Contents

List of Figures

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. In computer science, object detection [Wikipedia, 2008] 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 many 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 make inferences about the target objects' states, 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, many researchers in the area of computer vision 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 previous methods do not make full use of. 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 how to combine visual and spatial information encoded in local image features of the same object.

For fusion of information from different channels, voting systems are employed. Voting is preferred for its robustness in using local information, and its inference procedure's capability to use global information.

Many methods for detection mainly use either appearance information or motion information. Following some previous work [Jones et al., 2003], the first two methods will address that when well combining the information from these two channels, detection performance will be better.

The first method is developed mainly for real-time applications under limited computational power. This method can be considered as a three-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. The last step feeds the detection results from step two into a voting system. Since detection results are connected by their belonging trajectories, voting along the temporal dimension is responsible for giving the final decision of each object, when it disappears from the scene. Motion information plays a very important role in the method. The target objects are considered as possessing both particular appearance patterns and motion patterns. When the second step verifies the detection hypotheses using appearance information, a biased classifier is used. This classifier produces more false alarms to pursue higher detection rate. Then motion information is used to filter out the false alarms. Motion information in the form of trajectories also connects weak inferences and feeds the weak inferences into a voting system for the final results. In addition, the pipeline of this method is optimized in a hierarchical way. In the pipeline, 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 a tunnel environment, and gives promising detection results in 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 transform. 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 inferences with more consistence in the same motion group. On two datasets, the proposed method outperforms the state-of-the-art method.

While the second method performs well under complicated scene, it is relatively slow. This is due to the time-consuming property of methods based on Hough transform. 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 transform, 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 considers objects as point sets of, i.e., of 12-dimensional, while the first 10 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 for querying, the point set of a sub-image is used for querying. Pyramid Matching is used for accelerating the querying. The pro-

cedure ensures the full use of the visual and spatial information encoded in the image features of the same object. While giving promising detection results on two datasets, this method is confirmed to be much more efficient than the second method.

The thesis is organised as follows. Chapter 2 gives background and reviews some related work. Chapter 3 introduces the method aimed at efficient detection by combining motion and appearance information. Chapter 4 proposes the method that extends the Implicit Shape Model to incorporate motion information, and the method groups object parts for detection. Chapter 5 presents the method which detects by Pyramid Match Score. Chapter 6 concludes, and discusses about possible improvements of the proposed methods for future work.

Chapter 2

Background and Related Work

Every day, new technologies are brought to practice from labs, provide convenience to daily life, and promote advances in human culture. Among these technologies, the emerging of text searching engines like google is a great milestone. Retrieving images [Baidu, 2013] 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. Deciding semantic labels for images is one, and detection of target objects from images is another. The efforts in this thesis try to contribute to detecting target objects from images or image sequences.

This chapter first discusses possible advantages in human beings' visual perception against machines, and how these advantages will activate new machine vision methods in Section 2.1. Then abstractions in artificial intelligence which are closely related to machine vision are reviewed in Section 2.2. This chapter also reviews basics and advances in visual object detection in Section 2.3 and 2.4. Finally this chapter summarizes by giving why the efforts of the thesis are important in Section 2.5.

2.1 Vision in Humans and Computers

The computational power of humans is not any more competitive to computers, especially nowadays, when computational resources can be conveniently accessed from the cloud. Still when talking about the performance of visual recognition and detection under general situations, humans are still champions. 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?

Table 2.1: Comparisons of visual capabilities between computers and humans.

	Computers	Humans
Quality of sensors	*	
Computational capability	*	
Representative model		*
Decision making procedure		*
Information fusion mechanisms		*
Training quality		*

When compared with humans, computers will win at almost every aspect of hardware, as shown in Table 2.1. Besides visual sensors, new sensors are continuously developed for computers, and computers have so many choices for sensors of very high quality. What is worth mentioning is how Microsoft's Kinect advances pose recognition [Shotton et al., 2011]. Especially, for computational ability, [Merkle, 1989] believes human brains have a raw computational power between 10^{13} and 10^{16} operations per second, and modern computers are much more powerful.

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 believes that 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 amounts of examples, which makes the training quality very solid. The performance of new born babies also leads to considerations about what percentage the visual abilities in human are decided by their 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 its successful deployment in industry [Wikipedia, 2012] 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 possible for computers to share similar representative models with humans. 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 parallel training. The recent deep learning [Bengio, 2009] methods try to fusion representative model with decision making 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 making 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 fusion of the visual and spatial information of different parts of the same object. The efforts here shall belong to decision making procedure, and information fusion mechanism. Especially, in Chapter 4, a perceptual mechanism in human are combined into a voting system, and this results in a detection method with promising detection performance.

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 that fact that 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 to this goal, 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 believes 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, 2012]. Significant amount of engineering efforts are needed to win this kind of competitions. Another routine is that development of mechanisms of high abstraction 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.

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

The methods proposed in this thesis explore the detection procedure at a high abstraction level.

2.3 Basics of Detection Methods

In the previous section, the possible reason how humans perform better in detection is discussed. The main limitations of computers are of software. Based on this, researchers present new detection methods or new methods for supporting detection, which includes: 1) new image features or new representative models, 2) new decision making 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 an 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 requirements for image features, while some keypoint features [Bay et al., 2006; Lowe, 2004; Matas et al., 2002; Mikolajczyk & Schmid, 2004; Tomasi & Kanade, 1991] fulfill these requirements. 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 Making Procedure

Generally, robustness and computational efficiency are the two main pursues in proposing image features. Based on image features, how decisions are made are also of great importance. Dimension reduction methods like principal 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.

The are mainly two main categories of methods for decision making: discriminative methods, and generative methods. 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 differences 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 that 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 model 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 separates deep learning methods from traditional neural networks lies in the training procedure. Traditional neural networks are trained as a whole, which means the inferring for lots of parameters at the same time, and good 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 Exploring

Besides proposing representative models and decision making 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 check each differently scaled sub-image of a target image, and answer whether this sub-image contains an object of interest. While the number of all sub-windows is extremely large, and even enumerating 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 of [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 in distributed computations, the underlining 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 using 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, and methods most 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] tries 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 detec-

tion. 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 describing each object. Instead of being 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 methods also detect or recognize objects in 3D space. The method proposed in [Kise & Kashiwagi, 2011], tracks keypoints of the same object, generate features which include 3D information of the object accordingly, and feed the features to decision step. And the results are very appealing.

Also excellent performance of deep learning inspires new methods to reconsider object representations. 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 estimating the rotation angle of object. This method heavily relies on the SIFT features. An 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.

2.5 Chapter Conclusion

There methods are proposed in the thesis. The methods contribute to decision making in object detection. These methods try to improve existing voting systems to easily employ information in multiple channels or try to accelerate the decision making procedure by employing good mechanisms.

Human beings have their limitations in computational capabilities while perform better in visual tasks. Thus machine vision methods should make use of more information in a more effective manner to perform better. This encourages the efforts in this thesis. And especially the method of Chapter 4 learns from one mechanism of human beings.

High abstraction level in AI will be helpful for future visual detection methods. And the method in Chapter 3 has several layers in its pipeline, and makes it unique in detect specific objects. Information from all channels are then combined in a robust voting system for better performance.

From the basics of detection methods, it is clear decision making procedure is still the most challenging part in object detection. All methods of the thesis focus on decision making.

To summarise this chapter, the methods proposed in this thesis try to contribute to object detection by proposing effective and efficient mechanisms to combining information from multiple channels in robust voting systems. These efforts are encouraged by the human beings' amazing visual capabilities, these efforts try to act at a high abstraction level, and these efforts belong to the very challenging topic of decision making in object detection, as shown in Figure 2.1.

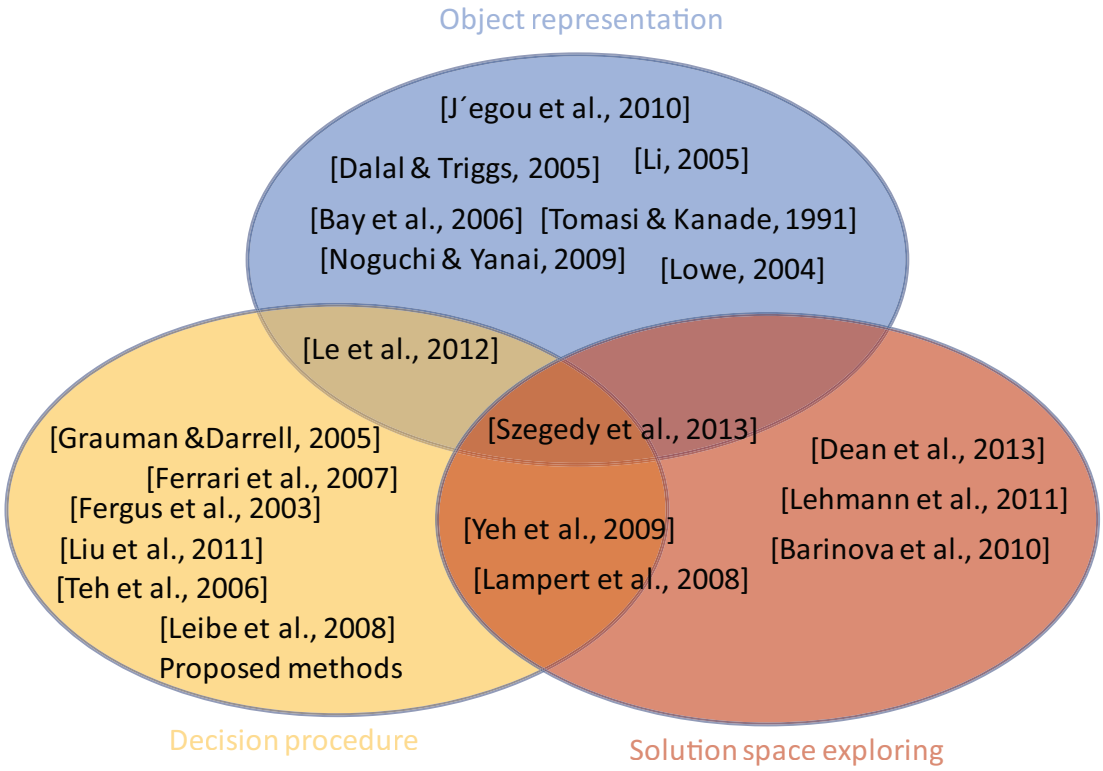


Figure 2.1: Categorization of the contribution from the proposed methods and a few recent methods [Barinova et al., 2010; Bay et al., 2006; Dalal & Triggs, 2005; Dean et al., 2013; Fergus et al., 2003; Ferrari et al., 2007; Grauman & Darrell, 2005; Jégou et al., 2010; Lampert et al., 2008; Le et al., 2012; Lehmann et al., 2011; Leibe et al., 2008; Li, 2005; Liu et al., 2011; Lowe, 2004; Noguchi & Yanai, 2009; Szegedy et al., 2013; Teh et al., 2006; Tomasi & Kanade, 1991; Yeh et al., 2009].

Chapter 3

Efficient Voting along Time Axis

3.1 Introduction

From time to time, efficient detection methods under limited computational power are important in various areas, such as driving assistance using embedded sensors. In this chapter, 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 the detecting pipeline, the method works in real time, and gives promising detection results in experiments.

As always, detection performance and efficiency are the two important aspects of this method. The clutter property of the sensed data makes the detection challenging. In extreme cases, target objects cannot be distinguished from noisy objects by appearance. The proposed method meets this challenge by making use of both appearance and temporal information of the target objects. There are three main steps in the method.

The first step deals with keypoints. 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 built 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 keypoints which pass verification, 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. At the end of this step, the keypoint clustered are labeled as positive or negative according to whether they pass both appearance and motion verifications.

Voting systems are employed in the last step. The output of the second step are some keypoint clusters that pass both appearance and motion verifications. These keypoint clusters are connected by trajectories. Voting is carried on along each trajectory when it ends. If the percentage of positive keypoint clusters connected by one trajectory is larger than a threshold, this trajectory is then considered as an emergency telephone indicator.

This pipeline is also designed with consideration for the requirement of efficiency. The method deals with the large amounts of information contained on one frame, following a hierarchical manner. The later a 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 the keypoint detection step of testing by intensity thresholds. Then in average, 10^3 keypoints are detected, and verified, leaving about 10^2 keypoints to be clustered. Afterwards, fewer than 10 keypoint clusters are left; these are dealt with by the very time-consuming steps of generating image features and tracking.

The advantage of this method is its ability to give promising detection results from cluttered data in real time. In addition, this method successfully combines bottom-up and classification methods, as well as combines both appearance and temporal information.

This chapter is organized as follows. Section 3.2 reviews related work. Section 3.3 introduces the steps of the method. Section 3.4 introduces an implementation of the method on data collected by infrared cameras in tunnels. Section 3.5 gives experimental results for data collected by infrared cameras and gives results by extending the method for data collected by ordinary cameras. Section 3.6 concludes.

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 considering 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 & 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 takes 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 a Hough transform method 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 belong to foreground objects, 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, these other methods 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 that the time complexity of solving a game-theoretic problem is low and this makes it efficient to solve conflicts between trackers. In [Zhang et al., 2008], the problem of maximum a posteriori is embedded to the max flow of a well designed graph. And it is able to recover missed detections on middle frames and works efficiently. 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 results in an expectation-maximization manner in later steps. And the most important concern of these methods is finding efficient solutions in data association problems.

This work is also related to feature grouping methods [Yarlagadda et al., 2010], detecting methods 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 Method Outline

The method is a three-step method. The first step of candidate object localization deals with keypoints. It takes original data as input, and outputs keypoint clusters as detection hypotheses. The second step of candidate object verification takes these keypoint clusters as input, verifies them by their appearance and motion information, and labels them as positive or negative. In the third step of voting along time axis, keypoint clusters connected by the same trajectory will vote for whether the trajectory is positive or negative when the trajectory ends, and this gives the final detection results.

3.3.1 Candidate Object Localization

There are a huge amount of information contained in even one frame of a video collected by cameras. In time-critical tasks, not all pixels in each frame are processed, instead keypoints can be detected, and verified. The appearance of keypoints belong to local features. Still the number of keypoints is large, and association of keypoints along time dimension is not feasible due to the constrain of time consumptions. Clustering the keypoints on the frame is feasible and will indicate the locations of target objects. Thus in this step, keypoints are detected from the input images, verified using appearance model, and then clustered to indicate candidate object locations, as shown in Figure 3.1.

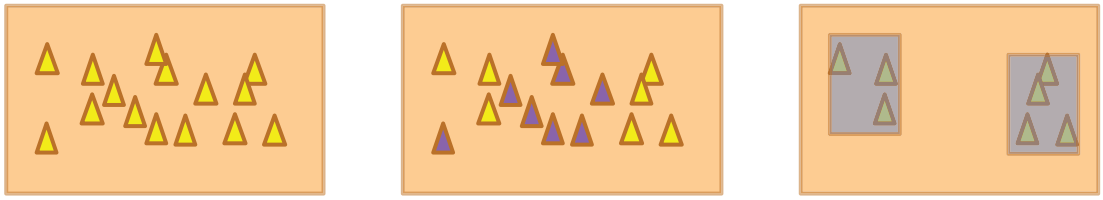


Figure 3.1: Keypoint detection, verification, and clustering. On the left, keypoints represented in yellow are detected on images. In the middle, yellow represents the keypoints pass verifications, while purple represents the keypoints fail to pass verifications. On the right, the keypoints which pass verifications are clustered, and blue rectangles represent candidate objects.

3.3.2 Candidate Object Verification

The number of candidate objects in the form of keypoint clusters is smaller compared with the number of keypoints. Expensive global appearance feature is feasible, and expensive association along time axis is also feasible. By assuming certain motion patterns of the target objects, the trajectories can also be used to verify candidate objects. Thus in this step, candidate objects are verified by appearance and motion, as shown in Figure 3.2.

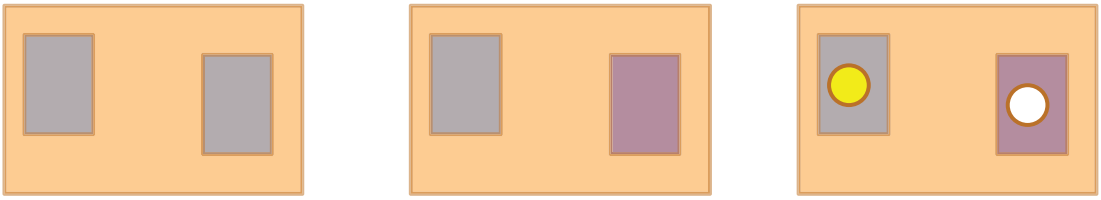


Figure 3.2: Candidate object verification by appearance and motion. On the left, blue rectangles represent the candidate objects given by the previous steps. In the middle, the blue rectangle represents that the candidate object is decided as positive by appearance, and purple rectangle represents negative by appearance. On the right, yellow circle represents that the candidate object is decided as positive by motion, and white circle represents negative by motion.

3.3.3 Voting along Time Axis

Trajectories not only connects the candidate objects, but also connects the local decisions made according to appearance and motion. To refine the results, when each trajectory ends, voting is employed, as shown in Figure 3.3.

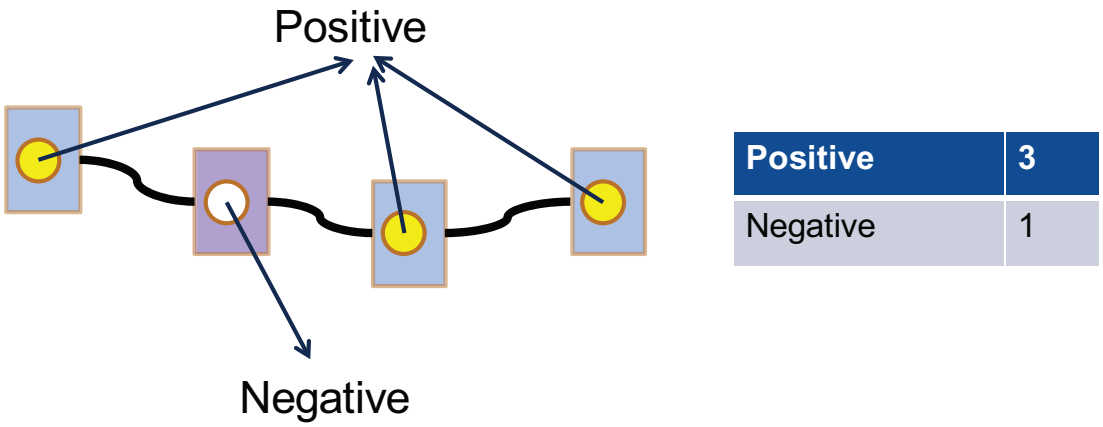


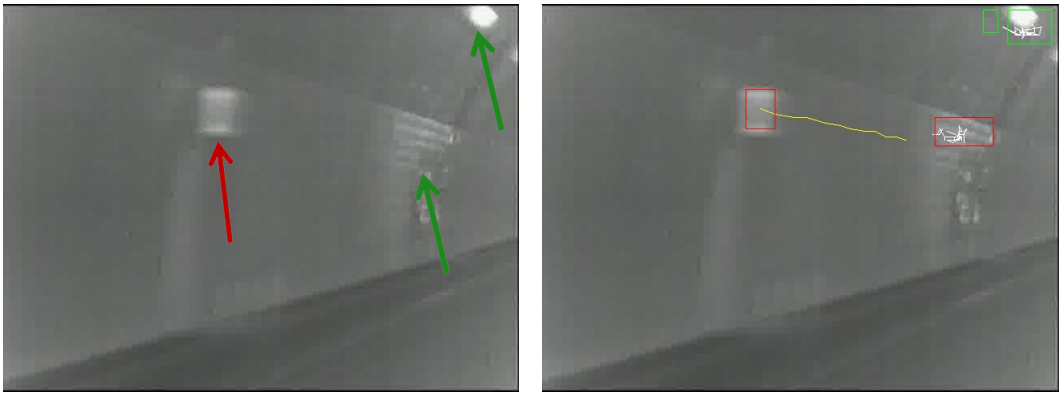
Figure 3.3: Voting along time axis. Local decisions vote along trajectories for positive and negative. In the example, there are 3 positive votes and 1 negative votes.

