Vision-based moving pedestrian recognition from imprecise and uncertain data

Dingfu Zhou

▶ **To cite this version:**

Dingfu Zhou. Vision-based moving pedestrian recognition from imprecise and uncertain data. Other. Université de Technologie de Compiègne, 2014. English. NNT : 2014COMP2162 . tel-01332947

HAL Id: tel-01332947

https://theses.hal.science/tel-01332947

Submitted on 16 Jun 2016

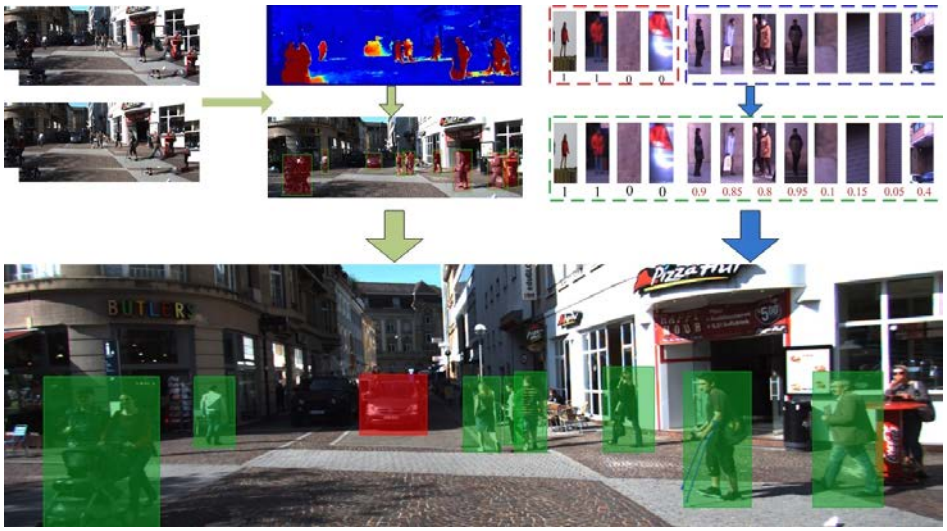
HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Par **Dingfu ZHOU**

Vision-based moving pedestrian recognition from imprecise and uncertain data

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



Soutenue le 05 décembre 2014
Spécialité : Information and Systems Technologies

D2162

University of Technology of Compiègne

Thesis

Submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Area of specialization: Information and Systems Technologies

by

Dingfu ZHOU

Vision-Based Moving Pedestrian Recognition from Imprecise and Uncertain Data

Heudiasyc Laboratory, UMR UTC/CNRS 7253

Defended on December 5th, 2014.
Thesis Committee:

Reviewers:	Yassine Ruichek	Pr UTBM	SET
	Arnaud Martin	Pr University of Rennes	IRISA
Examiners:	Philippe Bonnifait	Pr UTC	HEUDIASYC
	Marie Szafranski	MCF Université d'Évry	IBISC
Supervisors:	Vincent Frémont	MCF UTC	HEUDIASYC
	Benjamin Quost	MCF UTC	HEUDIASYC

Université de Technologie de Compiègne

Thèse

pour obtenir le grade de

Docteur

Spécialité: Technologie de l'information et des systèmes

par

Dingfu ZHOU

Reconnaissance de piétons par vision à partir de données imprécises et incertaines

Laboratoire Heudiasyc, Unité Mixte de Recherche UTC/CNRS 7253

Soutenue le 5 Décembre, 2014 devant le jury constitué de :

Rapporteurs:	Yassine Ruichek	Pr UTBM	SET
	Arnaud Martin	Pr University of Rennes	IRISA
Examinateurs:	Philippe Bonnifait	Pr UTC	HEUDIASYC
	Marie Szafranski	MCF Université d'Évry	IBISC
Directeurs de thèse:	Vincent Frémont	MCF UTC	HEUDIASYC
	Benjamin Quost	MCF UTC	HEUDIASYC

Acknowledgments

This PhD thesis was carried out under supervision of Dr. Vincent Frémont and Dr. Benjamin Quost at the Heudiasyc Laboratory in the University of Technology of Compiègne (UTC) from October 2011 to November 2014. It was financed with a scholarship by the China Scholarship Council (CSC) of Chinese government. All this work could not be accomplished without the aid of many people, in their different ways. Here, I would like to express my appreciation to the following.

