik, J. (2013). Fast, accurate detection of 100,000 object classes on a single machine. In *CVPR*. 10, 69

Degtyarev, N. & Seredin, O. (2010). Comparative testing of face detection algorithms. In *ICISP*, (pp. 200–209). 1

Estrada, F., Fua, P., Lepetit, V., & Susstrunk, S. (2009). Appearance-based keypoint clustering. In *CVPR*, (pp. 1279–1286). 45

Felzenszwalb, P. & Huttenlocher, D. (2005). Pictorial structures for object recognition. *IJCV*, *61*(1), 55–79. 1, 11, 15, 41

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *PAMI*, *32*(9), 1627–1645. 69

Felzenszwalb, P. F., Girshick, R. B., & McAllester, D. A. (2010). Cascade object detection with deformable part models. In *CVPR*, (pp. 2241–2248). 69

Felzenszwalb, P. F., McAllester, D. A., & Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *CVPR*, (pp. 1–8). 1, 11, 15, 69

Fergus, R., Perona, P., & Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In *CVPR*, (pp. II: 264–271). 1, 11, 15, 41, 68, 69

Ferguson, T. S. (1973). A bayesian analysis of some nonparametric problems. *Annals of Statistics*, *1*, 209–230. 9

Ferrari, V., Jurie, F., & Schmid, C. (2007). Accurate object detection with deformable shape models learnt from images. In *CVPR*, (pp. 1–8). 1, 11, 15, 66, 69

Gall, J. & Lempitsky, V. (2009). Class-specific hough forests for object detection. In *CVPR*, (pp. 1022–1029). 16, 41, 44

Gionis, A., Indyk, P., & Motwani, R. (1999). Similarity search in high dimensions via hashing. In *VLDB*, (pp. 518–529). 69

Gould, S., Fulton, R., & Koller, D. (2009). Decomposing a scene into geometric and semantically consistent regions. In *ICCV*, (pp. 1–8). 45

Grauman, K. & Darrell, T. (2005). The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, (pp. 1458–1465). 70, 72, 73, 75, 78

Gu, C., Lim, J., Arbelaez, P., & Malik, J. (2009). Recognition using regions. In *CVPR*, (pp. 1030–1037). 41

Harris, C. & Stephens, M. (1988). A combined corner and edge detector. In *Alvey Vision Conference*, (pp. 147–152). 51

Huang, C., Wu, B., & Nevatia, R. (2008). Robust object tracking by hierarchical association of detection responses. In *ECCV*, (pp. II: 788–801). 16

Isukapalli, R. & Greiner, R. (2003). Use of off-line dynamic programming for efficient image interpretation. In *IJCAI*, (pp. 1319–1325). 10, 45

Jégou, H., Douze, M., & Schmid, C. (2010). Improving bag-of-features for large scale image search. *IJCV*, *87*(3), 316–336. 2, 8, 11, 66, 68

Jiang, H. & Yu, S. (2009). Linear solution to scale and rotation invariant object matching. In *CVPR*, (pp. 2474–2481). 12, 68

Kise, K. & Kashiwagi, T. (2011). 1.5 million subspaces of a local feature space for 3d object recognition. In *ACPR*, (pp. 672–676). 6, 11

Kläser, A., Marszalek, M., & Schmid, C. (2008). A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, (pp. 1–10). 46

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 83–97. 24, 70

Lampert, C., Blaschko, M., & Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*, (pp. 1–8). 1, 11, 15, 41, 42

Lampert, C. H., Blaschko, M. B., & Hofmann, T. (2009). Efficient subwindow search: A branch and bound framework for object localization. *PAMI*, *31*(12), 2129–2142. 2, 70

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, (pp. 2169–2178). 66, 68

Le, Q. V., Ranzato, M., Monga, R., Devin, M., Corrado, G., Chen, K., Dean, J., & Ng, A. Y. (2012). Building high-level features using large scale unsupervised learning. In *ICML*. 6, 11

Lehmann, A., Leibe, B., & Van Gool, L. (2011). Fast prism: Branch and bound hough transform for object class detection. *IJCV*, *94*(2), 175–197. 10, 16, 46, 66

Leibe, B., Cornelis, N., Cornelis, K., & Van Gool, L. (2007). Dynamic 3d scene analysis from a moving vehicle. In *CVPR*, (pp. 1–8). 1, 11, 15, 44