3.4 Implementation on Data Collected by Infrared Cameras in Tunnels

The method is implemented to detect emergency telephone indicators in tunnels. The implementation aims to perform detection in real time, and serve as an effective unit in positioning automobiles in tunnel environment. In a tunnel environment, in addition to emergency telephone indicators, a lot of noisy objects also appear, e.g. ordinary lights, other vehicles, and other vehicles' shadows. And some of the noisy objects cannot even be distinguished from the target objects by appearance , as shown in Figure 3.4.

Positioning of vehicles acts as a fundamental role in autonomous driving, and is also of great importance for driving assistance, vehicle navigation, etc. When GPS sensors function properly, the task is easy. While in a tunnel environment, there are no GPS signals available for most of the time. A new positioning system which functions properly in a tunnel environment is necessary [Davidson et al., 2008]. An object detection method which can be used for positioning systems in tunnels is here proposed.

This method is part of an automated driving system in a NEDO project, "Development of Energy-saving ITS Technologies". The automated driving system is vehicle-oriented, and an express way is the main application scenario. No specific facilities are assumed to exist on road sides, while instead, the experimental vehicles are equipped



(a) (b)

Figure 3.4: 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.

with sensors and vehicle-to-vehicle communication systems. There are a few sensors used for positioning. For example, sensors used to extract white road lines to estimate the lateral position in the lane, GPS sensors, dead reckoning systems, and stereo far-infrared camera systems intended for obstacle detection. On the street, GPS sensors can be used for positioning. While positioning in tunnels is difficult since GPS signals are not available and no specific equipment on road sides is assumed. For positioning in tunnels, GPS sensors are used to record the position of tunnel entrances, and dead reckoning systems are used to infer position by continuously sensing the vehicles' speed and direction. However, errors of dead reckoning systems will accumulate. Thus, the proposed method uses far-infrared cameras installed on the vehicles to detect signs in the tunnels, which contain position information, and will be used to eliminate the accumulated errors.

In most tunnels on the expressways in Japan, there are many signs appearing at equal intervals. The method focuses 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]. If the emergency telephone indicators can be sensed while traveling in tunnels, and the distance from the

vehicle to the indicators can be estimated, then this information can be used to eliminate accumulated errors of dead reckoning systems. Detection methods, e.g. [Wojek et al., 2010], based on ordinary cameras fail due to darkness. Here the method uses far-infrared cameras, which are suitable in dark environments and are already installed on our experimental vehicles.

In a 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.

3.4.1 Candidate Object Localization

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 this case, keypoint detection is intended to provide hypotheses for emergency telephone indicators. Thus intensity is of great importance. This method employs a simple yet useful method to detect keypoints. Firstly, points are uniformly sampled for an offset of 6 in width, and 7 in height (the length of an emergency telephone indicator is larger than its width). In this manner the magnitude of instances is reduced by nearly two orders. Then points that pass the test, which verifies them by setting intensity thresholds, are considered as keypoints.

Here a 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, the possibility that as few points as possible, belonging to the emergency telephone indicators, are excluded, is also considered. The probability of one point falling into the defined interval based on its belonging to emergency telephone

(a) (b)

Figure 3.5: Keypoint detection.

indicators is given by,

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

Points for which the intensities fall in the pre-set thresholds, are detected as keypoints.

Keypoint Verification

As shown in Figure 3.5(b), the detected keypoints don't just belong to emergency telephone indicators, but also belong to the background. For training purposes, keypoints belonging to emergency telephone indicators are considered positive, all others are negative.

To verify the keypoints, the appearance of the sub-image around each keypoint is used. Intensity histograms are used to describe the appearance. Noisy keypoints not only come from the wall of the tunnel, but also from ordinary lights, other vehicles, and other vehicles' shadows. 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.

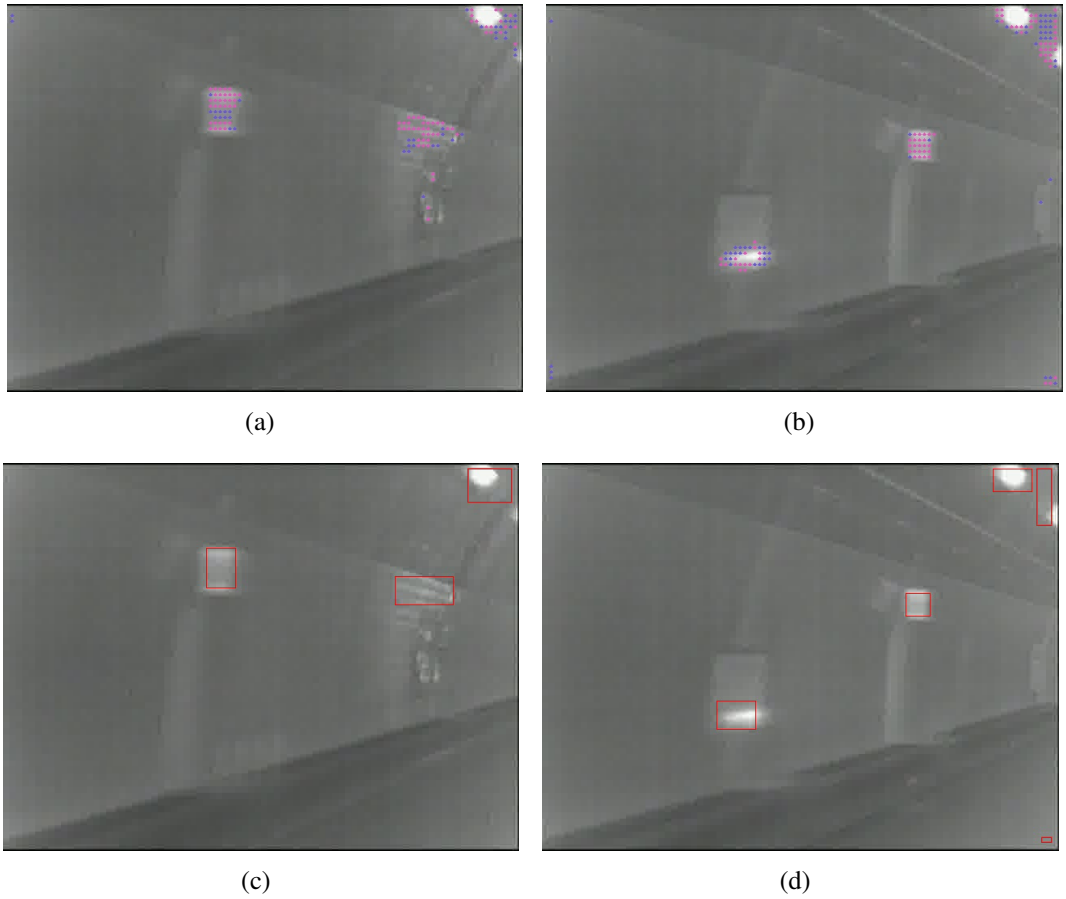
Let $\{C_1^i, i = 1, 2, ..., n_1\}$ denote the intensity histogram centers of the positive keypoints, and $\{C_0^i, i = 1, 2, ..., n_2\}$ the negative. For each C_1, the average Euclidean

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 an evaluation function of the positive feature centers. The positive feature centers are ranked by $Eu(\cdot)$, and the 10 positive feature centers with the 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 negative, otherwise it is considered positive. Here, for simplicity, unlike [Stauffer & Grimson, 1999], the same threshold is used for all components of the mixture.



(a) (b)

(c) (d)

Figure 3.6: Keypoint verification and clustering. Red circles mark keypoints which pass the verification, while blue marks failed ones. Rectangles mark keypoint clustering results.

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 to decide whether the keypoints belong to the emergency telephone indicators. 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 the time dimension is high. So the keypoints are clustered, then data association in time dimension only is needed 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. Then the mst is split by cutting edges larger than a threshold. This results in a grouping of the keypoints, denoted by, $\gamma = \{\mathbf{g}\}$.

3.4.2 Candidate Object Verification

Keypoint Cluster Verification by Appearance

For each keypoint cluster, the smallest bounding rectangle is considered a detection hypothesis, as shown in Figure 3.6(c) and Figure 3.6(d).

In the case of emergency telephone indicator detection, there are three main sources of noise: ordinary lights, other vehicles, and other vehicles' shadows. 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 intensities of their corresponding sub-images in the collected data gets lower. At a certain distance, the intensities of the ordinary lights are almost the same as the intensities of the emergency telephone indicators'. For ordinary lights of which the intensities are higher than the intensities of emergency telephone indicators', the transition regions from the lights to tunnel walls will have similar intensities as the emergency telephone indicators'. This indicates that although locally the emergency telephone indicators share the same appearance with ordinary lights, globally they can still be distinguished by appearance. As for other vehicles and their shadows, their intensity range is very close to the intensity range of the emergency telephone indicators', and they can hardly be distinguished by appearance alone.

At this step, the keypoint clusters are verified by their appearance, ideally excluding

keypoint clusters belonging to ordinary lights. An Adaboost machine is trained using intensity histograms of the emergency telephone indicators and ordinary lights. The appearance of other vehicles is close to that of 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 a 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 7 times as large as the weights of the negative training examples. Since in practice, whether each keypoint cluster is a target object is decided by both appearance and motion information. The difficulties of excluding noisy objects can be left for later steps.

Figure 3.7: Keypoint cluster verification by appearance. Red rectangles: positive detection hypotheses, and green: negative detection hypotheses.

Keypoint Cluster Tracking

Not all noisy detection hypotheses can be excluded by using appearance, as shown in Figure 3.7. To distinguish keypoint clusters belonging to other vehicles and their shadows, the keypoint clusters are tracked through frames to generate trajectories.

In this case of keypoint cluster tracking, the problem 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 by finding the 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 indicate if $n_{\mathbf{g}}^j$ 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 the Hungarian algorithm [Kuhn, 1955] gives a near-optimal solution, we follow a very simple manner for the solution by finding the best matched pairs and excluding them until no matching pairs can be found.

Keypoint Cluster Verification by Motion

As shown in Figure 3.15, the trajectories from keypoint clusters belonging to emergency telephone indicators are different from other objects' trajectories. In this step, the temporal information encoded in the trajectories is used to further verify the keypoint clusters. A linear model is used to fit each trajectory, and the Pearson Correlation Coefficient(PCC) of the fitting is the criteria for the decision.

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 PCC of the fitting is defined as,

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

Where r is used to decide if the trajectories of the keypoint clusters belong to emergency telephone indicators or not.

Object Labels

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

3.4.3 Voting along Time Axis

Each trajectory not only connects the detection responses, but also connects the decisions for 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 of other frames. The final results are based on the trajectories of decisions. When one trajectory ends, if more than 80% of the decisions it connects are positive, then this trajectory is considered positive.

The procedure is as follows: 1) each detection result along a trajectory which encodes local appearance and motion patterns votes for whether the trajectory is positive or not, and 2) if the voting percentage is larger than a threshold, a final decision is made that the object is positive.

3.4.4 Pipeline Summary

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

	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 summaries the steps in the pipeline based on whether the step employs appearance or motion information, whether the step uses online or offline information, and whether the method is discriminative or generative. Generally speaking, methods that employ motion information and discriminative methods are employed at later steps. This is because the computational cost is high for motion information, and discriminative methods need more information to guarantee performance. Online information are also mainly used in later steps, this is because without information provided by models trained offline, there are no instances to generate online information. Also the computational cost of online information is higher since all calculations are on the fly.

3.5 Experimental Results

In this section, this method is tested based on detection performance and efficiency. Two experiments and their results are reported in this section.

3.5.1 Dataset One

Data

To collect data, infrared cameras are mounted on top of the experimental vehicle, and then we take several tours of the Awagatake tunnel. In the first experiment, Approximately 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.

Implementation Settings

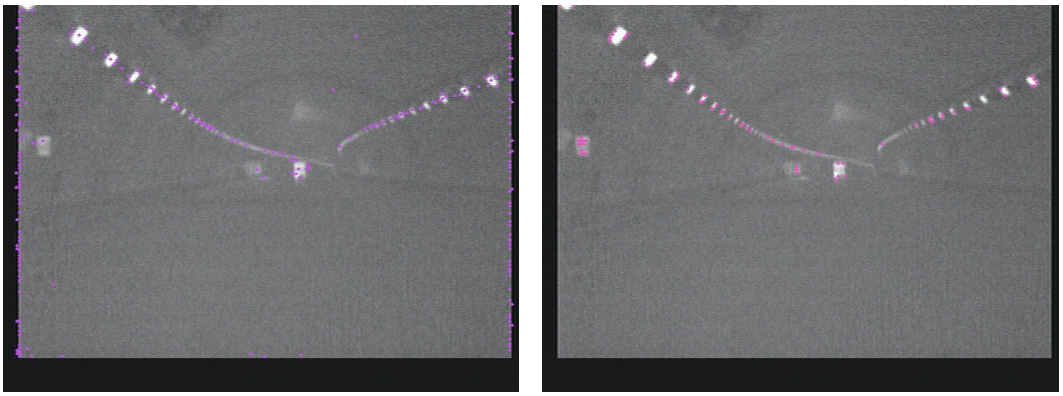
All models are trained using data from one tour, while evaluated on data from another tour. Firstly, all emergency telephone indicators are marked in the form of rectangles on all frames from the training tour.

To set intensity thresholds for keypoint detection, a Gaussian distribution is assumed for the intensities of points belonging to target objects. Following the 3σ principle, $I_{\mathbf{x}}^{th1}$ is set to 160 and $I_{\mathbf{x}}^{th2}$ to 190. 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, are used to estimate $I_{\mathbf{x}}^{th1}$ and $I_{\mathbf{x}}^{th2}$. In Figure 3.8, we also compare the keypoint detection results with the results using SIFT.

The approximate sub-images of the emergency telephone indicators are manually marked, and used for training the mixture model of keypoint verification. The detected keypoints falling into the sub-image are marked as positive, all other points are marked negative. Note that this model and the training method may not be very accurate, since more accurate marking requires more manual effort.

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. By using these k values, both feature sets are over-segmented. The threshold to verify keypoints $D_{A_{\mathbf{x}}}^{th}$ 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.



(a) (b)

Figure 3.8: 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.

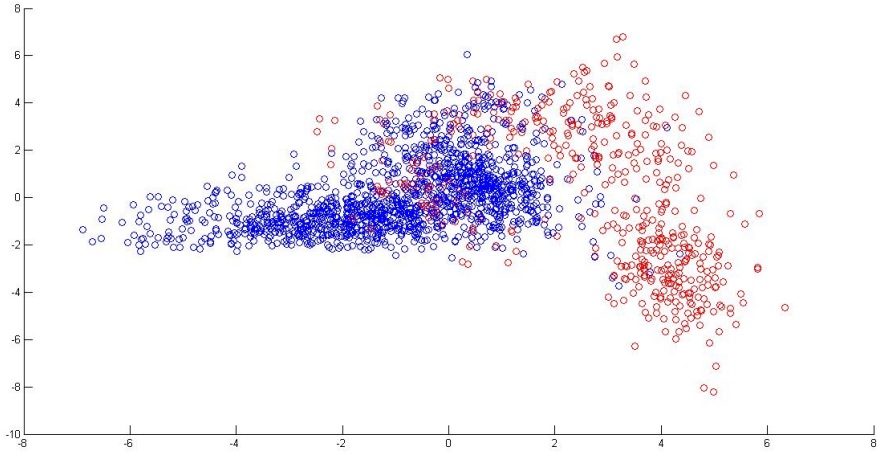
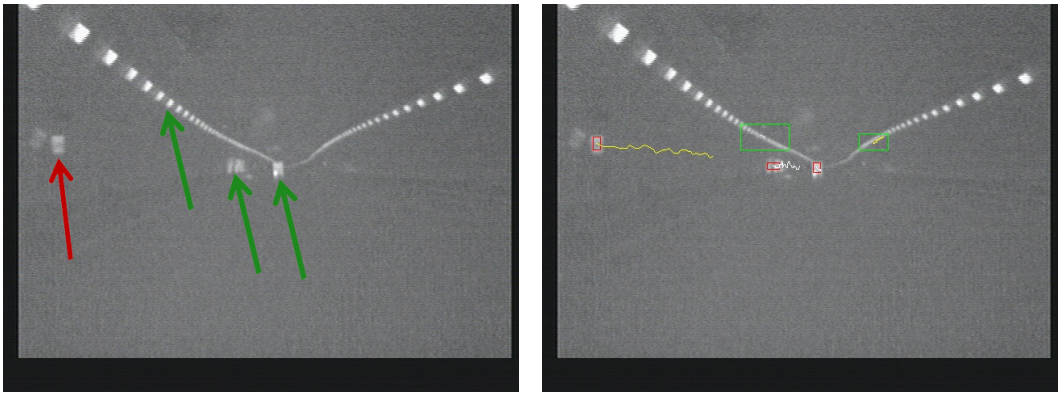


Figure 3.9: Dimension reduction of intensity histograms manually marked as positive and negative, which have 2 dimensions by principle component analysis. Blue circles: positive, and red: negative.



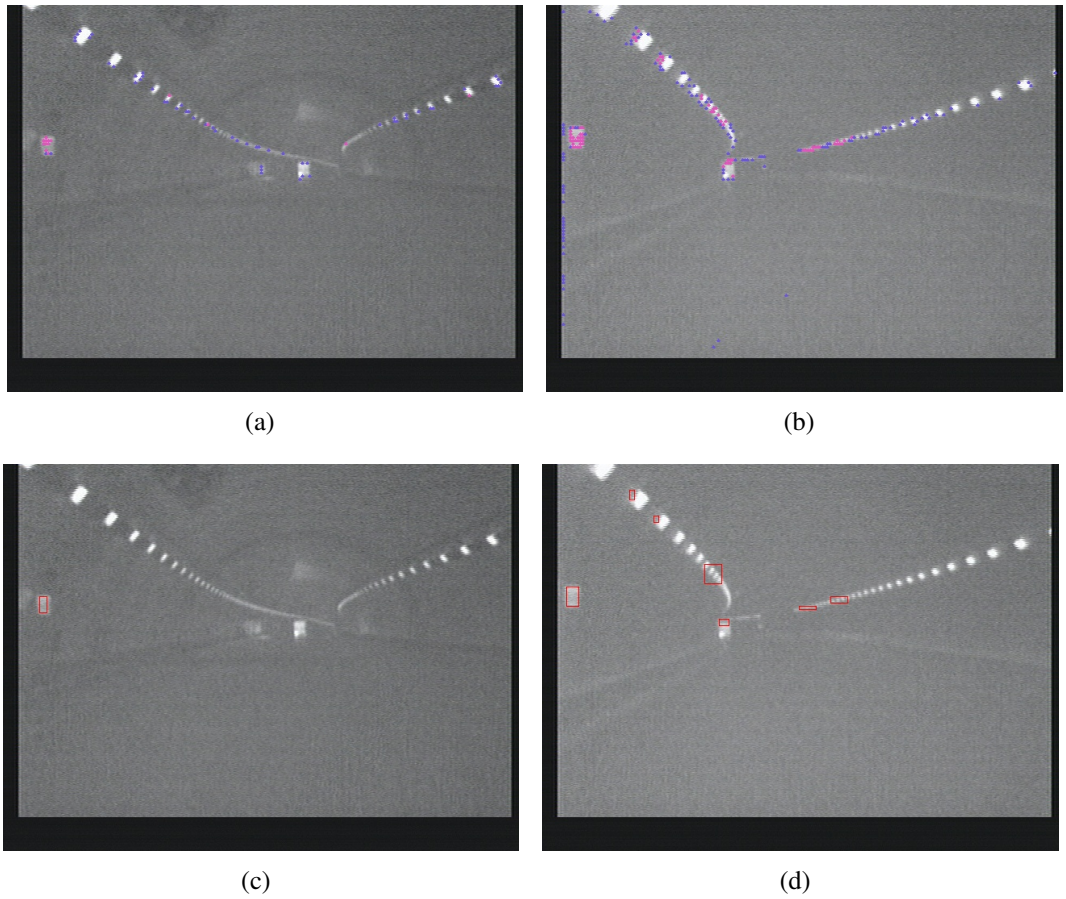
(a) (b)

Figure 3.10: Original data and detection results. Red arrow points to the target object in 3.10(a).