Foremost, I would like to express my sincere gratitude to my two advisors for their continuous support to my PhD study and research. Thanks for their valuable advices, patience, motivation, enthusiasm, and selfless. Their guidance helped me through all the time of my research and writing papers. In particular, I want to thank them for giving me detailed reviews, comments and corrections of my dissertation during the writing this thesis.

I would like to thank those who served as my jury members: Yassine Ruichek, Arnaud Martin, Philippe Bonnifait and Marie Szafranski. My sincere appreciation will be given for their time and energy.

My sincere thanks also goes to Prof. Mingyi He, my Masters thesis advisor, who gives me much help for applying the PhD position in UTC and suggestions during my research. At the same time, I also thanks to Dr. Yuchao Dai (Research School of Computer Science, Australian National University) for the stimulating discussions with him.

I would like to thank all staff and colleagues of Heudiasyc, whose friendships and encourage made me have a memorable experience at UTC. Also I thank my friends in UTC: Xiao Liu, Zui Tao, Liqi SUI, Yuliang Hou, Chunlei Yu, Bihao Wang, Philippe Xu etc. My life in France would not be so rich and colorful without them.

Last but not the least, I would like to thank my parents for giving birth to me and supporting me selflessly throughout my life and thanks to Nan Li, for her love, support and patience.

Abstract

Vision-based Advanced Driver Assistance Systems (ADAS) is a complex and challenging task in real world traffic scenarios. The ADAS aims at perceiving and understanding the surrounding environment of the ego-vehicle and providing necessary assistance for the drivers if facing some emergencies. In this thesis, we will only focus on detecting and recognizing moving objects because they are more dangerous than static ones. Detecting these objects, estimating their positions and recognizing their categories are significantly important for ADAS and autonomous navigation. Consequently, we propose to build a complete system for moving objects detection and recognition based on vision sensors.

The proposed approach can detect any kinds of moving objects based on two adjacent frames only. The core idea is to detect the moving pixels by using the Residual Image Motion Flow (RIMF). The RIMF is defined as the residual image changes caused by moving objects with compensated camera motion. In order to robustly detect all kinds of motion and remove false positive detections, uncertainties in the ego-motion estimation and disparity computation should also be considered. The main steps of our general algorithm are the following: first, the relative camera pose is estimated by minimizing the sum of the reprojection errors of matched features and its covariance matrix is also calculated by using a first-order errors propagation strategy. Next, a motion likelihood for each pixel is obtained by propagating the uncertainties of the ego-motion and disparity to the RIMF. Finally, the motion likelihood and the depth gradient are used in a graph-cut-based approach to obtain the moving objects segmentation. At the same time, the bounding boxes of moving object are generated based on the U-disparity map.

After obtaining the bounding boxes of the moving object, we want to classify the moving objects as a pedestrian or not. Compared to supervised classification algorithms (such as boosting and SVM) which require a large amount of labeled training instances, our proposed semi-supervised boosting algorithm is trained with only a few labeled instances and many unlabeled instances. Firstly labeled instances

are used to estimate the probabilistic class labels of the unlabeled instances using Gaussian Mixture Models after a dimension reduction step performed via Principal Component Analysis. Then, we apply a boosting strategy on decision stumps trained using the calculated soft labeled instances. The performances of the proposed method are evaluated on several state-of-the-art classification datasets, as well as on a pedestrian detection and recognition problem.

Finally, both our moving objects detection and recognition algorithms are tested on the public images dataset KITTI and the experimental results show that the proposed methods can achieve good performances in different urban scenarios.