Leibe, B., Leonardis, A., & Schiele, B. (2008). Robust object detection with interleaved categorization and segmentation. *IJCV*, *77*(1-3), 259–289. 1, 2, 11, 15, 41, 44, 78

Leibe, B. & Schiele, B. (2003). Interleaved object categorization and segmentation. In *BMVC*, (pp. 759–768). 1, 11, 15, 44

Leibe, B., Schindler, K., & Van Gool, L. (2007). Coupled detection and trajectory estimation for multi-object tracking. In *ICCV*, (pp. 1–8). 16, 45

Liu, C., Yuen, J., & Torralba, A. (2011). Nonparametric scene parsing via label transfer. *PAMI*, *33*(12), 2368–2382. 68

López-Sastre, R. J., Tuytelaars, T., & Savarese, S. (2011). Deformable part models revisited: A performance evaluation for object category pose estimation. In *ICCV Workshops*, (pp. 1052–1059). 69

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *IJCV*, *60*, 91–110. 8, 17, 27, 45

Lucas, B. D. & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *IJCAI*, (pp. 674–679). 45

Maji, S., Berg, A. C., & Malik, J. (2008). Classification using intersection kernel support vector machines is efficient. In *CVPR*, (pp. 1–8). 1, 11, 15

Maji, S. & Malik, J. (2009). Object detection using a max-margin hough transform. In *CVPR*, (pp. 1038–1045). 1, 11, 15, 16, 44

Matas, J., Chum, O., Martin, U., & Pajdla, T. (2002). Robust wide baseline stereo from maximally stable extremal regions. In *BMVC*, (pp. 384–393). 8

Meeds, E., Ross, D., Zemel, R., & Roweis, S. (2008). Learning stick-figure models using nonparametric bayesian priors over trees. In *CVPR*, (pp. 1–8). 8

Merkle, R. C. (1989). Energy limits to the computational power of the human brain. *Foresight Update*, *No. 6*. 6

Mikolajczyk, K., Leibe, B., & Schiele, B. (2006). Multiple object class detection with a generative model. In *CVPR*, (pp. I: 26–36). 1, 11, 12, 15, 44

Mikolajczyk, K. & Schmid, C. (2004). Scale & affine invariant interest point detectors. *IJCV*, *60*(1), 63–86. 8

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., & Van Gool, L. (2005). A comparison of affine region detectors. *IJCV*, *65*(1-2), 43–72. 2

Mohottala, S., Ono, S., Kagesawa, M., & Ikeuchi, K. (2009). Fusion of a camera and a laser range sensor for vehicle recognition. In *OTCBVS*, (pp. 16–23). 46

Ohba, K. & Ikeuchi, K. (1997). Detectability, uniqueness, and reliability of eigen windows for stable verification of partially occluded objects. *PAMI*, *19*(9), 1043–1047. 1, 11, 15, 41

Ojala, T., Pietikäinen, M., & Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, *29*(1), 51–59. 8

Okada, R. (2009). Discriminative generalized hough transform for object dectection. In *ICCV*, (pp. 2000–2005). 16, 41, 44

Olshausen, B. & Field, D. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*, 607–609. 70

Ommer, B. & Malik, J. (2009). Multi-scale object detection by clustering lines. In *ICCV*, (pp. 484–491). 44

Ono, S., Xue, L., Banno, A., Oishi, T., & Ikeuchi, K. (2012). Global 3d modeling and its evaluation for large-scale highway tunnel using laser range sensor. In *ITS World Congress*. 17

Pan, S. J. & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359. 9

PASCAL. Visual Object Classes Challenge 2012. http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2012. [Online; accessed 22-May-2013]. 7

Pedersoli, M., Vedaldi, A., & Gonzàlez, J. (2011). A coarse-to-fine approach for fast deformable object detection. In *CVPR*, (pp. 1353–1360). 69

Pinto, N., Doukhan, D., DiCarlo, J. J., & Cox, D. D. (2009). A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS Computational Biology*, *5*(11). 8

Pirsiavash, H. & Ramanan, D. (2012). Steerable part models. In *CVPR*, (pp. 3226–3233). 10, 69

Rahtu, E., Kannala, J., & Blaschko, M. B. (2011). Learning a category independent object detection cascade. In *ICCV*, (pp. 1052–1059). 10