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 3.9, 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 3.10, the original data and sample detection results are given. In Figure 3.11, the results from keypoint verification and clustering are given. In Figure 3.12, the results from keypoint cluster verification by appearance information are given. In Figure 3.13, the results from keypoint cluster tracking are given. And at last, in Figure 3.14, the final detection results are given.



(a) (b)

(c) (d)

Figure 3.11: Keypoint verification and clustering.

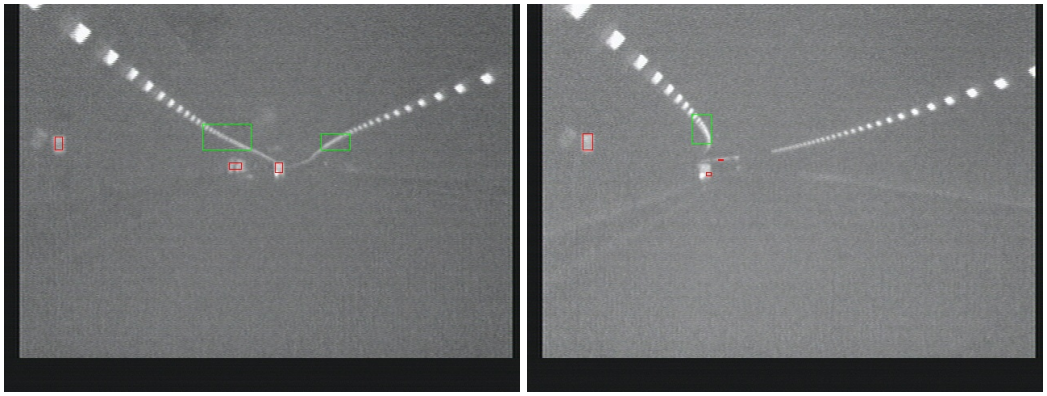


Figure 3.12: Keypoint cluster verification by appearance. Red rectangles: positive detection hypotheses, and green: negative detection hypotheses.

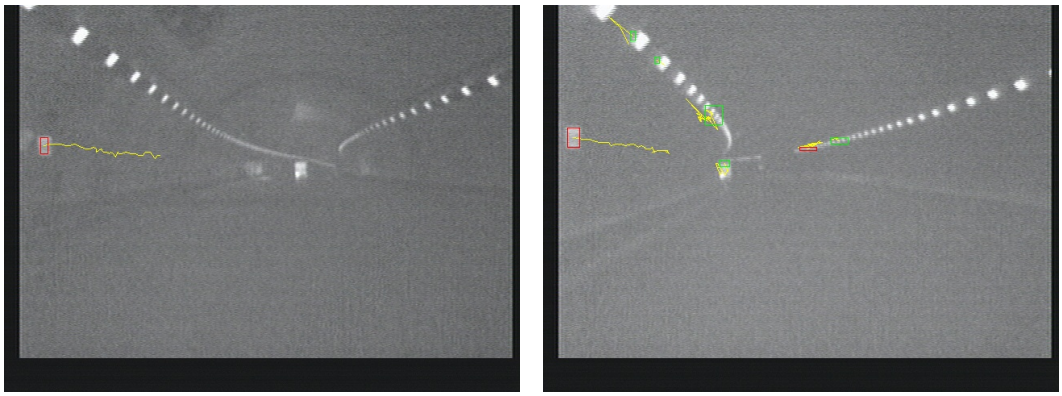


Figure 3.13: Keypoint cluster tracking.

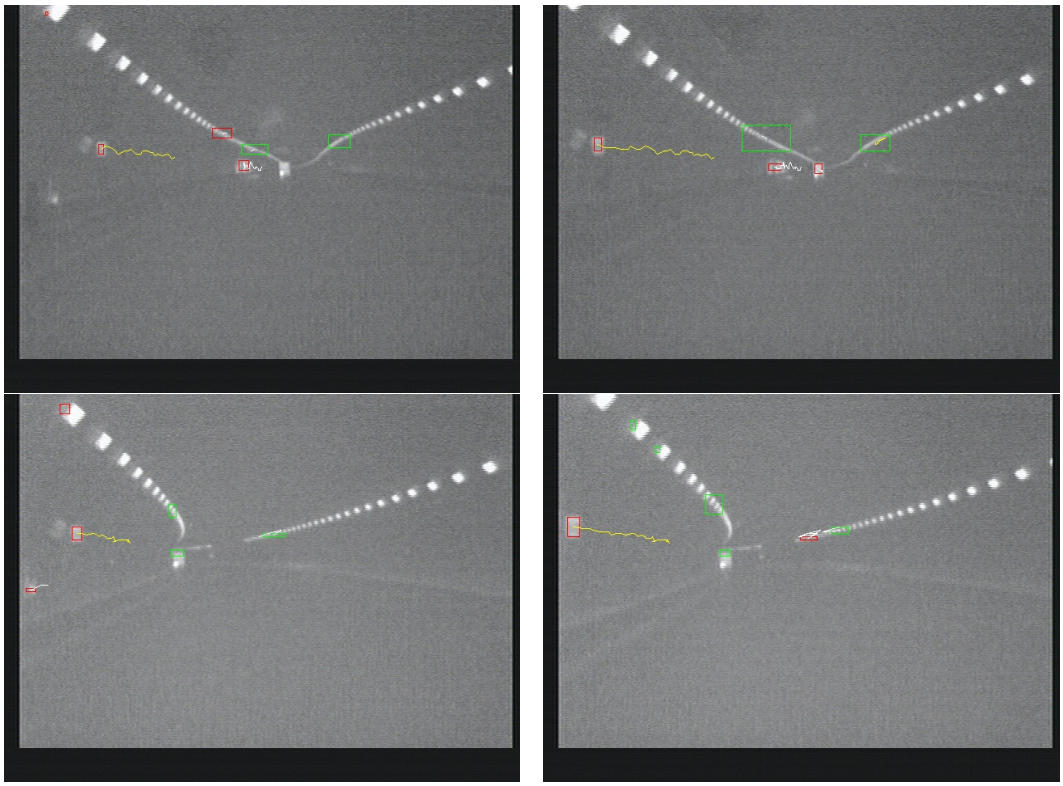


Figure 3.14: Detection results.

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.

Table 3.2: Detection rate and false alarm rate.

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

3.5.2 Dataset Two

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 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

Being the same with experiment one, all models are trained using data from one tour, and evaluated on data on another tour.

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 used to distinguish other vehicles and their shadows is trained by intensity histograms of positive keypoint clusters and negative keypoint clusters. We manually mark positive and negative keypoint clusters. If the Adaboost machine is trained by averagely weighted training examples, its correct rate on the training examples is overall 84%. When trained using this bias weighted training examples, its correct rate is 94% for the positive training examples, and 77% for the negative training examples. 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.

The main difference with experiment one in the pipeline here is how the Adaboost machine is trained. The same weights are assigned to positive and negative training examples during training the Adaboost machine in experiment one, 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 filtered out by a more powerful later 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.

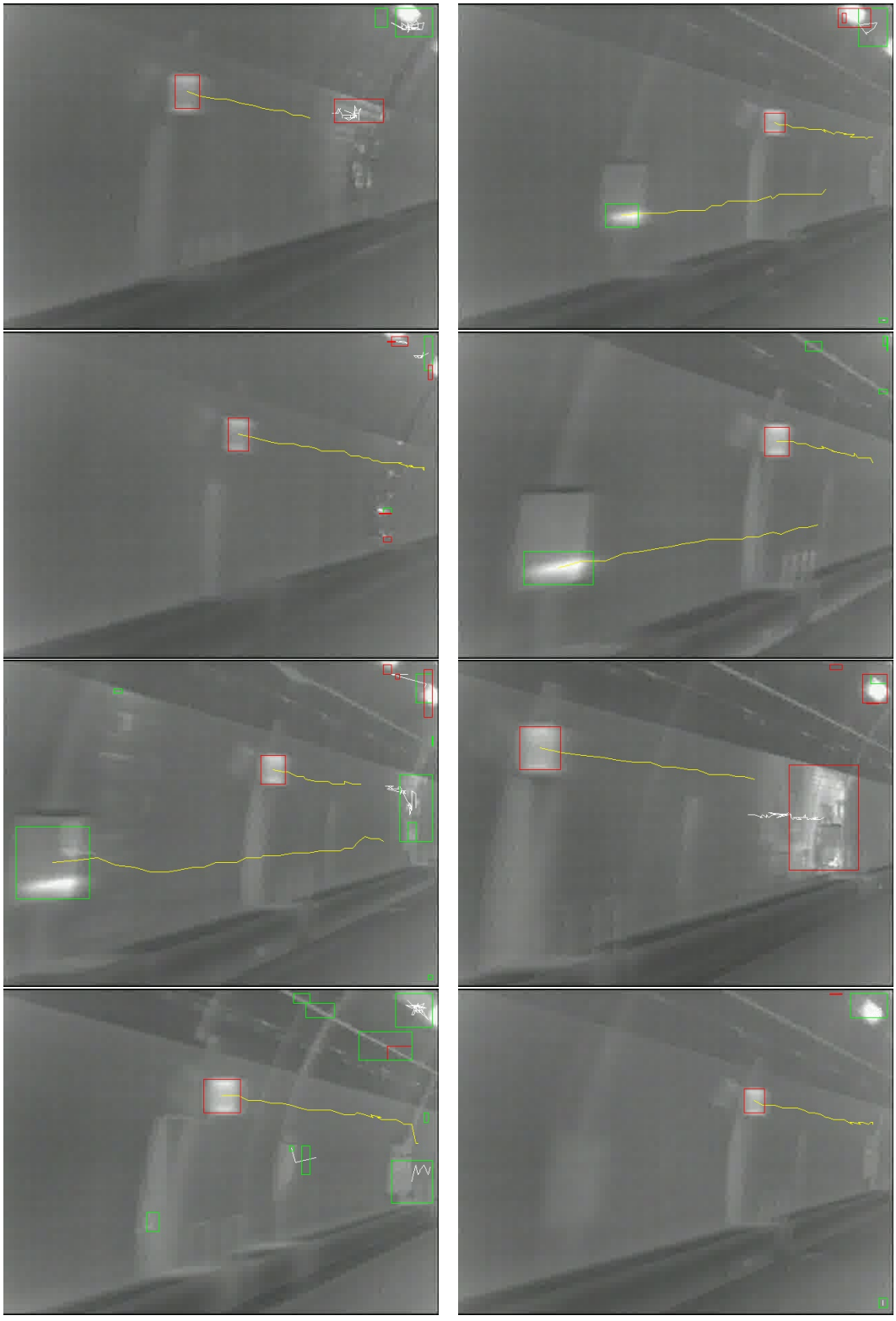


Figure 3.15: Detection results.

Detection Results

Using 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 3.15.

Table 3.3: Detection rate and false alarm rate.

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

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 better than experiment one, because the sensed images are much clearer, and because this more effective training of the Adaboost machine.

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 last 15 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.

Table 3.4: Final detection rate and false alarm rate.

Total number	22
Correctly labeled	22
Miss detections	0
False alarms	0
Detection rate	100%
False alarm rate	0%

3.5.3 P-campus

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

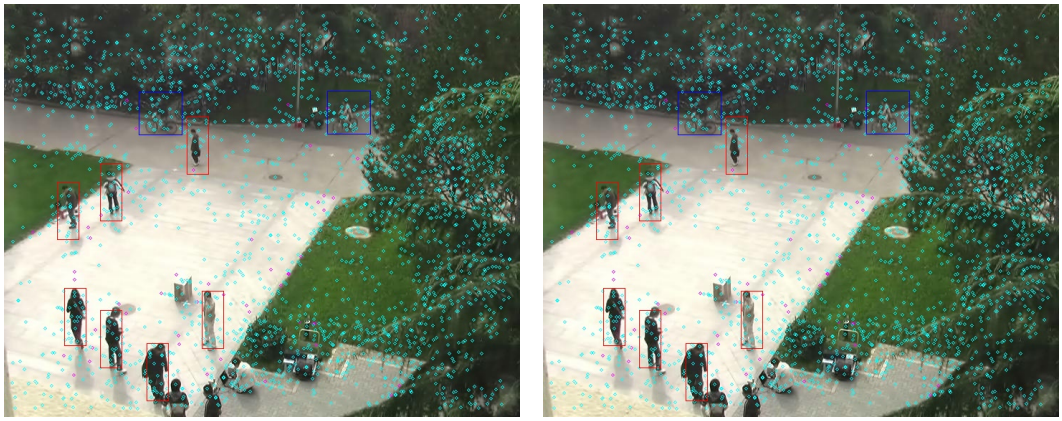


Figure 3.16: Results of keypoint verification. Rectangles mark the manually marked ground-truth boxes. Pink points mark the points labeled as negative in keypoint verification step, others mark keypoints of the positive.

keypoint detector and descriptor.

A new keypoint verification step is proposed. Keypoints from training examples are clustered using k-means as a mixture model. A 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 3.17. 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 3.16, 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 3.17. And this leads to infeasible later steps.

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.

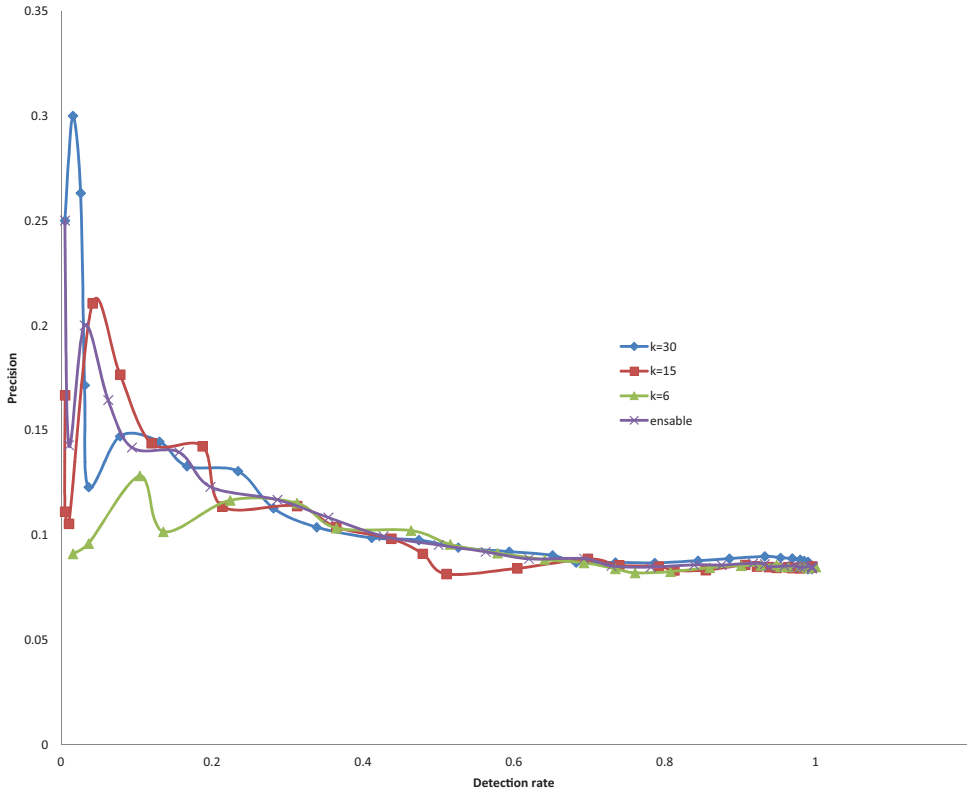


Figure 3.17: Evaluation of keypoint verification. Keypoint verification performance of k-means mixture models with different parameters. Here, k is the main parameter, while the model of ensemble uses all mixture models.

3.6 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%.

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

Common Fate Hough Voting

4.1 Introduction

The reason that the method proposed in Chapter 3 fails to work in complex scenarios, while performs well in detecting simple objects, is that its model is simple. Pedestrians and bicycle riders are actually structured, and different parts of them have different appearance models. For detecting complex structured objects, the method in this chapter assumes different appearance models for different object parts, and makes use of motion information.

Effective video-based detection methods are of great importance to intelligent transportation systems (ITS), and here we propose a method to localize and label objects.

The method is able to detect pedestrians and bicycle riders in a 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. Our method embeds the principle in an Implicit Shape Model (ISM). In our 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 vote belonging to the same object part is assigned a weight according to its consistency 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 our method performs better in both position and label

estimations. Experiments show the effectiveness of our 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 methods, decisions of whether a target object exists or not are made for part of or all of the sub-images in a test image. Besides the attractive performance and being capable of combining various kernels, these methods are favorable because they consider each object as a whole during detection. However, they share limited aspects with visual perception in human beings, and their 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 provide confidence of the locations being the potential objects' centers. Locations of objects are decided according to the converged confidence. These methods are favorable for their robustness to partial deformation and ease of training. To human beings, this kind of method seems to be more natural. And in our work, we combine a mechanism of visual perception in humans, 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 of keypoint descriptors [Leibe et al., 2008], image patches [Gall & Lempitsky, 2009; Okada, 2009], or image regions [Gu et al., 2009]. Each code not only encodes one object part's appearance, but also its offset to the object center and the class label. During the detection step, object parts are detected on each test image. Then every object part is matched against the codebook, and several codes nearest in appearance are activated. 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 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 the case of 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 pools the features which belong to each successive detection response. In Hough transform based methods, a similar mechanism is also employed in [Barinova et al., 2010], however, this effort is employed after the forming of a Hough image. During the forming of a Hough image, two kinds of votes make detection challenging: (1) votes cast by object parts from near objects make the peaks corresponding to different objects mixed up, and (2) votes cast 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 the training images' background part to the codebook. During training a very clean codebook can be built with the foreground marked, which requires manual efforts. Otherwise, a large amount of training examples are needed for the effectiveness of the codebook, and this decreases efficiency.

In videos, motion information is also available by simple tracking of object parts. Thus we propose 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 that 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 our 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. Using 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 the object parts which are more "agreeable" within the motion group. This results in votes corresponding to true detection responses to be more likely 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).