Résumé

La mise en oeuvre de systèmes avancés d'aide à la conduite (ADAS) basés vision, est une tâche complexe et difficile surtout d'un point de vue robustesse en conditions d'utilisation réelles. Une des fonctionnalités des ADAS vise à percevoir et à comprendre l'environnement de l'ego-véhicule et à fournir l'assistance nécessaire au conducteur pour réagir à des situations d'urgence. Dans cette thèse, nous nous concentrons sur la détection et la reconnaissance des objets mobiles car leur dynamique les rend plus imprévisibles et donc plus dangeureux. La détection de ces objets, l'estimation de leurs positions et la reconnaissance de leurs catégories sont importants pour les ADAS et la navigation autonome. Par conséquent, nous proposons de construire un système complet pour la détection des objets en mouvement et la reconnaissance basées uniquement sur les capteurs de vision.

L'approche proposée permet de détecter tout type d'objets en mouvement en fonction de deux méthodes complémentaires. L'idée de base est de détecter les objets mobiles par stéréovision en utilisant l'image résiduelle du mouvement apparent (RIMF). La RIMF est définie comme l'image du mouvement apparent causé par le déplacement des objets mobiles lorsque le mouvement de la caméra a été compensé. Afin de détecter tous les mouvements de manière robuste et de supprimer les faux positifs, les incertitudes liées à l'estimation de l'ego-mouvement et au calcul de la disparité doivent être considérées. Les étapes principales de l'algorithme sont les suivantes: premièrement, la pose relative de la caméra est estimée en minimisant la somme des erreurs de reprojection des points d'intérêt appariées et la matrice de covariance est alors calculée en utilisant une stratégie de propagation d'erreurs de premier ordre. Ensuite, une vraisemblance de mouvement est calculée pour chaque pixel en propageant les incertitudes sur l'ego-mouvement et la disparité par rapport à la RIMF. Enfin, la probabilité de mouvement et le gradient de profondeur sont utilisés pour minimiser une fonctionnelle d'énergie de manière à obtenir la segmentation des objets en mouvement. Dans le même temps, les boîtes englobantes des objets mobiles sont générées en utilisant la carte des U-disparités.

Après avoir obtenu la boite englobante de l'objet en mouvement, nous cherchons à reconnaître si l'objet en mouvement est un piéton ou pas. Par rapport aux algorithmes de classification supervisée (comme le boosting et les SVM) qui nécessitent un grand nombre d'exemples d'apprentissage étiquetés, notre algorithme de boosting semi-supervisé est entraîné avec seulement quelques exemples étiquetés et de nombreuses instances non étiquetées. Les exemples étiquetés sont d'abord utilisés pour estimer les probabilités d'appartenance aux classes des exemples non étiquetés, et ce à l'aide de modèles de mélange de gaussiennes après une étape de réduction de dimension réalisée par une analyse en composantes principales. Ensuite, nous appliquons une stratégie de boosting sur des arbres de décision entraînés à l'aide des instances étiquetées de manière probabiliste. Les performances de la méthode proposée sont évaluées sur plusieurs jeux de données de classification de référence, ainsi que sur la détection et la reconnaissance des piétons.

Enfin, l'algorithme de détection et de reconnaissances des objets en mouvement est testé sur les images du jeux de données KITTI et les résultats expérimentaux montrent que les méthodes proposées obtiennent de bonnes performances dans différents scénarios de conduite en milieu urbain.

Contents

List of Symbols

Moving Object Detection

$\mathbf{X}_W = (X, Y, Z, 1)^T$	A homogenous 3D world point in world coordinate system
$\mathbf{X}_C = (X, Y, Z, 1)^T$	A homogenous 3D world point in camera coordinate system
$\mathbf{x} = (u, v, 1)^T$	A homogenous 2D image point in 2D image coordinate
$\mathbf{R}$ and $\mathbf{t}$	Camera relative pose expressed by rotation and translation matrix
$\Theta = (r_x, r_y, r_z, t_x, t_y, t_z)^T$	Camera relative pose expressed by roll, pitch, yaw angles and translation vector.
$\mathbf{K}$	Camera intrinsic matrix
f	The focal length of the camera (in pixels)
b	Base line of the stereo vision system (in meters)
$\mathbf{P}(X, Y, Z)$	3D point
$\mathbf{p}$	Corresponding image point projected from same 3D point $\mathbf{P}$
$\mathbf{I}$	Image frame
$(u_{t,l}, v_{t,l})^T$	2D point location in image coordinate
$(\hat{u}_{t,l}, \hat{v}_{t,l})^T$	Predicted 2D point location in image coordinate
$\hat{\mathbf{x}}$ and $\mathbf{x}$	Predicted and detected image points vector
$\mu_{\mathbf{x}}$ and $\Sigma_{\mathbf{x}}$	Mean and covariance matrix of variable $\mathbf{x}$
$\mathbf{J}$	Jacobian matrix w.r.t each variable
$\mathbf{f}(.)$	Represents the non-linear image point projection function from previous frame to current frame
d	Disparity
$U_d(u, v)$	Uncertainty of the disparity value at pixel (u, v)
μ	Mahalanobis distance associated to the residual image motion flow