Rasmussen, C. E. & Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press. 9

Rowley, H., Baluja, S., & Kanade, T. (1998). Rotation invariant neural network-based face detection. In *CVPR*, (pp. 38–44). 12

Russakovsky, O. & Ng, A. (2010). A steiner tree approach to efficient object detection. In *CVPR*, (pp. 1070–1077). 15

Schneiderman, H. & Kanade, T. (2004). Object detection using the statistics of parts. *IJCV*, *56*(3), 151–177. 1, 11, 15

Shawe-Taylor, J. & Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press. 68, 70

Shechtman, E. & Irani, M. (2007). Matching local self-similarities across images and videos. In *CVPR*, (pp. 1–8). 8

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., & Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *CVPR*, (pp. 1297–1304). 6

Sincich, L. C. & Horton, J. C. (2005). THE CIRCUITRY OF V1 AND V2: Integration of Color, Form, and Motion. *Annual Review of Neuroscience*, *28*(1), 303–326. 46

Song, H. O., Zickler, S., Althoff, T., Girshick, R., Fritz, M., Geyer, C., Felzenszwalb, P., & Darrell, T. (2012). Sparselet models for efficient multiclass object detection. In *ECCV*, (pp. 716–723). 69

Stauffer, C. & Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. In *CVPR*, (pp. 2246–2252). 19

Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei., D. M. (2006). Hierarchical dirichlet process. *Journal of the American Statistical Association*, 1566–1581. 2, 9

Tomasi, C. & Kanade, T. (1991). Detection and tracking of point features. Technical report, IJCV, Carnegie Mellon University. 8, 45, 51

Tuzel, O., Porikli, F., & Meer, P. (2006). Region covariance: A fast descriptor for detection and classification. In *ECCV*, (pp. II: 589–600). 8, 45, 51

Viola, P. A. & Jones, M. J. (2004). Robust real-time face detection. *IJCV*, *57*(2), 137–154. 1, 11, 15

Wang, Z., Kagesawa, M., Ono, S., Banno, A., & Ikeuchi, K. (2010). Emergency light detection in tunnel environment: An efficient method. In *ACPR*, (pp. 628 – 632). 34

Wertheimer, M. (1938). Laws of organization in perceptual forms (partial translation). In Ellis, W. B. (Ed.), *A Sourcebook of Gestalt Psychology*, (pp. 71–88). Harcourt, Brace. 42

Wikipedia. Object Detection. `http://en.wikipedia.org/wiki/Object_detection`. [Online; accessed 22-May-2013]. 1

Wikipedia. Siri. `http://en.wikipedia.org/wiki/Siri_(software)`. [Online; accessed 22-May-2013]. 6

Wojek, C., Roth, S., Schindler, K., & Schiele, B. (2010). Monocular 3d scene modeling and inference: Understanding multi-object traffic scenes. In *ECCV*, (pp. IV: 467–481). 16, 17, 46

Yang, L., Zheng, N., Chen, M., Yang, Y., & Yang, J. (2009). Categorization of multiple objects in a scene without semantic segmentation. In *ACCV*, (pp. 303–312). 11, 46

Yang, M., Yu, T., & Wu, Y. (2007). Game-theoretic multiple target tracking. In *ICCV*, (pp. 1–8). 16

Yarlagadda, P., Monroy, A., & Ommer, B. (2010). Voting by grouping dependent parts. In *ECCV*, (pp. V: 197–210). 16, 44

Yeh, T., Lee, J., & Darrell, T. (2009). Fast concurrent object localization and recognition. In *CVPR*, (pp. 280–287). 1, 10, 11, 15, 41

Zhang, L., Li, Y., & Nevatia, R. (2008). Global data association for multi-object tracking using network flows. In *CVPR*, (pp. 1–8). 16, 45

Zhu, L. L., Chen, Y., Yuille, A., & Freeman, W. (2010). Latent hierarchical structural learning for object detection. In *CVPR*, (pp. 1062–1069). 69

Acknowledgement

Thank professor Ikeuchi for taking care of me and leading me in my research.

Thank Kagesawa San.

Thank Ono San. Thank Banno San. Thank every staff and student in computer vision lab for all the help.

Thank all the people helped me with my life in Tokyo.

I just have experienced too much during my doctor course.