Due to the combination of motion analysis results and the Hough transform frame-

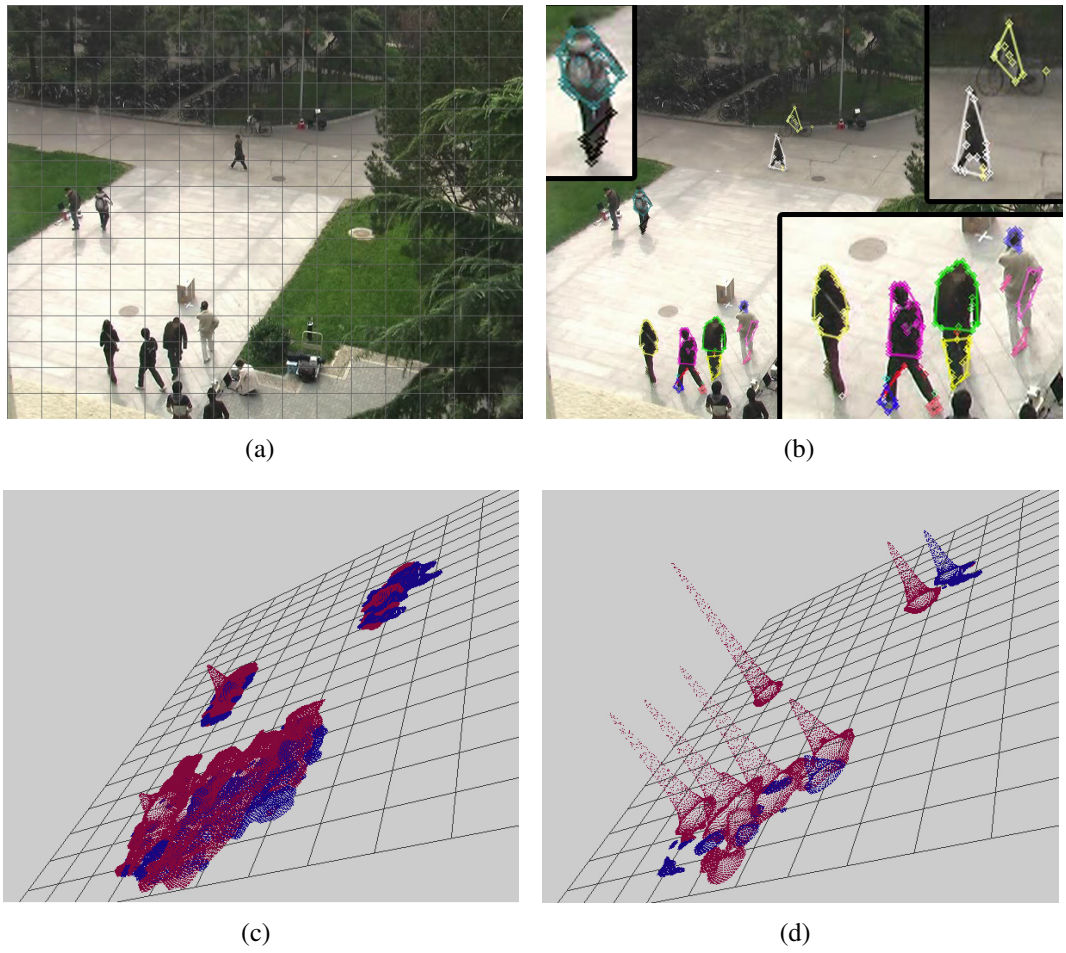


Figure 4.1: Merit of the proposed method. (a) Original image. (b) Motion grouping results. Some parts are enlarged to show details. (c) Original Hough image. (d) Hough image formed using this method. The grids in (c) and (d) correspond to the grids in(a).

work, and by assigning different weights to each object part's votes, the proposed method has several appealing properties:

- The method's ability to estimate object position and label 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 used to combine grouping results of object parts is very general, and thus can be easily expanded.

The remaining chapter is organized as follows. Section 4.2 reviews related work. Section 4.3 gives potential application background. 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. Section 4.7 summarises.

4.2 Related Work

This method is most closely 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 are making 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 the methods [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 between the voted centers, local features are first grouped, resulting in a

more consistent vote 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 any real entities.

The method is also very related to detection or recognition methods in videos [Dalal et al., 2006; Javed, 2005; Levin, 2003; Nair & Clark, 2004; Noguchi & Yanai, 2009], especially those employ motion information [Andriluka et al., 2008; Gavrila & Munder, 2007; Grabner et al., 2008]. The methods of [Andriluka et al., 2008; Gavrila & Munder, 2007; Grabner et al., 2008] improve both detection and tracking performance by coupling the two problems. Motion information of these methods are used at global level of the objects. Also efforts are made to propose online learning/adaption for the detectors [Javed, 2005; Levin, 2003; Nair & Clark, 2004; Yang et al., 2013]. These methods often suffer from model drifting. Differences of motion in the form of 2D or 3D [Brostow et al., 2008] are often used as cue to indicate the existence of objects. The main information used by background substraction [Bouwmans et al., 2009] is the difference of local appearance caused by motion, and this servers as detectors for ordinary tracking methods. Especially, [Jones et al., 2003] considers appearance and motion combination in such scenarios. Motion information can be also combined into features, which are usually used for pose recognition [Chen et al., 2010]. And the method of [Brostow & Cipolla, 2006] considers about using the differences and similarities between trajectories for detection. There are also some methods [Cutler & Davis, 1999] modeling the motion of an object class, and try to detect motion patters, in order to indicate objects. These methods are limited to particular objects. The proposed method is different with these method in the manner of how motion information is used. Motion information is generated at local level and in an online manner without any prior motion knowledge. The proposed method does not adapt its detector in the aspect of building new appearance models, instead it helps with the detector of appearance. Also the method does not use the appearance difference caused by motion or detect particular motion patterns, instead, it grouping the object parts robustly by motion, and incorporate the grouping result into a voting system.

The work is also related to keypoint clustering methods. In [Estrada et al., 2009], SIFT [Lowe, 2004] is used as keypoint feature, color histograms are used as appearance

feature, region covariances [Tuzel et al., 2006] as texture feature, and (x, y) coordinates as spatial feature. Then for a pair of keypoints, 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 corners [Tomasi & Kanade, 1991] are extracted as keypoints and motion information is gained by employing KLT tracker [Lucas & Kanade, 1981] to trace out the extracted keypoints 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 Euclidean distance between every point pair. The other captures the changing quantity between two trajectories by calculating the variance of the Euclidean distance among all pairwise point distances. The clustering procedure is all about making pairwise decisions on whether to merge two smaller clusters or not in a Minimal Description Language manner.

This work is also related methods that 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], an 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 provides psychological evidences for 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 the main targets. Currently pedestrians are also considered as important subjects of ITS applications, and bicycles also are becoming very popular for environmental and money-saving reasons. In Japan, the number of traffic accidents among bicycles and pedestrians is very large. Thus we tackle an issue of 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 method focuses on techniques from the area of computer vision for detection under surveillance scenarios. It is also assumed by the method that target objects captured by the surveillance cameras do not change much in scale.

4.4 Common Fate Hough Transform

Probabilistic standpoints are very appealing because of inference ease. However, as pointed out in [Lehmann et al., 2011], placing an Implicit Shape Model (ISM) in a probabilistic framework is not satisfactory. Especially, describing weights of the votes as priors does not make sense. A Hough transform can be simply considered as the transformation from a set of object parts, $\{\mathbf{e}\}$, to a confidence space of object hypotheses, $C(\mathbf{x}, l)$. Where $\mathbf{x}$ is the coordinate of the object center, and 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. This section describes how a Hough image for the 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 is 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 $\mathbf{x_V}$.

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 an $\tilde{l}$-class object is given by,

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

A uniform weight is assumed for each object part, and $w(\mathbf{e}_j) = \frac{1}{M}$. 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. Assume $\mathbf{e}_m \in \mathbf{g}$ and $\mathbf{e}_n \in \mathbf{g}$. Those votes of $\mathbf{e_m}$ which are more "agreeable" than 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 based on $V_{\mathbf{e}_n}$ and the confidence that $V_{\mathbf{e}_m}$'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 based on $\mathbf{e}_n$, and the confidence that $V_{\mathbf{e}_m}$'s voted center is correct, as,

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

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 excluding 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 should 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:

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

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 determine $w(\tilde{V}_{\mathbf{e}_m})$, 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 used to form the Hough image are the iteratively converged weights.

The grouping result $\gamma = \{\mathbf{g}\}$, can be replaced by grouping results based on other information, for example our 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 similarities.

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 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 in which both trajectories exist.

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 minimum spanning tree is then built using the calculated similarities. The built minimum spanning tree 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. Max operation is used in the definitions of both D_1 and D_2. This is helpful because very often two trajectories are of different lengths, and under such situations, max operation will have better stability than other operations, e.g., average, that consider only overlapping frames.

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 obtained 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 portion, and the entire 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. This will handle scale changes of a small range, since the six image patches are not of the same scale. And the method's ability of handling scale changes is limited. So it can only be used in surveillance situations where the scales of target objects change in a limited range.

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 as 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,

$$\arg\max_{u_i,v_{ij}} \sum_{i=1}^{O} u_i \ln(\sum_{j=1}^{M} v_{ij} C(\mathbf{x}_{H_i}, l_{H_i}; \mathbf{e}_j))$$

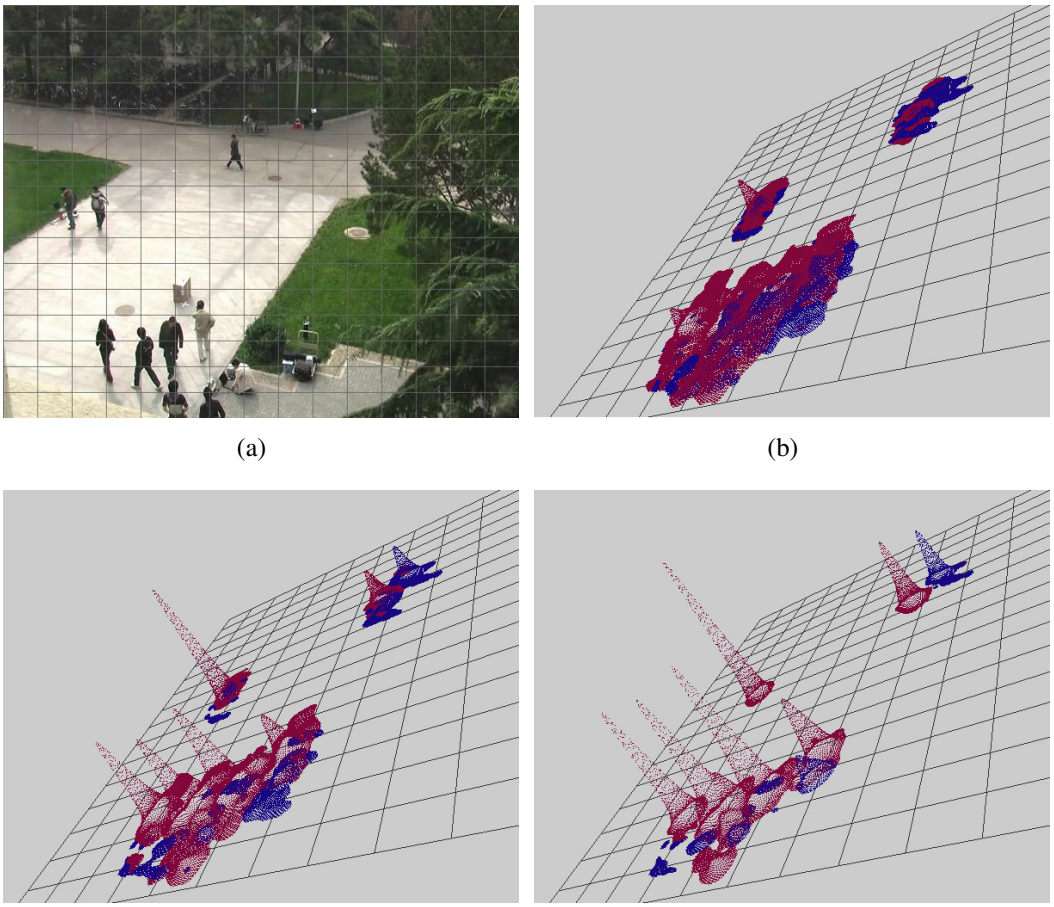
$$s.t.: \; u_i = 0 \text{ or } u_i = 1, \forall\, i;$$

$$v_{ij} = 0 \text{ or } v_{ij} = 1, \forall\, i, \forall\, j;$$

$$\sum_{i=1}^{O} v_{ij} = 1, \forall\, j;$$

$$\sum_{j=1}^{M} v_{ij} \leq u_i, \forall\, i\ .$$

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 when the confidence of the chosen object is lower than a given threshold.



(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.

Algorithm 4.1 Greedy Maximization

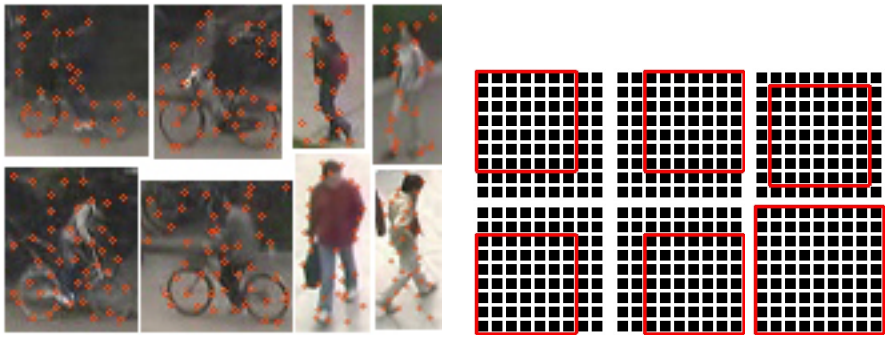
Let ε be the set of object parts, C_{th} be the low confidence threshold to accept detection responses, and $\hat{\mathbf{h}}$ be the local maxima of $\mathbf{h}$

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 $\{H_i\}$

4.6 Experimental Results

In experiments, advantage 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 a benchmark, and then tested on a dataset of several animals.

4.6.1 P-campus



(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.

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 bicycle riders and 171 pedestrians are randomly selected from the marked ground truths. Harris corners are detected on these randomly selected training images, 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 Figure 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. Testing data is formed by the 79 frames, on which the ground truth bounding boxes are marked. Harris corners are detected, and region covariances are generated in the same manner as for the training images. For each Harris corner on one testing image, the corresponding region covariances are matched against the codebook for the most similar codes. Some of the training examples will appear in the test sequences. The emphasis of this experiment is to verify the proposed

Figure 4.5: Motion grouping results.

framework's ability of combining motion information. Both the proposed method and the benchmark method use the same training and testing images, so the comparison is fair and proves the effectiveness of the proposed method.

For motion grouping, each keypoint is tracked through 10 frames before, and 10 frames after the current frame. The similarity of two 21-point trajectories is defined using only the frames in which both trajectories exist. 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 so that it is larger than only 10% of the measured D_1s, and so is D_{th}^2. By doing this, 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.

In order 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

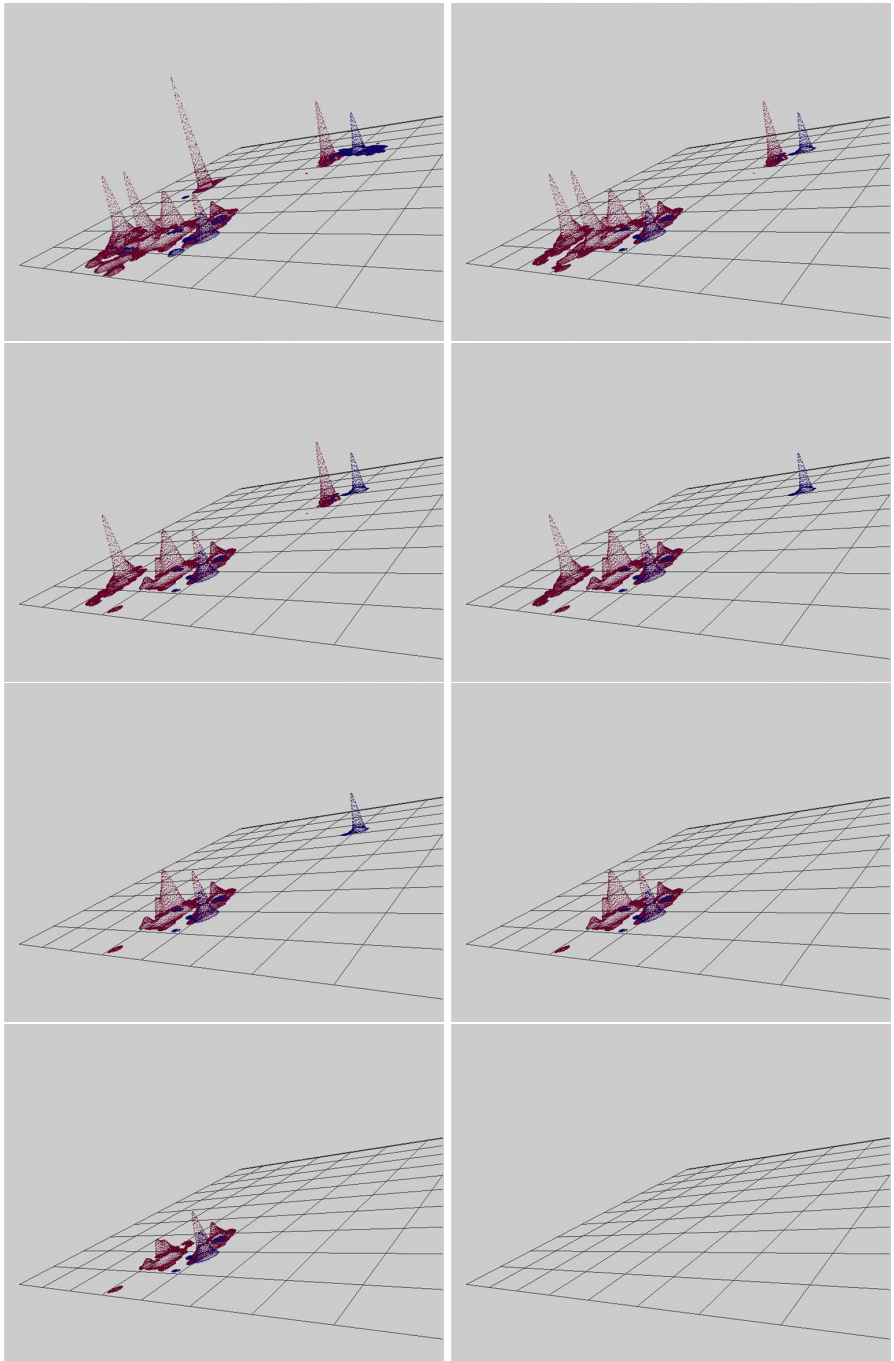


Figure 4.6: Inference procedure.