ξ	Motion likelihood computed by residual image motion flow and its covariance matrix
h_i and h_c	The height of object i and camera in world coordiate system
θ	The camera tilt angle

Pedestrian Recognition

$F(x)$	Final decision of boosting classifier
$f_t(x)$	Weak learner (or weak classifier) t
α_t	Weight (confidence) of t_{th} weak learner
ϵ_t	Classification error of weaklearner t
D^L, $D^U$and D	Labeled, unlabeled and all training data
n, m and N	Number of labeled, unlabeled and all training samples
$\Psi = \{\theta_k, \mu_k, \Sigma_k\}$	kth component of the GMM model
$\pi_{i,k}$	Probabilistic class label of sample i estimated by GMM for class k
$G(\lambda_t)$	Gini index of tree node λ_t
$p_k(\lambda_t)$	Class probability for current node λ_t of class k
n_t, n_l and n_r	Number of samples in node λ_t, left child node of node λ_t and right child node of node λ_t
n_{kt}	Number of samples in node λ_t from class k
γ	Ratio of unlabeled samples
d	Data dimension of feature vector
m^+ and m^-	Number of positive and negative samples
m^+_{mis} , m^-_{mis}	Number of positive and negative samples that have been misclassified
m_{mis}	Number of total samples that have been misclassified
α and β	Additional border ratio in horizontal and vertical directions of object bounding box
ρ	Ratio threshold for single or grouped object bounding box
s_x and s_y	Sliding stride for grouped bounding box in horizontal and vertical directions

Acronyms

ADAS	Advanced Driver Assistance Systems
CART	Classification And Regression Tree
EM	Expectation Maximization
GIMF	Global Image Motion Flow
GMM	Gaussian Mixture Model
GPS	Global Positioning System
GPU	Graphics Processing Unit
HOG	Histograms of Oriented Gradients
LIDAR	LIght Detection And Ranging
MLE	Maximum Likelihood Estimates
MOF	Measured optical flow
MOD	Moving Object Detection
PCA	Principal Component Analysis
RANSAC	RANdom SAmple Consensus
RIMF	Residual image motion flow
ROI	Region of Interest
SfM	Structure from Motion
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SVM	Support Vector Machine
SVS	Stereo Vision System
ZNCC	Zero mean Normalized Cross-Correlation

List of Figures

Chapter 1

Introduction

Contents

1.1 Background

Over the past decades, many researchers from different research fields such as robotics, automotive engineering and signal processing have been devoting themselves to the development of intelligent vehicle systems. Making the vehicles perceive and understand their surrounding environment automatically is a challenging and important task. Due to the improvement of the sensor technologies, processing techniques and researchers' contributions, several ADASs have been developed for various purposes such as forward collision warning systems, parking assist systems, blind spot detection systems and adaptive cruise control systems. Furthermore, some fully autonomous vehicles have been also developed in last few years (see Fig. (1.1). The driverless car in Fig. (1.1)-(a) belongs to the team of Stanford university, while the Google's driverless cars (Fig. (1.1)-(b)) are at the testing stage now.