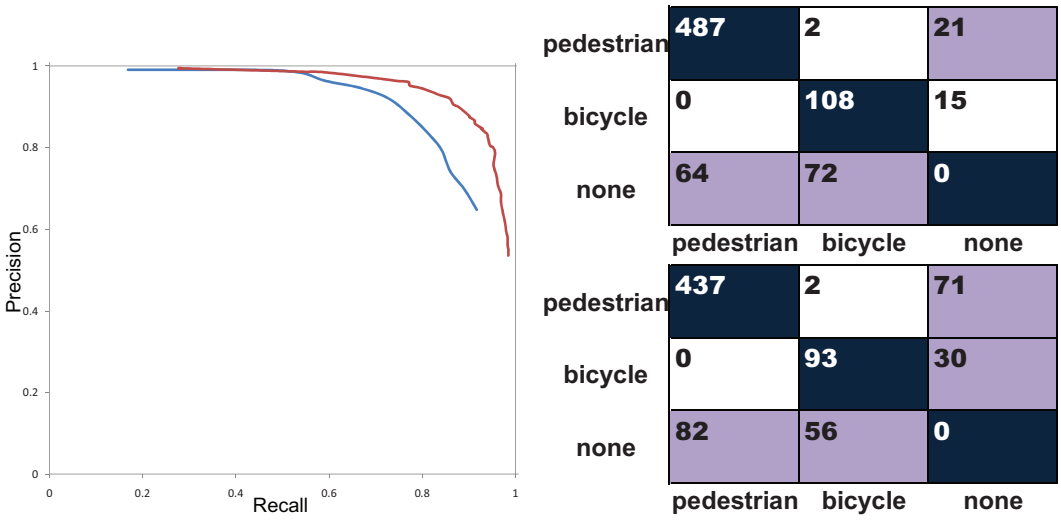
curves are based on σ, while d is set to 10. Here σ is the most important parameter.

Comparisons

For comparison, detection is done on the Hough images formed with and without motion grouping results. The same codebook and the same parameter 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, the proposed method is compared with the benchmark method when the two methods have a 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.



(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).

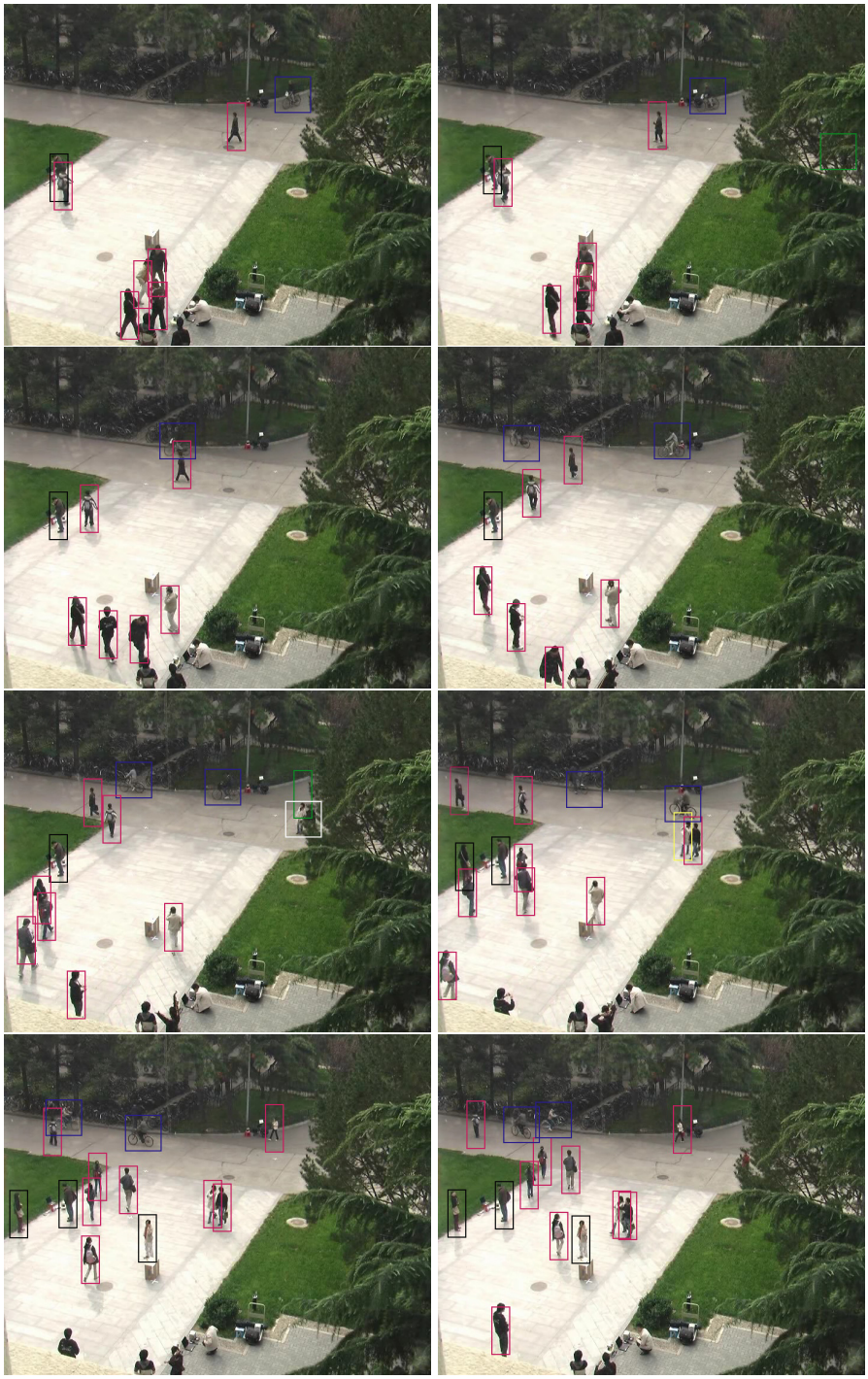


Figure 4.8: Results. Red rectangles and blue rectangles mark correctly detected pedestrians and bicycle riders. Yellow rectangles mark missed detections. White rectangles mark correctly detected but not correctly labeled objects. Green rectangles mark false alarms. Black rectangles mark static objects,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

4.6.2 Wild-scene

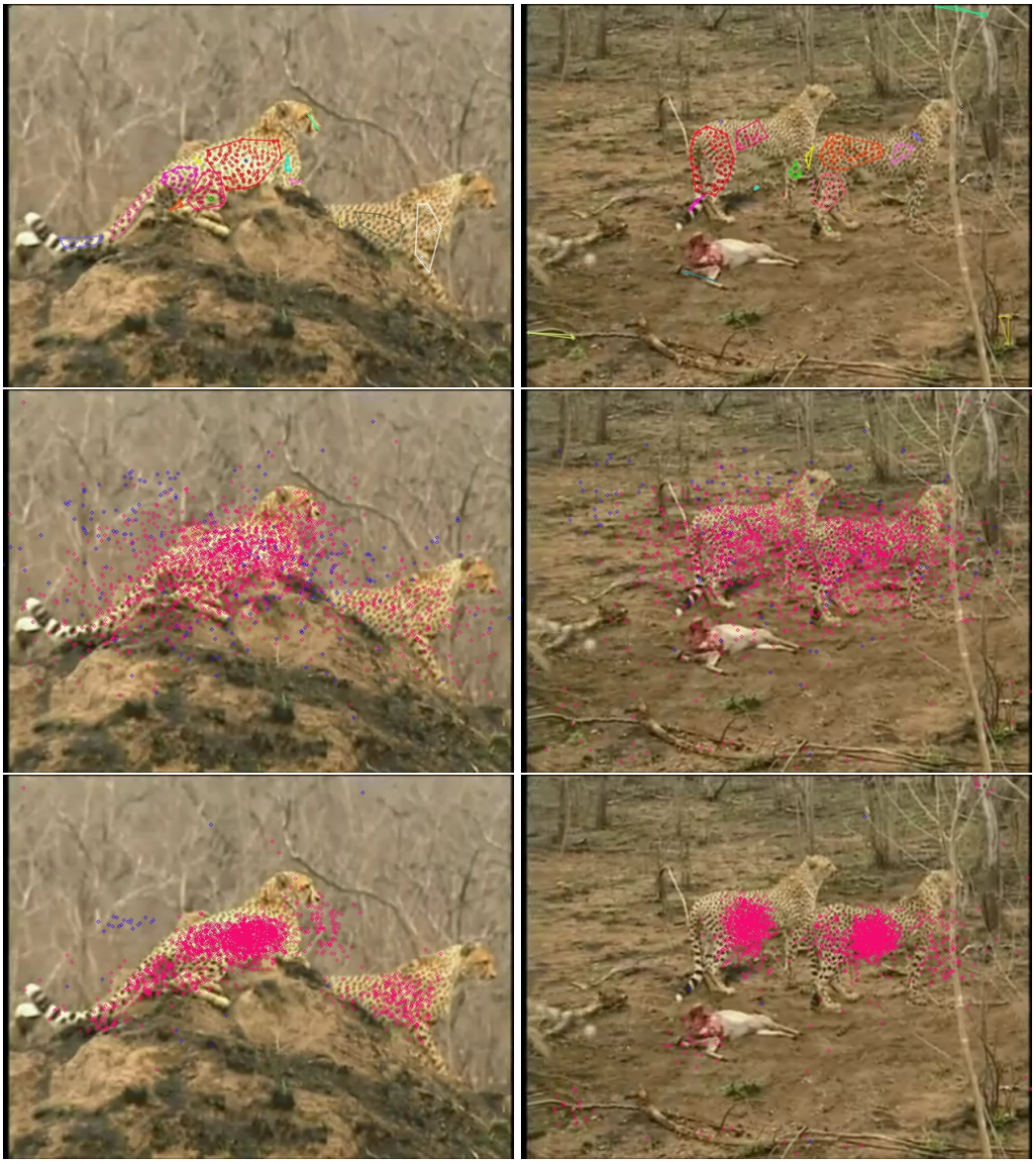


Figure 4.9: Effect of the proposed weight assignment. Red circles are voted center for leopards, while blue ones are voted centers for tigers. On the top are the motion grouping results. In the middle are the voted centers according to the best matched codes. On the bottom are the voted centers voted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

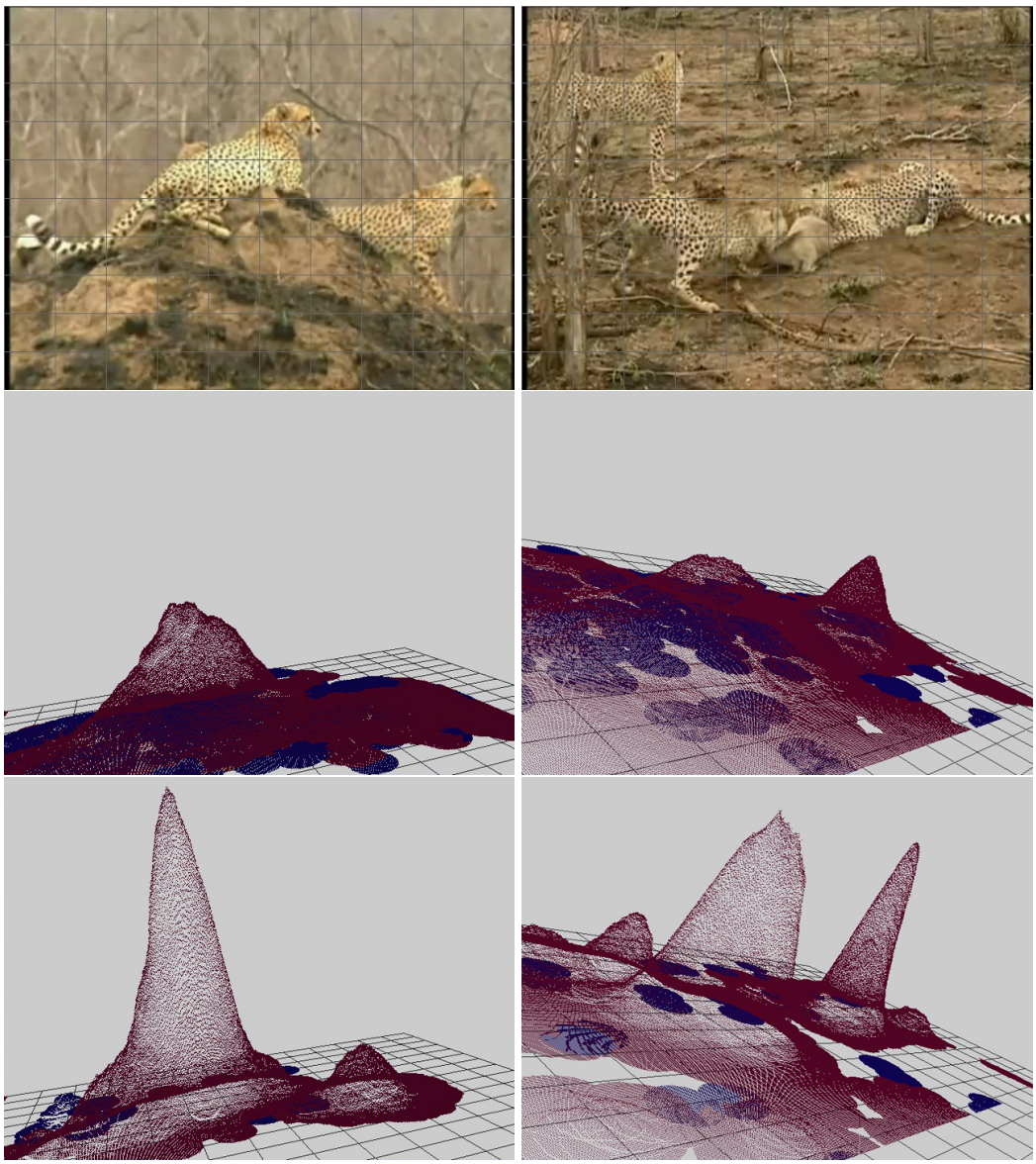


Figure 4.10: Example Hough images. On the top are the original images. In the middle are the Hough images formed by votes with uniform priors. On the bottom are the Hough images formed by votes with the proposed weights. Red indicates leopards, and blue indicates tigers. Note for the two leopards, there 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.

Dataset

In order to show that our method can be used for general purposes, we test our method on complicated scenes, especially, complicated background. Even in these cases, our method works well, which shows robustness of our method. A mini dataset is built upon leopards and tigers of the family Felidae. Note especially that the image feature used by this method belongs to the type 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 of the animals are in the side view.

Implementation settings

Most implementation settings are the same as the settings used 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 using 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 4.11: 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

Fast Voting by Pyramid Match

5.1 Introduction

Bag-of-features [Jégou et al., 2010; Li, 2005] 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 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 principal 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 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 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.
- Time complexity of detection 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 compares time complexities between the proposed method with other methods, discusses about the insights of the method, and explains why the proposed method is effective. Section 5.6 concludes this chapter.

5.2 Related Work

Detection methods still mainly follow the sliding-window schema or share similar structures with methods based on Hough transform. While the focus of the later is to infer about object status by use each online feature as a query against a well-trained codebook. These methods fail to consider target objects as a whole at the beginning. The problem of sliding-window schema 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 is information which are not made use of in images. The positional information is not fully made use of, even of the method of [Shawe-Taylor & Cristianini, 2004]. While [Fergus et al., 2003] provides a method to model the relationship between object parts, there are too many parameters to estimate in their model, which requires a 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 capacity of model will not affect performance, and this 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 expensive in human power. The successes of HOG [Dalal & Triggs, 2005] on pedestrians benefit from its capability 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 a 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 these methods for optimization, and it is capable of representing information in a more complex form. 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 that 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 extensions 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 evolutions include evolving of object part number, 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 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] and methods combining sliding window with Hough transform [Ohba & Ikeuchi, 1997].

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 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

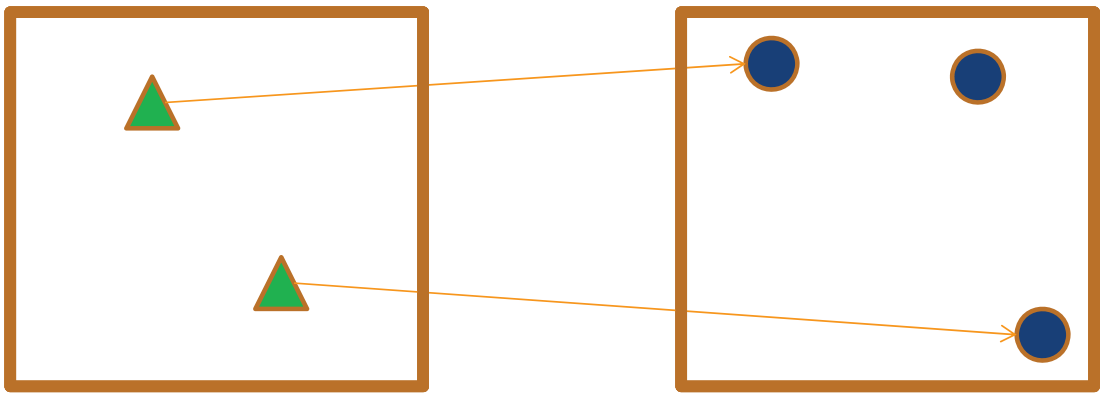


Figure 5.1: A best one-one match problem in 2D space. There are two points in the first point set, three in the second point set. The arrows show correspondence between the two point set.

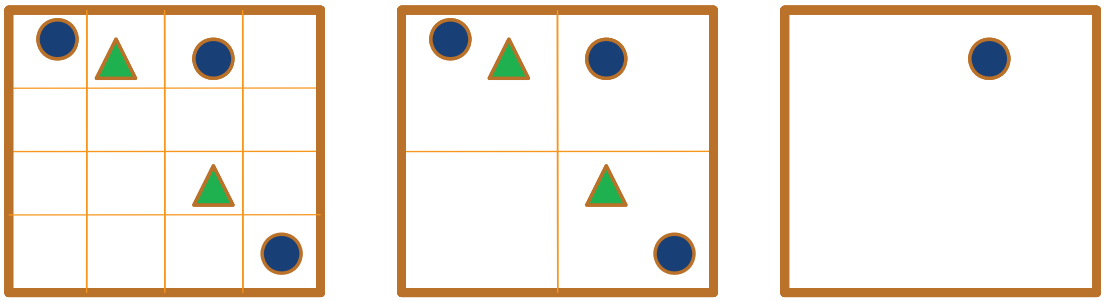


Figure 5.2: Pyramid matching procedure which takes the 2D one-one match problem in Figure 5.1 as an example. The pyramid matching method divides the 2D space from fine to coarse in the 2D space. Notice, the triangle points belong to point set one, and the circle points belong to point set two. In the left, each dimension is divided into 4, results in totally 16 grids, and 0 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.

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 numbers 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 follows, 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 to give S_1^1 and S_2^1, and 3) continue until $i = L$ or one point set is empty.

Then how to construct the dividing methods in γ and how to define the corresponding weight, ω_i, for each $\mathbf{g}_i \in \gamma$ are left to be defined. And these also belong to the factors which distinguish the proposed method from [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

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

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 considering 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 belongs 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 as 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,

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

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 Generation of Dividing Method Set

$\gamma \leftarrow \emptyset$, $r \leftarrow 2 \times (l_{max} - 1)$
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.