As one of the most famous pioneers in the field of intelligent vehicles, this latter has been already tested on different US states roads these years. In August 2012, Google announced that they have completed over 300,000 miles without any

accident. However, the excellent performances are mainly based on some expensive equipments (about $150,000 in total), such as a $70,000 LIDAR (Light Detection And Ranging) system, a Velodyne 64-beam lidar (as in Fig. (1.2)-(a)), etc. Based on the Velodyne lidar, a detailed 3D environment map can be generated. Using this local generated map together with high-resolution world maps, the car can drive itself with the help of other models such as lane, pedestrian and vehicles detection and traffic lights recognition. Although the LIDAR sensors, which have the advantages of high precision and independence of the ambient light conditions, are the most widely used senors in intelligent vehicle systems, they come with some drawbacks as listed below:

1. A high price. A good LIDAR with high resolution may be more expensive than a car.

2. Slow refresh rates. Normally, a LIDAR builds the environment map by scanning a scene, while the map is distorted by the movement of the host vehicle and the motion of surrounding objects if the refresh rates is not high enough.

3. High energy consumption. The electrical power is limited in vehicles, while as an active sensor, a LIDAR is power consuming.

Due to these drawbacks, some researchers move their attentions to other sensors, such as cameras. Compared to other sensing systems, stereo vision based perception systems are closer to the two eyes of human beings. Both of them build the 3D world

(a) *Junior: Driverless car of Stanford University in 2007*

(b) *Google's Driverless Car in 2014*

Figure 1.1: *Examples of autonomous vehicles*

(a) *Velodyne high definition Lidar-64 lasers* **(b)** *Point Grey research's stereo vision camera systems: Bumblebee2*

Figure 1.2: *Examples of Lidar and camera sensors*

by using 2D images from different views. Moreover, cameras have the following properties:

1. Low price. A common webcamera with a moderate resolution only costs about $10. Even a special 3D stereo vision camera costs about $500, which is also much cheaper than a LIDAR system.

2. Color information. Chrominance of color images can be useful for detecting and understanding different objects.

3. Semantic and geometric information. From high resolution and colored images, traffic and brake lights, turn signals and lane lines can be recognized based on different approaches. Lidar systems are expert in telling whether something is there, while vision based systems are able to figure out what it is. In addition, based on the computer vision theory, 3D environment map can also be reconstructed from multi-view images.

4. Low energy consumption. Compared to active vision systems (LIDAR and laser), passive vision systems (e.g., cameras) need less power to work.

Although the camera sensors have various advantages, they also have to face various challenges. Algorithms development and computing power are two determining factors for the vision based systems. The computing problem is being figured out step by step with the developing of multi-core computers, parallel computing techniques

and Graphics Processing Unit (GPU) hardware. According to Moore's law, overcoming the problem of low computation efficiency is just the matter of time. The development of new algorithms for computer vision is the real bottleneck for vision based systems. Although facing these challenges, several specific sub-problems of the whole environment perception systems have been well studied, and promising results have been obtained, such as lane detection [1, 2], road detection [3, 4, 5], detection of traffic sign [6, 7], pedestrians [8, 9] and of other vehicles [10].

1.2 Objectives

Among various challenges in the whole environment perception system, we focus on the specific problem of moving object detection (MOD) and recognition in this dissertation. Generally speaking, there are two main objectives in this thesis: first, detect and segment the moving objects based on stereo images only; second, recognize these moving objects as pedestrian or non-pedestrian based on a recognition step.

1.2.1 Moving Objects Detection

Moving objects are the most common traffic participants and the traffic accidents are frequently caused by their abnormal behaviors. Although the problem of moving objects detection in images has been widely studied in the field of computer vision and various approaches have been proposed, it is still a challenging task when cameras are installed on a mobile platform, such as mobile vehicle, robot, etc. In this case, all the background is moving because of the movement of the camera. In order to distinguish the real moving objects from the background, image changes caused by camera motion should be compensated firstly. Here, the proposed algorithm should be applied in urban environment through a stereo camera rig mounted on the top of vehicle.

For a successful moving objects detection system, several requirements should be satisfied. First of all, the system should be able to detect and segment any types of motion in the images including partially moving objects, small moving objects and partially occluded moving objects. Supervised learning based approaches have been widely and successfully used for object detection. However, they face a big trouble to detect partially or seriously occluded objects. These methods fail because features used to train the classifiers describe an object as a whole in the training process,

while the features of occluded object are different in the detection process due to some parts of the object have been replaced by the foreground occlusions. Part-based models [9] can solve the occlusion problem to some extent, however, they also fail to detect seriously occluded objects.

Second, the system should be able to or has the possibility to be used in real world applications. One essential requirement is that, for the detection in the current frame, the algorithm can only use the images from current and past frames, while the information from the future frames cannot be used. By taking this into account, some methods based on the whole image sequence analysis [11, 12] cannot be used. Computation efficiency is another crucial factor for the real applications. Compared to mufti-frames based approaches [13, 14], detection based on two frames [15, 16] can reduce the processing time.

1.2.2 Pedestrian Recognition

After obtaining the regions of interest (ROI) of the moving objects in the previous detection step, furthermore, identifying the nature of them is also significant for ADAS. With the help of these information, drivers or systems may make the right decisions to reduce the possibility of accident. Additionally, we only recognize the object as a pedestrian or not since they are the most vulnerable road users, but the approach can be easily extended to cyclists, cars, etc.

Classification algorithms, such as boosting and SVM (Support vector machine), are widely and successfully used for pedestrian detection and recognition. However, the performance of a classifier highly depends on training samples. It is easy to understand that the classifiers can achieve high detection rates when the training and testing samples share the similar data distribution (e.g., obtained in similar environments). However, generic classifiers trained using public dataset may exhibit poor detection rates in some specific scenes due to several reasons: the testing and training images are taken from different viewpoints, resolutions, light conditions, etc. In other words, they share different data distributions. In fact, building labeled training data on specific scenes requires a lot of extra labeling efforts and sometimes it is not practical in many scenarios. On the other hand, unlabeled samples are more easy to obtained by different approaches like foreground object detection [16, 12], generic object detectors [17, 18, 19] (which are trained using public dataset), etc.

Therefore, the second objective of this thesis is to train a semi-supervised classifier with few labeled samples and a large amount of unlabeled instances and then to improve the pedestrian recognition rates in some specific scenes.

1.3 Thesis Contributions

In this thesis, an approach to detect and recognize moving objects from mobile stereo vision is presented. The main contributions of our research are as following:

The first main contribution of this research lies in consideration of the uncertainties in ego-motion estimation and disparity map calculation to estimate the pixel motion likelihood. Uncertainties are inevitable in the whole MOD system and they may result in a lot of false positive detections if they are ignored. Based on the assumption of additive Gaussian noise, the covariance matrix of the ego-motion can be computed by propagating the noise from the feature extraction and matching process. Then the covariance matrix of RIMF is calculated for each pixel based on the ego-motion covariance matrix, disparity uncertainty and pixel location noise via a first-order error propagation strategy. The motion likelihood of each pixel can be easily computed once the covariance matrix of the RIMF is known.

The combination of motion likelihood and depth value into a graph cut optimization framework to segment the moving object is another contribution of this thesis. After obtaining the motion likelihood of each pixel, a fixed threshold cannot result in accurate detection performance, for example some false positive detections which are caused by the uncertainties in the dense optical flow estimation process at object boundaries.

In order to obtain a global segmentation of the moving objects, we resort to an energy minimization framework, where the motion likelihood is taken as the data term of the energy function and the depth values are used to build the boundary regularization term. Finally, a global optimal solution is obtained by using the graph-cut minimization algorithm.

The third contribution of this work comes from the soft label based boosting algorithm which is trained by data with both soft and hard class labels. The soft class labels are used to represent the hidden information of the unlabeled samples and they can be estimated through clustering methods or other classifiers. In order to train samples with soft class labels, a novel approach has been proposed under the boosting framework. Decision trees are often used as weak classifier in the boosting algorithm, however, the classical decision trees only take hard labeled samples as inputs. Here, a variant of the decision trees which can take both soft and hard labeled training samples has been applied. At the same time, cost-weighed classification error is used to replace the 0-1 classification error to measure the classification accuracy of an instance. The weak classifier has been iteratively learned by updating the distribution weights of the training samples. The weights of the classifier are

then computed according to the classification accuracy of each weak classifier.