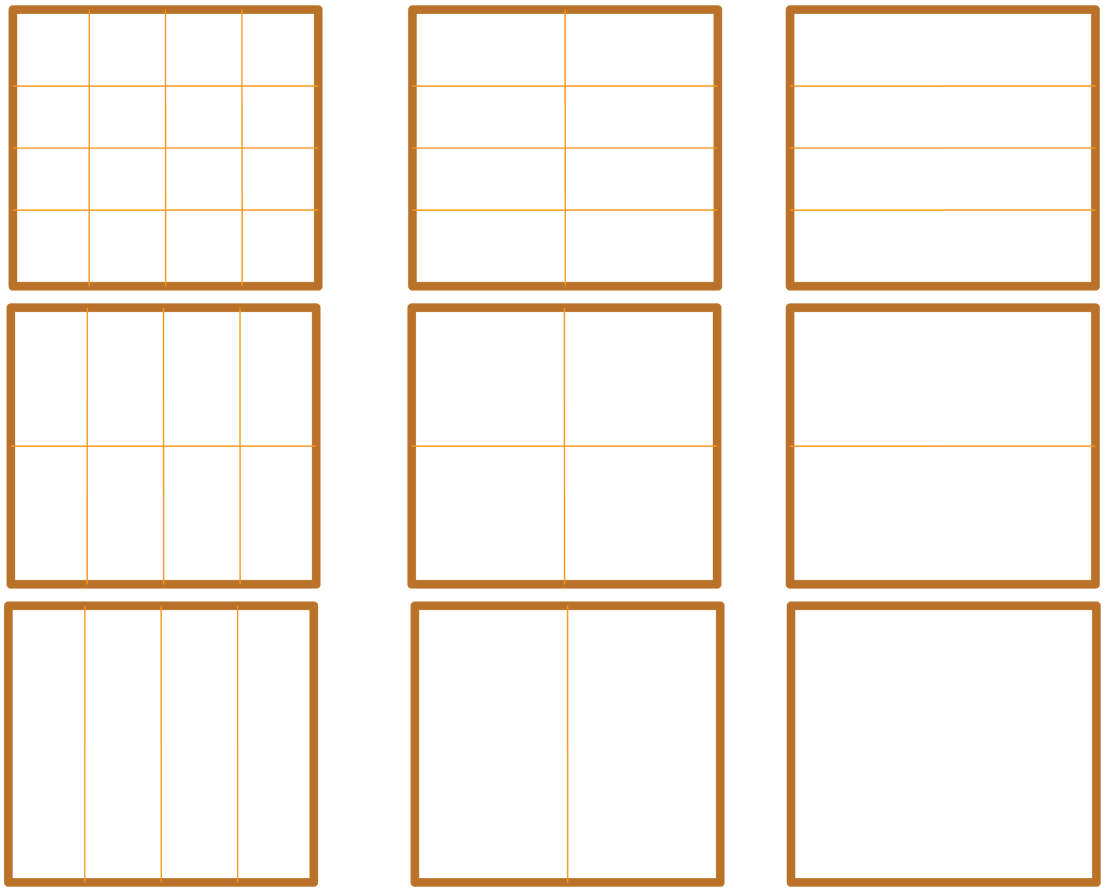


Figure 5.3: An example set of methods to divide the visual-spatial space. $x-$ and $y-$ coordinates represent visual and spatial information respectively. From left to right, the first line is $g(2,2)$, $g(1,2)$, and $g(0,2)$. The second line is $g(2,1)$, $g(1,1)$, and $g(0,1)$. And the third line is $g(2,0)$, $g(1,0)$, 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

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

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,

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

Where,

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

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 and very similar 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 the 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),

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

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,

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

Note, about $\mathrm{PMS}(I_p; a, b, c)$, it is not $\mathrm{P}(S_{I_p}, S_{\mathbf{T}}; \gamma)$, but $\mathrm{P}(S_{I_p}, S_{\mathbf{T}} - S_{I_p}; \gamma)$.

After all positive and negative training examples are given, the objective function will only contain a, b, and c. Thus for such a parameter estimating problem, brute-force solutions will be feasible, especially that $\exp(\cdot)$ can be easily expanded.

Till now, Algorithm 5.1 gives how $S_{\mathbf{T}}$ can be generated, Algorithm 5.3 gives how γ is defined, how ω is estimated is given in (5.4) and (5.7) respectively, so Algorithm 5.2 can be used for detection.

5.4 Experimental Results

The key value of the method is the proposed matric between an object hypothesis and the super template trained from training examples. Accordingly, two experiments are carried on and the results are reported in this chapter.

5.4.1 UIUC Cars

UIUC cars [Agarwal et al., 2004] can be considered as one of the most famous datasets, and it has been used as benchmarks in the area of detection. There are 1,050 training images, of which 550 are positive, and 500 are negative. There are 200 target objects in 167 test images without significant scale changes. In the evaluation following, together with the 200 target objects, 669 negative rectangles from the test images are extracted from testing images for the evaluation.

Performance of the method upon different parameters will be evaluated on the UIUC cars, and compared with DPM [Felzenszwalb et al., 2008]. In the experiments, (5.4) is used. So there are only two parameters left, which will lead to difference in performance, which are, the largest fineness level, l_{max} and the dimension of SIFT after PCA, $(d-2)$.

In Figure 5.4, are some detection results on UIUC cars.

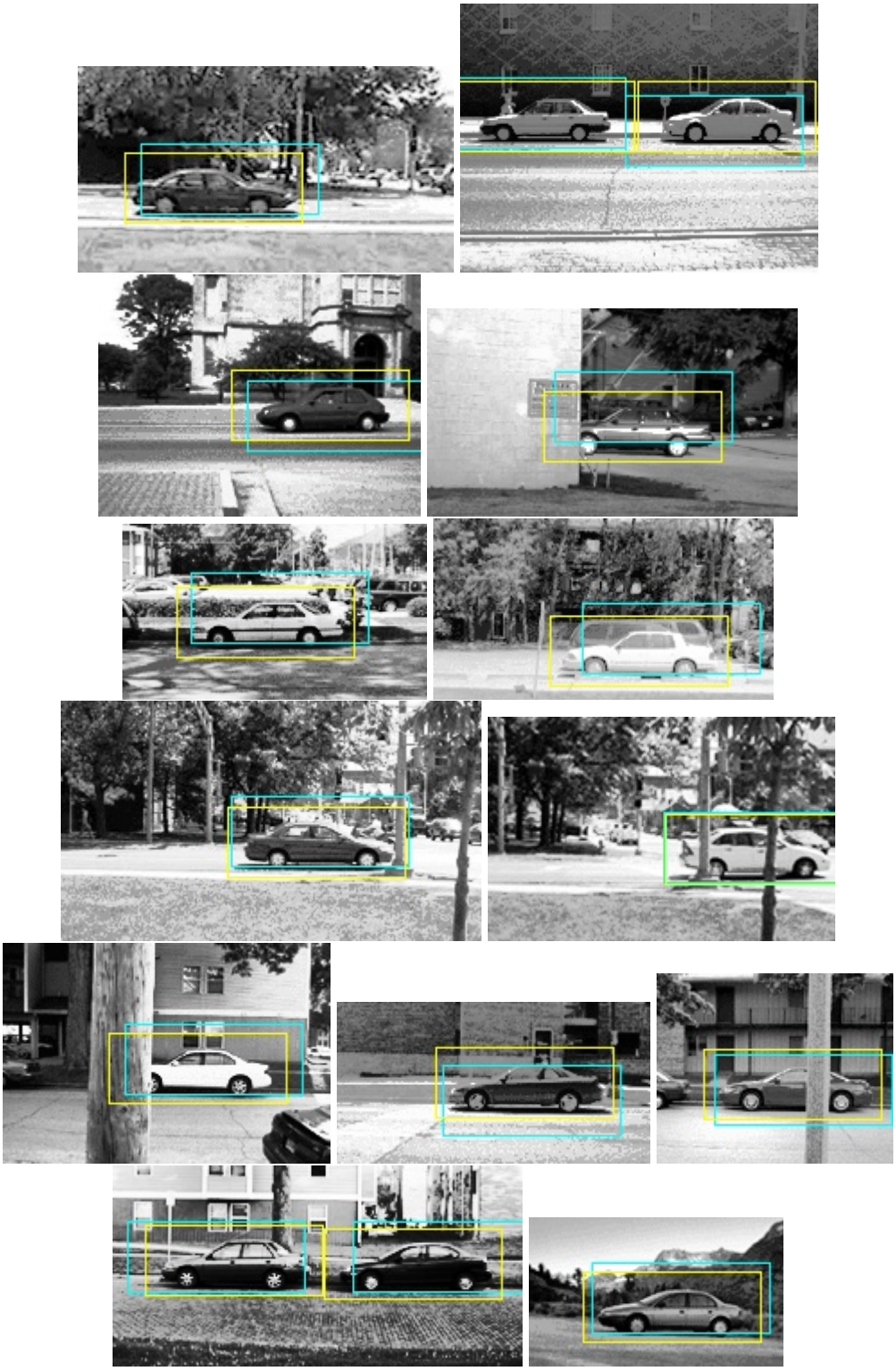


Figure 5.4: Detection results on UIUC cars [Agarwal et al., 2004]. Yellow color marks ground truths, while blue marks detections.

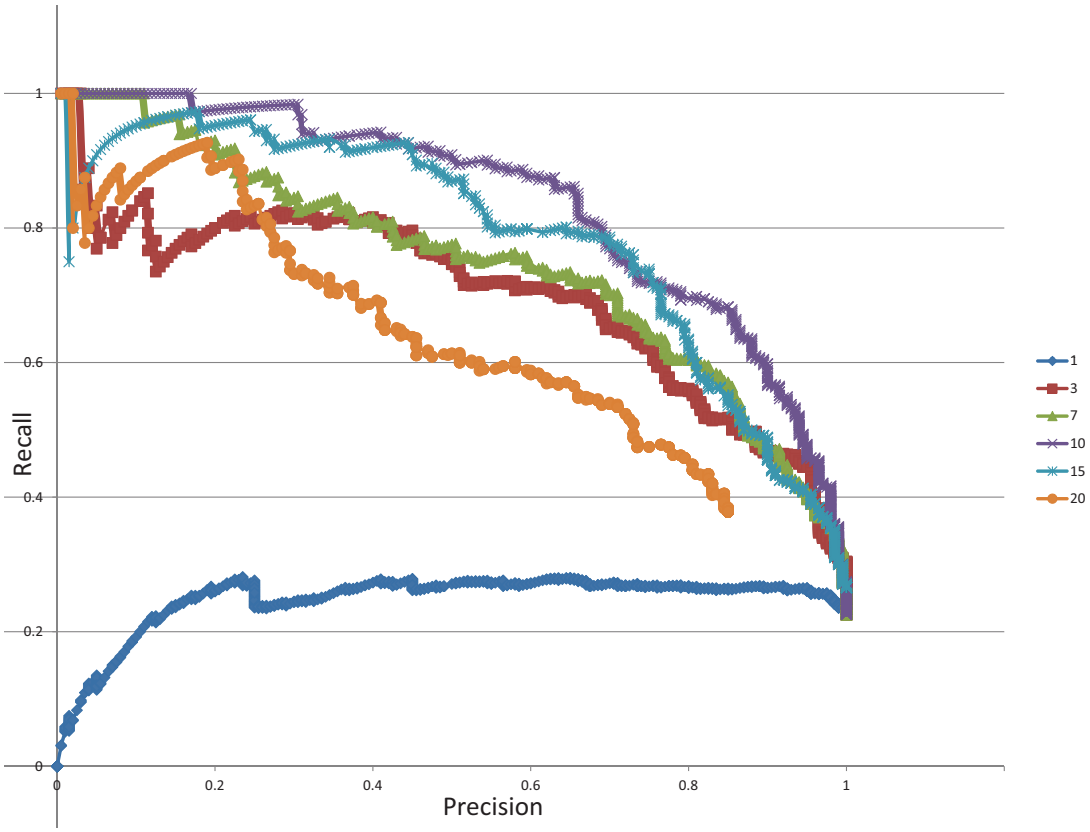


Figure 5.5: Result evaluation on UIUC cars with different $(d-2)$s.

In Figure 5.5, are the results given by using different $(d-2)$s. Precision-recall curves are generated directly by calculating pyramid match score. Obviously, when appearance information takes 10 dimensions, the performance is the best. The result is consistent with [Grauman & Darrell, 2005] and [Liu et al., 2008].

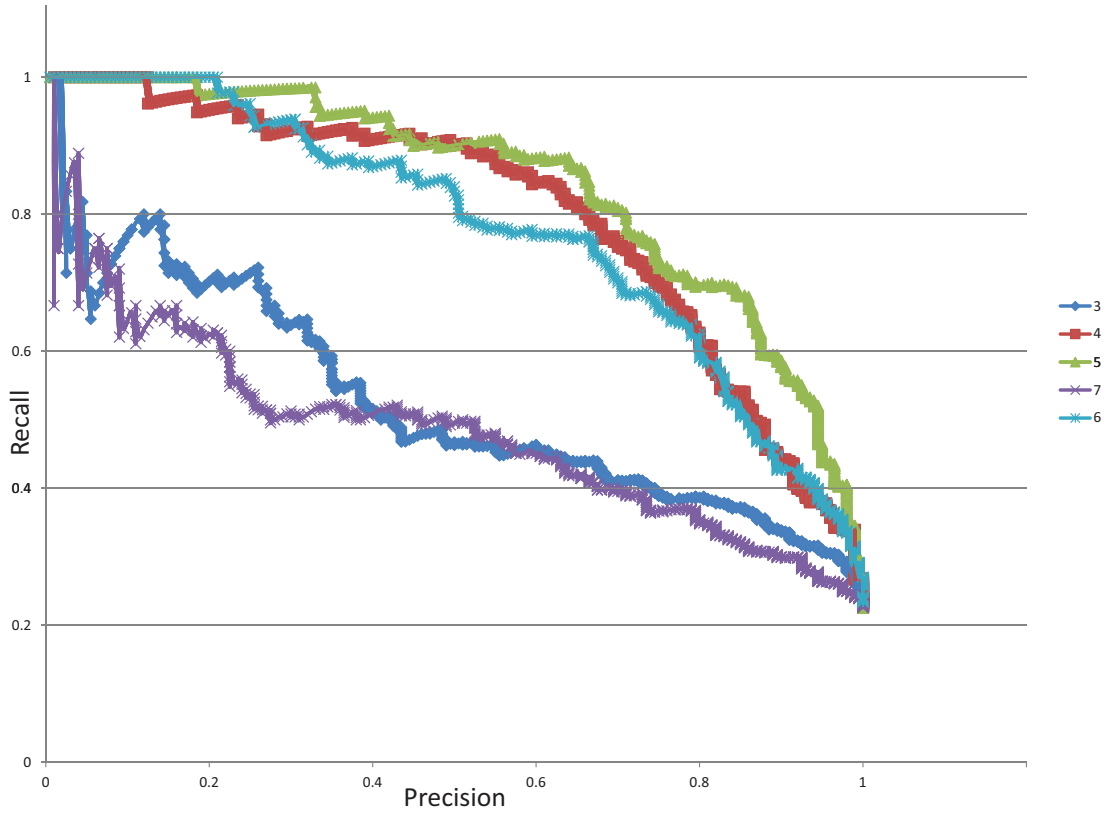


Figure 5.6: Result evaluation on UIUC cars with different l_{max}s.

In Figure 5.6, detection performance by using different l_{max}s is given. Note, the best performance is given at $l_{max} = 5$. The reasons are two-fold: 1) the weights are given by (5.4), and favor larger fineness levels, and 2) both training examples and testing examples are of size 100×40, and $2^5 = 32$ is the best dividing parameter for positional information.

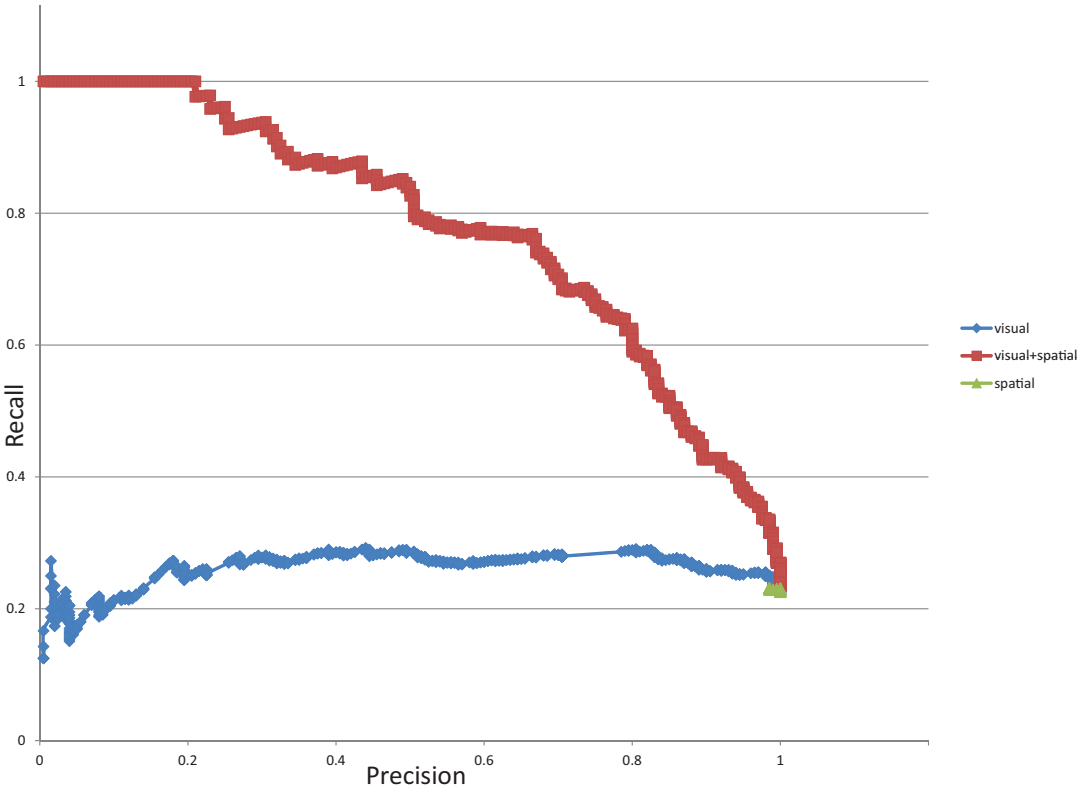


Figure 5.7: Result evaluation on UIUC cars by using visual, spatial, and visual + spatial information.

In Figure 5.7, are the evaluation of visual and spatial information. Actually spatial information alone is not able to distinguish positive and negative target objects at all. The performance of just using visual information is much worse than using visual-spatial information.

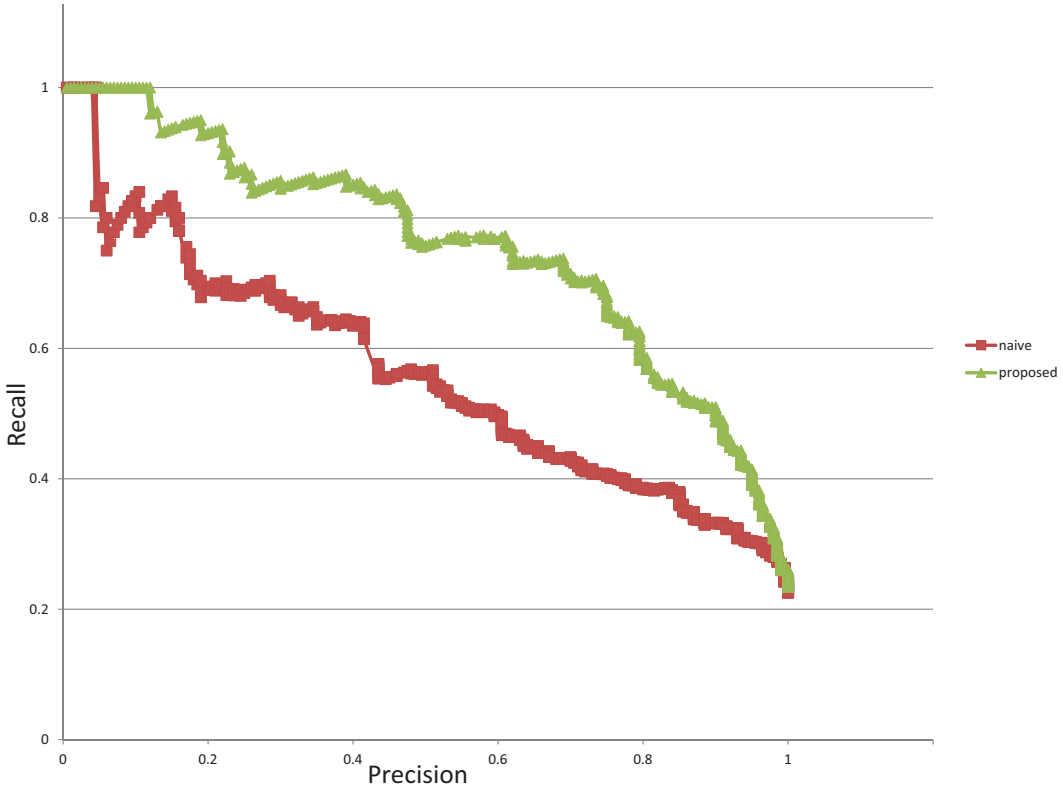


Figure 5.8: Result evaluation on UIUC cars by using γ which is generated using Algorithm 5.3 with $l_{max} = 5$ and $\gamma = \{g(4,4), g(3,3), g(2,2), g(1,1), g(0,0)\}$.

How the visual-spatial space is divided is also critical to the performance of the method, and in Figure 5.8, results are evaluated by using the proposed space-dividing methods and [Grauman & Darrell, 2005].

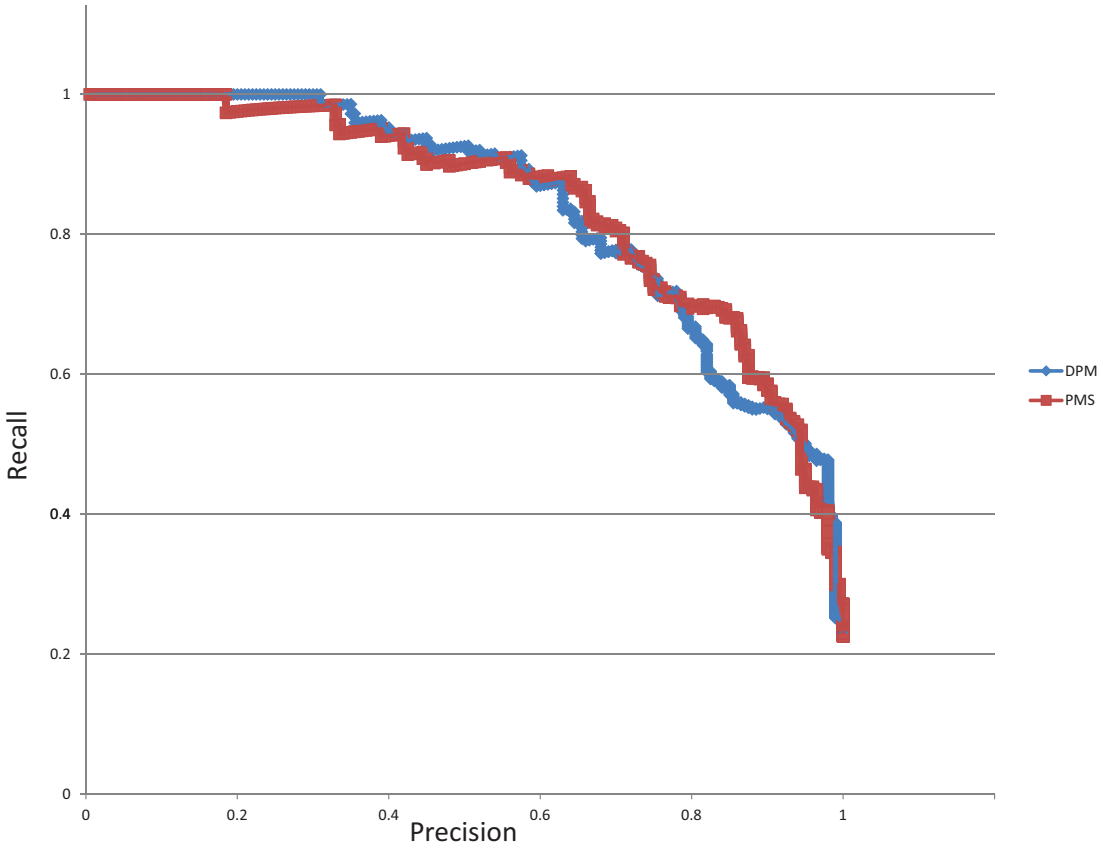


Figure 5.9: Result evaluation on UIUC cars between PMS and DPM.

In Figure 5.9, performance are evaluated between the proposed method, and a state-of-the-art method, DPM [Felzenszwalb et al., 2008]. The proposed method performs no worse than DPM, while the training time between the two models are 1 minute vs 16 hours. In this experiment, the training speed of the proposed method is 1,000 times faster. Also during the training of the proposed method, since (5.4) is used, only the positive training examples are used, while DPM uses both positive and negative training examples.

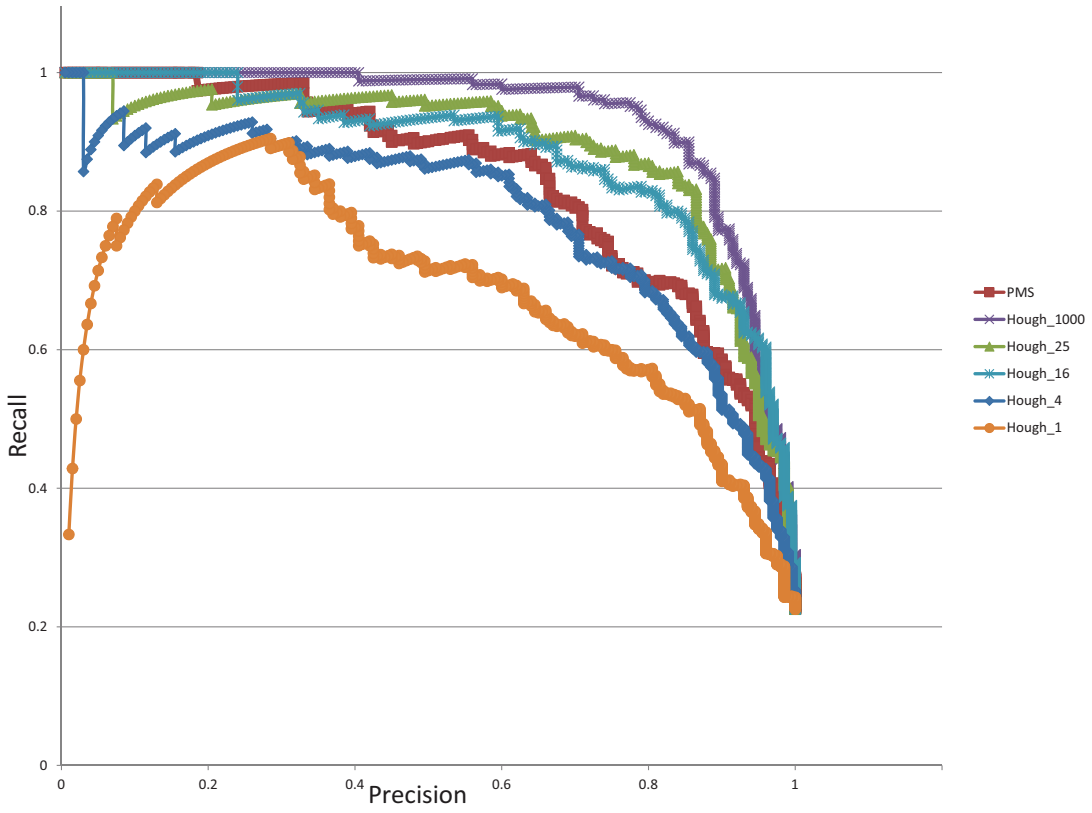


Figure 5.10: Compare the proposed method with different-speed Hough transforms.

In Figure 5.10, performance are evaluated among the proposed method and variants of Hough transform methods. By taking into consideration of the size of the codebook, the original Hough transform method is 1,000 (the indexes in Figure 5.10 mean the times of time consumptions of the variants) time-consuming than the proposed method in giving the confidence of one sub-window's being a target object. Naive k-means method is employed to accelerate the original Hough transform method during

training by condensing the size of the codebook. The original Hough transform method performs better than the proposed method. When the time consumptions are the same, the proposed method performs much better. And the proposed method performs better than the variant which is 4 times time-consuming.

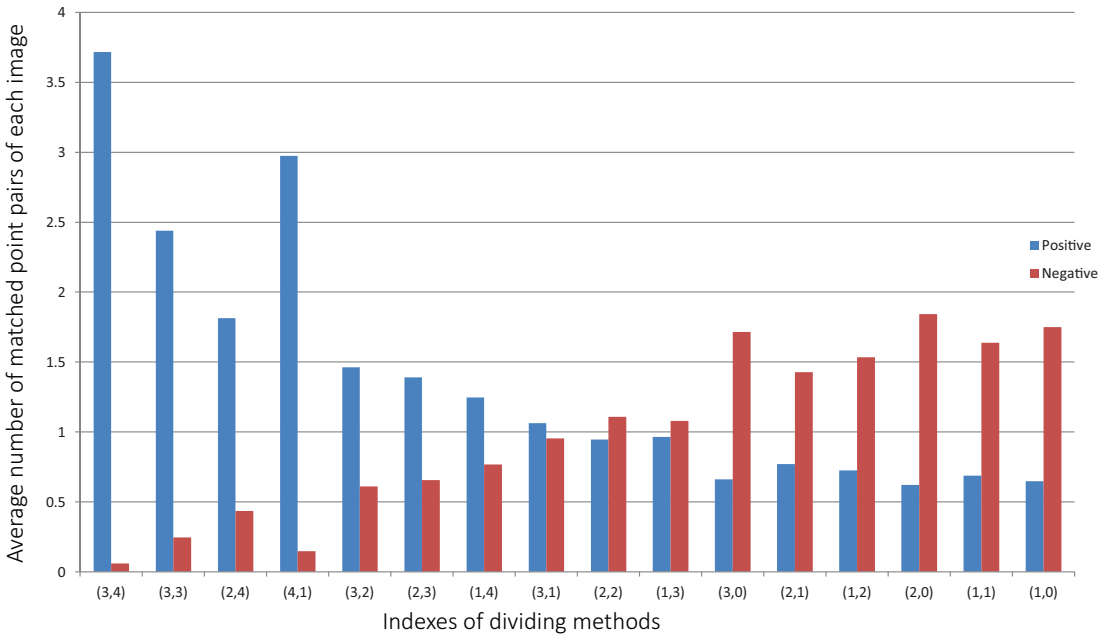


Figure 5.11: Comparison of average matched numbers between positive and negative test examples under different visual-spatial space dividing parameters.

In Figure 5.11, the average numbers of matched pairs of positive and negative test examples are shown. It is clear that under finer dividing methods, positive test examples possess larger average numbers of matched pairs, while under coarser dividing methods, negative test examples possess larger matched numbers of matched pairs.

5.4.2 P-campus

The method is also tested on a dataset of pedestrians [Wang et al., 2010]. For a fair comparison, both training images and test images are exactly same for method comparisons. In Figure 5.12, there are some detection results on the dataset.

In Figure 5.13, there are the results of comparing the proposed method with ordinary Hough transform and common fate Hough transform [Wang et al., 2010]. It can

be seen that, common fate Hough transform performs the best since the method employs motion information. The proposed method almost perform as well as the Hough transform on this dataset.

The method detects at a frame rate of 8 frames per second with multi-thread accelerations, while common fate Hough transform needs two minutes to deal with one frame in its old implementation. For a very fair comparison, multi-thread accelerations are disabled for this method, and both [Barinova et al., 2010] and [Wang et al., 2010] are re-implemented and re-compiled under the same settings. Then all three methods run on the same computer, and the time consumptions are reported in Table 5.1. Obviously, the proposed method consumes the shortest time. There are more than 5,800 codes in the codebook, and the proposed method is more than 200 times faster than a method based on Hough transform to deal with one candidate sub-image in theory. When dealing with all candidate sub-images in one frame, it is only about 5 times faster. This is because that calculations for one sub-image can be cached and can be used for the decisions on its neighboring sub-images. For example, if two sub-images share the same image feature on the frame, and if best matched codes are found for this image feature, these codes can be used for decisions of both the sub-images.

Table 5.1: Time consumptions of the proposed method, [Barinova et al., 2010], and [Wang et al., 2010]. Total time means the running time on all the 79 testing frames, and average time means average running time per frame.

	Total time (ms)	Average time (ms)
Proposed method	363,678	4,603
Hough transform	1,986,583	25,146
Common fate Hough transform	8,296,161	105,014

The training time of the three methods are also compared. Both Hough transform and common fate Hough transform spend 483 ms for training, while pyramid match score spend 978 ms for training. All methods spend less than one second for training.

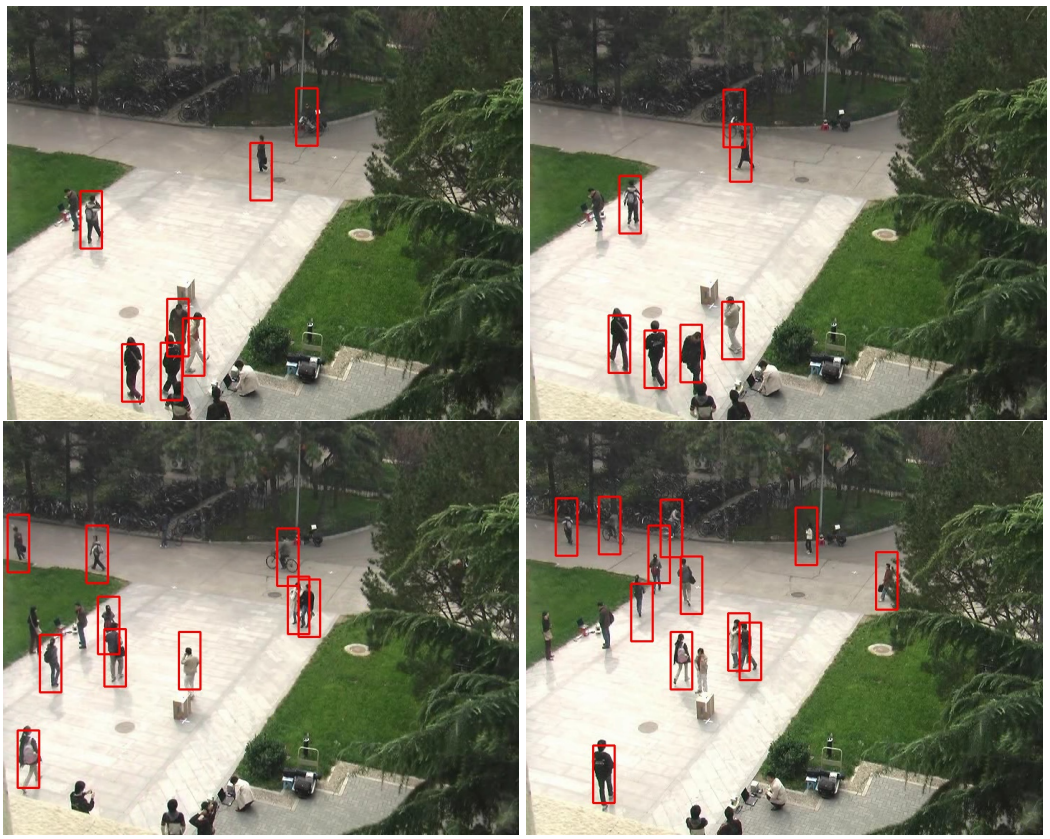


Figure 5.12: Detection results on P-campus dataset.

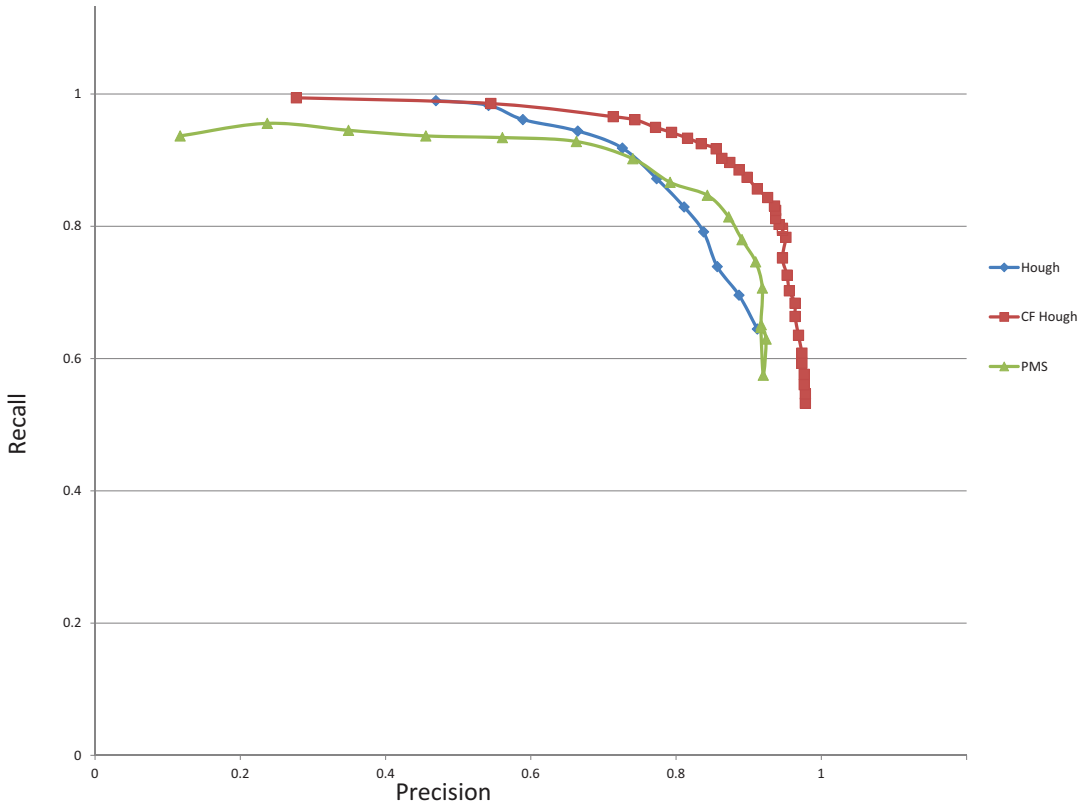


Figure 5.13: Precision-Recall curves for the method, Hough transform, and common fate Hough transform.

5.5 Discussion

5.5.1 Time Complexity

The time complexity of find a sub-optimal best matching between two point sets is $O(dml_{max}^2)$ when pyramid matching is employed. Here, d is the dimension of each visual-spatial point, m is the size of the point set of each object hypothesis, and l_{max} is the largest fineness level, which is usually 5. The match complexity of [Grauman & Darrell, 2005] is $O(dmL), L = l_{max}$. However, this is just the time complexity of matching between two point sets. To give the confidence of a hypothesis, the time complexity of the proposed method is still $O(dml_{max}^2)$, while for [Grauman & Darrell, 2005], it is $O(dml_{max}n_{sup})$, where n_{sup} is the number of support vectors in the used SVM, which is usually larger than 5. So, considering about the time complexity to give confidence for a hypothesis, the proposed method is no worse than [Grauman & Darrell, 2005]. Also n_{sup} is related to the training dataset, while l_{max} is less related. The case of [Lazebnik et al., 2006] is almost the same with [Grauman & Darrell, 2005].