Finally, the proposed MOD system has been tested on various image sequences recorded in different traffic scenes including inner city streets, country roads, and highway scenarios. Experimental results show that the proposed approach can effectively detect different kinds of motion within a certain range (less than 40m), even in some hard cases such as partial occlusion and degenerate motion[1]. At the same time, the soft label based boosting classifier is also applied in each ROI to verify whether there is a pedestrian inside or not. The recognition results show that the proposed classifier gives better performances than a classifier trained using only few labeled samples.

Parts of this thesis have been published in the following international conference papers,

[**1**] Dingfu ZHOU, Benjamin Quost and Vincent Fremont. Soft Label Based Semi-Supervised Boosting for Classification and Object Recognition. In Control, Automation, Robotics and Vision, 2014 Proceedings, IEEE.

[**2**] Dingfu ZHOU, Vincent Fremont, Benjamin Quost and Bihao Wang. On Modeling Ego-Motion Uncertainty for Moving Object Detection from a Mobile Platform, In Intelligent Vehicles Symposium, 2014 Proceedings, IEEE, pages 1332-1338.

[**3**] Dingfu ZHOU, Vincent Fremont and Benjamin Quost. Moving Objects Detection and Credal Boosting Based Recognition in Urban Environments. Cybernetics and Intelligent Systems, IEEE Conference on, 2013, pages 24-29.

1.4 Organization of the Thesis

In this manuscript, we systematically introduce the methods, theoretical foundation, experimental design, results and conclusion of the research. Vision-based moving object detection is presented in Chapter 2, and semi-supervised boosting based pedestrian recognition is described in Chapter 3.

A general introduction and an overview of works related to the MOD problem in ADAS is given at the beginning of Chapter 2. The detailed descriptions of our

[1] The 3D object moves along the epipolar plane formed by the two camera centers and the object itself, whereas its 2D projections move along the epipolar lines.

proposed stereo-vision-based moving object detection are then introduced. Finally, our approach is tested on several image sequences on public dataset and the experimental results and analysis are provided.

In Chapter 3, a general description of pedestrian recognition features, boosting and their application on semi-supervised learning is presented first. Then, the proposed soft-label based semi-supervised boosting algorithm is introduced. Real experiments on public datasets for classification and pedestrian recognition are provided, before a conclusion given at the end of the chapter.

Chapter 4 gives a conclusion of our research work as well as perspectives on future research.

Chapter 2

Vision-Based Moving Object Detection

Contents

2.1 Introduction

In this chapter, we study a state-of-the-art approach for ADAS focusing on the problem of MOD in urban traffic environments. Detecting them from dynamic scenes is a fundamental task for obstacle avoidance and path planning. Being able to detect these moving objects and estimate their positions and dynamic information is crucial for the development of ADAS and autonomous navigation. Traditionally, mobile robotics research in navigation relies on the assumption of static environments, however, this assumption will easily collapse when other moving objects are also involved in the environments. Therefore, robust MOD results can also help to improve the performances of Simultaneous Localization and Mapping (SLAM) [20] and Structure-from-Motion (SfM) [21] which are two basic research fields in the robotics and intelligent vehicle systems to reconstruct the environment and the motion of the vehicle.

Compared to highways, inner city traffic is more difficult, thus the task of ADAS is more challenging and currently is still an unsolved problem. The roads in urban city are crowded with different kinds traffic participants. Fig. (2.1) gives two typical traffic scenes of the urban roads which include various moving objects such as cars, buses, vans, pedestrians and cyclists. At the same time, the surrounding scenario changes arbitrarily with the motion of ego-vehicle. In addition, illegal behaviors are also common in inner city because of the complex traffic environment, such as speeding, running the red light and illegal parking. Due to the reasons mentioned above, the drivers should be cautious and careful in urban city. At the same time, the ADAS are needed to provide help for them in some emergency circumstances.