In theory, the time complexity of methods based on Hough transform based methods should be $O(dmn_{tr}^{\frac{1}{1+\varepsilon}})$, if the kNN framework is employed as in Chapter 4. Here n_{tr} is the size of the codebook, which in the case of UIUC cars is about $50 \times 500 = 25,000$. And, $\varepsilon > 0$ is a factor which affects the quality of the kNN. And compared to PMS, this is 1,000 times time-consuming. However, methods based on Hough transform usually does not consider about object detection in the sliding window manner, which makes such methods unable to well deal with scale changes.

The time complexity of DPM should be an exponential function to the number of parts used. Tricks can be used to avoid such heavy calculation in exchange of performance.

Methods employing bag-of-features schema, and using linear SVM have the lowest time complexity, which is usually $O(dmn_{sup})$, while such methods are currently no longer the dominant.

In summary, the proposed method is competitive in the aspect of time complexity, when considering about giving a reliable confidence to an object hypothesis.

5.5.2 Usage of Visual and Spatial Information

Since there are two main information channels of the local features, the methods which make full use of such information should be robust and effective.

Let's take Figure 5.3 as reference. Bag-of-features schema consider similarity between objects in $g(2,0)$. Actually, [Grauman & Darrell, 2005] considers matched image features in $\{g(2,0), g(1,0), g(0,0)\}$, while [Lazebnik et al., 2006] considers in a single fineness level of visual information, i.e., $\{g(1,2), g(1,1), g(1,0)\}$. Generally speaking, method based on Hough transform only consider about the features in larger fineness level of both visual and spatial information, i.e., $\{g(2,2), g(2,1), g(1,2)\}$. And methods based on Hough transform, when in a bottom-up manner, can not deploy the object parts in a way similar to DPM. For example, an object part will contribute to an object it does not belong to.

Together with DPM, PMS considers about visual and spatial information at all fineness levels, and is able to model the correlation between visual and spatial information. However, the way is different. DPM picks several object part at a certain fineness level of visual information, and makes assumptions of their corresponding fineness level of spatial information. For example, DPM will use the most top-left grid of $g(2,1)$, and use the most top-right grid of $g(1,2)$. This grid-picking procedure is achieved by structure SVMs. In the case of PMS, the philosophy is different. Instead of finding some very representative features at certain locations, all features are used, the noisy features are averaged out, since a larger number, i.e., 50, of features are used, compared to DPM, which usually performs good with 6 local features.

Another advantage of PMS is that the number of parameters which need estimating is very small. While, $d = 10$ is an easy conclusion. Other parameters, l_{max}, a, b, and c are all super parameters, which actually contain abundant information. What is attractive is that even using (5.4) to directly assign weights, the performance on UIUC cars is still promising. These results are consistent with [Grauman & Darrell, 2005].

To summarize, the way how the proposed method combines visual and spatial information of local features is theoretically promising.

5.6 Chapter Conclusion

This chapter proposes a method to efficiently and effectively combine visual and spatial information of the local image features. Experiments show the method is comparable with state-of-the-art detection methods on benchmark datasets. What make the method different are: 1) pyramid matching between a codebook and an object hypothesis, 2) the set of methods to dividing the visual-spatial space, and 3) the way to define or learn weights for corresponding dividing method.

The underlying principle of PMS, which is defined by the order of dividing methods in Algorithm 5.3, is that use the template to explain every visual-spatial point of an object hypothesis at the best visual-spatial level.

To the best of the author's knowledge, this is the first attempt of pyramid matching between a codebook and an object hypothesis.

Chapter 6

Conclusion and Outlook

In Chapter 3, a method to efficiently combine motion and appearance information is proposed. The method gives promising results under simple scenarios. To improve the results of Chapter 3 on complex scenarios, in Chapter 4, the Hough transform framework is extended to incorporate motion information, and performs well on two datasets. Still the Hough transform framework itself is troublesome when considering about efficiency. This lead to the method of Chapter 5, which is a new voting system, and is very different in the using of visual and spatial information. The method's effectiveness is proven by experiments, and also it is theoretically promising.

Visual object detection by computers is still and, in near future, continues to be a very open problem. A very hopeful effort would be combining motion, appearance and location information of local features in a robust voting system.

This thesis focuses on improving voting-based detection methods's detection performance by fusion of information of different channels. And in practice, the efficiency of voting is an obstacle, which lead to a new efficient voting system. The method of Chapter 3 uses voting to summarise local visual and motion patterns. The simple appearance model and the linear assumption for motion make it only suitable for particular application cases. The method performs well in detection thermal features in tunnels. The method of Chapter 4 extends the implicit shape model with motion. This method does not have assumptions for motion model, instead, online motion information are used for clustering local visual patterns, which results in better voting results. Though performing promisingly, the method's efficiency is prevented by the time consuming property of voting systems. In Chapter 5, a new voting system is proposed, by

proposing new visual-spatial space dividing method for pyramid matching, the speed of voting is highly enhanced.

Thus detection performance is improved by combining motion information with appearance information in voting, and voting efficiency is improved by proposing new voting mechanisms in the thesis.

In future, the method proposed in Chapter 5 will be extended to incorporate motion information similarly to the method of Chapter 4. Hopefully, this effort will result in a method which is both efficient and effective in using motion information.

The methods of Chapter 3 and Chapter 4 improve detection accuracy by combining motion information. And the method in Chapter 5 is very efficient in both training and detection. Nowadays, there are large amounts of video data, while it is still difficult for computers to understand the meanings of these videos or to make use of the videos. A few projects have started towards this end [Chen et al., 2013; Le et al., 2011]. Still these methods are limited on static images, and are not efficient in training. In future, the proposed methods might be good for such projects.

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. 94, 95

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

Andriluka, M., Roth, S., & Schiele, B. (2008). People-tracking-by-detection and people-detection-by-tracking. In *CVPR*, (pp. 1–8). 52

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

Baidu (2013). 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. 48

Barinova, O., Lempitsky, V., & Kohli, P. (2010). On detection of multiple object instances using hough transforms. In *CVPR*, (pp. 2233–2240). 1, 11, 14, 17, 49, 51, 61, 62, 64, 103

Bay, H., Tuytelaars, T., & Van Gool, L. (2006). Surf: Speeded up robust features. In *ECCV*, (pp. 404–417). 9, 14, 23

Bay, H., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (surf). *CVIU*, *110*(3), 346–359. 42

BIBLIOGRAPHY

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *PAMI*, *24*(4), 509–522. 9

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

Bouwmans, T., El Baf, F., & Vachon, B. (2009). Statistical background modeling for foreground detection: A survey. In *HPRCV*, (pp. IV: 181–199). 52

Brostow, G. & Cipolla, R. (2006). Unsupervised bayesian detection of independent motion in crowds. In *CVPR*, (pp. I: 594–601). 18, 52, 53

Brostow, G., Shotton, J., Fauqueur, J., & Cipolla, R. (2008). Segmentation and recognition using structure from motion point clouds. In *ECCV*, (pp. I: 44–57). 52

Brox, T. & Malik, J. (2010). Object segmentation by long term analysis of point trajectories. In *ECCV*, (pp. V: 282–295). 18, 53

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

Chen, M.-y., Mummert, L., Pillai, P., Hauptmann, A., & Sukthankar, R. (2010). Controlling your tv with gestures. In *Multimedia information retrieval*, (pp. 405–408). 52

Chen, X., Shrivastava, A., & Gupta, A. (2013). Neil: Extracting visual knowledge from web data. 110

Cutler, R. & Davis, L. (1999). Robust real-time periodic motion detection, analysis, and applications. *PAMI*, *22*, 781–796. 52

Dalal, N. & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR*, (pp. 886–893). 1, 9, 11, 14, 17, 81

Dalal, N., Triggs, B., & Schmid, C. (2006). Human detection using oriented histograms of flow and appearance. In *ECCV*, (pp. 428–441). 52

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*. 21

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, 14, 82

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). 52

Felzenszwalb, P. & Huttenlocher, D. (2005). Pictorial structures for object recognition. *IJCV*, *61*(1), 55–79. 1, 11, 17, 48

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *PAMI*, *32*(9), 1627–1645. 82

Felzenszwalb, P. F., Girshick, R. B., & McAllester, D. A. (2010). Cascade object detection with deformable part models. In *CVPR*, (pp. 2241–2248). 82

Felzenszwalb, P. F., McAllester, D. A., & Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *CVPR*, (pp. 1–8). vi, 1, 11, 17, 82, 94, 101

Fergus, R., Perona, P., & Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In *CVPR*, (pp. II: 264–271). 1, 11, 14, 17, 48, 81

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, 14, 17, 79, 82

Gall, J. & Lempitsky, V. (2009). Class-specific hough forests for object detection. In *CVPR*, (pp. 1022–1029). 17, 48, 51

Gavrila, D. M. & Munder, S. (2007). Multi-cue pedestrian detection and tracking from a moving vehicle. *IJCV*, *73*(1), 41–59. 52

Gionis, A., Indyk, P., & Motwani, R. (1999). Similarity search in high dimensions via hashing. In *VLDB*, (pp. 518–529). 82

Gould, S., Fulton, R., & Koller, D. (2009). Decomposing a scene into geometric and semantically consistent regions. In *ICCV*, (pp. 1–8). 53

Grabner, H., Leistner, C., & Bischof, H. (2008). Semi-supervised on-line boosting for robust tracking. In *ECCV*, (pp. 234–247). 52

Grauman, K. & Darrell, T. (2005). The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, (pp. 1458–1465). vi, 14, 82, 83, 85, 86, 87, 88, 91, 97, 99, 106, 107

Gu, C., Lim, J., Arbelaez, P., & Malik, J. (2009). Recognition using regions. In *CVPR*, (pp. 1030–1037). 48

Harris, C. & Stephens, M. (1988). A combined corner and edge detector. In *Alvey Vision Conference*, (pp. 147–152). 59

Huang, C., Wu, B., & Nevatia, R. (2008). Robust object tracking by hierarchical association of detection responses. In *ECCV*, (pp. II: 788–801). 18

Isukapalli, R. & Greiner, R. (2003). Use of off-line dynamic programming for efficient image interpretation. In *IJCAI*, (pp. 1319–1325). 10, 53

Javed, O. (2005). Online detection and classification of moving objects using progressively improving detectors. In *CVPR*, (pp. 696–701). 52

Jégou, H., Douze, M., & Schmid, C. (2010). Improving bag-of-features for large scale image search. *IJCV*, *87*(3), 316–336. 2, 9, 11, 14, 79, 81

Jiang, H. & Yu, S. (2009). Linear solution to scale and rotation invariant object matching. In *CVPR*, (pp. 2474–2481). 12, 81

Jones, M., Viola, P., Viola, P., Jones, M. J., Snow, D., & Snow, D. (2003). Detecting pedestrians using patterns of motion and appearance. In *ICCV*, (pp. 734–741). 2, 52

Kise, K. & Kashiwagi, T. (2011). 1.5 million subspaces of a local feature space for 3d object recognition. In *ACPR*, (pp. 672–676). 7, 12

Kläser, A., Marszalek, M., & Schmid, C. (2008). A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, (pp. 1–10). 53

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 83–97. 30, 83

Lampert, C., Blaschko, M., & Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*, (pp. 1–8). 1, 11, 14, 17, 48, 49

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, 82

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, (pp. 2169–2178). 79, 81, 106, 107

Le, Q. V., Monga, R., Devin, M., Corrado, G., Chen, K., Ranzato, M., Dean, J., & Ng, A. Y. (2011). Building high-level features using large scale unsupervised learning. *CoRR*, *abs/1112.6209*. 110

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*. 7, 12, 14

Lehmann, A., Leibe, B., & Van Gool, L. (2011). Fast prism: Branch and bound hough transform for object class detection. *IJCV*, *94*(2), 175–197. 11, 14, 17, 54, 79

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, 51

Leibe, B., Leonardis, A., & Schiele, B. (2008). Robust object detection with interleaved categorization and segmentation. *IJCV*, *77*(1-3), 259–289. 1, 3, 11, 14, 48, 51, 91

Leibe, B. & Schiele, B. (2003). Interleaved object categorization and segmentation. In *BMVC*, (pp. 759–768). 1, 11, 17, 51

Leibe, B., Schindler, K., & Van Gool, L. (2007). Coupled detection and trajectory estimation for multi-object tracking. In *ICCV*, (pp. 1–8). 17, 18, 53

Levin, A. (2003). Unsupervised improvement of visual detectors using co-training. In *ICCV*, (pp. 626–633). 52

Li, F.-f. (2005). A bayesian hierarchical model for learning natural scene categories. In *CVPR*, (pp. 524–531). 14, 79

Liu, C., Yuen, J., & Torralba, A. (2011). Nonparametric scene parsing via label transfer. *PAMI*, *33*(12), 2368–2382. 14, 81

Liu, Y., Wang, X., & Zha, H. (2008). Dimension amnesic pyramid match kernel. In *AAAI*, (pp. 652–658). 97

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). 82

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *IJCV*, *60*, 91–110. 9, 14, 23, 34, 52

Lucas, B. D. & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *IJCAI*, (pp. 674–679). 53

Maji, S., Berg, A. C., & Malik, J. (2008). Classification using intersection kernel support vector machines is efficient. In *CVPR*, (pp. 1–8). 1, 11, 17

Maji, S. & Malik, J. (2009). Object detection using a max-margin hough transform. In *CVPR*, (pp. 1038–1045). 1, 11, 17, 51

Matas, J., Chum, O., Martin, U., & Pajdla, T. (2002). Robust wide baseline stereo from maximally stable extremal regions. In *BMVC*, (pp. 384–393). 9

Meeds, E., Ross, D., Zemel, R., & Roweis, S. (2008). Learning stick-figure models using nonparametric bayesian priors over trees. In *CVPR*, (pp. 1–8). 9

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, 17, 51

Mikolajczyk, K. & Schmid, C. (2004). Scale & affine invariant interest point detectors. *IJCV*, *60*(1), 63–86. 9

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. 1

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). 53

Nair, V. & Clark, J. J. (2004). An unsupervised, online learning framework for moving object detection. In *CVPR*, (pp. 317–325). 52

Noguchi, A. & Yanai, K. (2009). Extracting spatio-temporal local features considering consecutiveness of motions. In *ACCV*, (pp. 458–467). 14, 52

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, 17, 48, 82

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. 9

Okada, R. (2009). Discriminative generalized hough transform for object dectection. In *ICCV*, (pp. 2000–2005). 17, 48, 51

Olshausen, B. & Field, D. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*, 607–609. 82

Ommer, B. & Malik, J. (2009). Multi-scale object detection by clustering lines. In *ICCV*, (pp. 484–491). 51

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*. 22

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

PASCAL (2012). Visual Object Classes Challenge 2012. http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2012. [Online; accessed 22-May-2013]. 8

Pedersoli, M., Vedaldi, A., & Gonzàlez, J. (2011). A coarse-to-fine approach for fast deformable object detection. In *CVPR*, (pp. 1353–1360). 82

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). 9

Pirsiavash, H. & Ramanan, D. (2012). Steerable part models. In *CVPR*, (pp. 3226–3233). 10, 82

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). 17

Schneiderman, H. & Kanade, T. (2004). Object detection using the statistics of parts. *IJCV*, *56*(3), 151–177. 1, 11, 17

Shawe-Taylor, J. & Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press. 81, 82

Shechtman, E. & Irani, M. (2007). Matching local self-similarities across images and videos. In *CVPR*, (pp. 1–8). 9

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. 54

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). 82

Stauffer, C. & Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. In *CVPR*, (pp. 2246–2252). 25

Szegedy, C., Toshev, A., & Erhan, D. (2013). Deep neural networks for object detection. In *NIPS*, (pp. 2553–2561). 14

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, 14

Tomasi, C. & Kanade, T. (1991). Detection and tracking of point features. Technical report, IJCV, Carnegie Mellon University. 9, 14, 53, 59

Tuzel, O., Porikli, F., & Meer, P. (2006). Region covariance: A fast descriptor for detection and classification. In *ECCV*, (pp. II: 589–600). 9, 53, 59

Viola, P. A. & Jones, M. J. (2004). Robust real-time face detection. *IJCV*, *57*(2), 137–154. 1, 11, 17

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). 42, 102, 103

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. 49

Wikipedia (2008). Object Detection. `http://en.wikipedia.org/wiki/Object_detection`. [Online; accessed 22-May-2013]. 1

Wikipedia (2012). Siri. `http://en.wikipedia.org/wiki/Siri_(software)`. [Online; accessed 22-May-2013]. 7

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). 18, 23, 53

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). 12, 53

Yang, M., Yu, T., & Wu, Y. (2007). Game-theoretic multiple target tracking. In *ICCV*, (pp. 1–8). 17

Yang, Y., Shu, G., & Shah, M. (2013). Semi-supervised learning of feature hierarchies for object detection in a video. In *CVPR*. 52

Yarlagadda, P., Monroy, A., & Ommer, B. (2010). Voting by grouping dependent parts. In *ECCV*, (pp. V: 197–210). 17, 18, 51, 52

Yeh, T., Lee, J., & Darrell, T. (2009). Fast concurrent object localization and recognition. In *CVPR*, (pp. 280–287). 1, 10, 11, 14, 17, 48

Zhang, L., Li, Y., & Nevatia, R. (2008). Global data association for multi-object tracking using network flows. In *CVPR*, (pp. 1–8). 17, 53

Zhu, L. L., Chen, Y., Yuille, A., & Freeman, W. (2010). Latent hierarchical structural learning for object detection. In *CVPR*, (pp. 1062–1069). 82

Acknowledgement

First and foremost, I would like to thank my advisor, Professor Katsushi Ikeuchi. It is a great honor for me to be a Ph.D. student of Professor Ikeuchi. He leads me in my research. He shows to me how a top researcher works. He helps and comforts me during my most tough times. He gives me the chance to enjoy my research and my life in Japan. For me, Professor Ikeuchi is more than an advisor in research, and also an advisor in life. The tree years I spend in Ikeuchi lab is a lifelong treasure.

I would like to thank Dr Masataka Kagesawa, and Associate Professor Shintaro Ono for co-advising me. They are patient with my questions, they lead me and help me in my research, they help me in my personal affairs. They help me in a way I cannot appreciate more. While being co-advisors, they are also friends. I would also like to thank Dr Atsuhiko Banno, who gives me suggestions and helps me in my research.

I would like to thank Associate Professor Takeshi Oishi, Dr Bo Zheng, and Dr Rei Kawakami of computer vision lab. Thank them for their help in research and life during my Ph.D. period. I would like to thank Mr Yasuhide Okamoto, Ms Keiko Motoki, Ms Yoshiko Matsuura and Ms Yuko Nishine for their help.

I want to thank the students in Ikeuchi lab for their help, and discussions in research.

I thank the staff belonging to the department of Computer Science, and the staff in Graduate School of Information Science and Technology. I thank them for all the help.

I also want to show my appreciations to the defense committee. Thank Professor Takeo Igarashi. Thank Associate Professor Shigeo Takahashi. Thank Professor Reiji Suda. Thank Professor Kiyohara Aizawa. And thank Professor Tatsuya Harada.

I thank all the people who help me with my study and life in Tokyo.

I want to thank Professor Hongbing Zha, who is my master advisor and encouraged me to continue with research in computer vision. Also the idea in Chapter 4 has its

root in some work when I was under Professor Zha' supervision.

I thank my grandmother Mrs. Zhong, and my father Sishun Wang for bringing me up. They should have been very happy to see my doctor graduation.

I thank my wife MIN/Li for quitting her job to follow me to Tokyo. I thank her for taking care of me and our daughter Hetong Wang, which makes my focusing on study and research feasible.