2.1.1 State of the Art

Robust scene perception and MOD in urban environments attract many resarchers' attention recently and a lot of works have been done by using different sensors. Lidar systems can provide accurate 3D world points that can be used for ego-motion estimation, 3D local grid map updating and moving objects detecting and tracking in urban environment [22]. In [23], the moving objects are detected by fusing the information from the Lidar range scanner and an enhanced map using the Dempster-Shafer theory. In [24, 25], laser together with GPS and camera have been used for vehicle localization and autonomous navigation. First, a sensor selection step that is applied to validate the coherence of the observations from different sensors; then the information provided by the validated sensors is fused with an unscented information

(a)

(b)

Figure 2.1: *Two typical traffic situations in inner city*

filter. While Lidar sensors can directly provide important 3D points for the traffic scene, it will also lose a lot of detailed information of the objects such as texture, appearance and color, etc. This is why camera based systems have become popular in the last few years. The recent works in [26, 27] shows that vision-based approaches give promising results as well. Several vision-based MOD approaches are introduced respectively as below according to the number of cameras they used.

Monocular Camera

Background subtraction is a widely used approach for detecting moving objects in videos from static monocular camera. In this case, regions of interest can easily be detected if the background model can be accurately built [28]. Adaptive Gaussian Mixture Model is well known for background modeling by recursively updating the Gaussian parameters and simultaneously setting the appropriate number of components for each pixel [29]. However, background subtraction is generally based on a static background hypothesis which is often not applicable in real outdoor scenes due to wind, rain or illumination changes brought by weather. In addition, background subtraction can not be applied to handle the problem when the camera also moves.

Detection becomes difficult when the camera and the surrounding objects move simultaneously, because the camera and objects motions become coupled in the apparent motion field. The epipolar constraint, such as the fundamental matrix, is

a commonly used constraint for motion detection between two views [30]. By using the epipolar constraint, a 3D point can be considered as moving if its projected 2D pixel in the first view does not lie on the epipolar line induced by its matched pixel in the second view. However, the epipolar constraint is not able to detect all kinds of 3D motion, for example a special 3D motion which is called degenerate motion[1] can not be detected. Additionally, an accurate estimate of fundamental matrix is impossible when the camera undergoes a nearly pure translation between the two views. Other constraints such as flow vector bound constraint [31, 32] together with epipolar constraint have been used to detect the degenerate motion. If a scene point moves with the degenerate motion, the direction of its optical flow will coincide with epipolar lines. However, the length of its image motion can be predicted if its depth information is known. Hence, the moving points can be distinguished if the real optical flow violates the predicted image motion.

2D planar homography is another common used technique for detecting the moving objects [33, 13, 34, 35]. The homography is used as a global image motion model to compensate the camera motion between two consecutive frames. Pixels which are consistent with the homography matrix are recognized as the static planar background, while these inconsistent ones may belong to moving objects or to static 3D structure with large depth variance (parallax pixels). In order to remove the parallax pixels, additional geometric constraints [33] or mean shift clustering strategy [35] are necessary.

Stereo Camera Rig

Compared to monocular vision, stereo vision system (SVS) provides depth or disparity information using images provided by the left and right cameras. Dense or sparse depth/disparity maps which are computed by global [36] or semi-global [37] matching approaches can be used to build 3D information of the environment. By obtaining the 3D information, any kinds of motion can be detected theoretically, even the case of degenerate motion mentioned above. In [38, 39], 3D points cloud are reconstructed from linear stereo vision system first and then objects are detected based on a spectral clustering technique from the 3D points. Common used methods for MOD in stereo rig can be divided into sparse feature based [40, 41, 27] and dense scene flow based approaches [42, 16, 43].

The sparse scene flow has been used to detect the moving objects in [41]. First,

[1] The 3D point moves along the epipolar plane formed by the two camera centers and the point itself, whereas its 2D projections move along the epipolar lines.

the scene flow for each world point is computed by using the matched and tracked features in five stereo frames. Then the world points are clustered into different groups according to their average velocities in the past five frames. Finally, global nearest neighbor (GNN) based objects association is applied to refine the detection by removing some false detections. Sparse feature based methods can be easily realized in real-time applications. However, some moving objects can not be detected if few features have been detected on them.

In [42], a prediction of the optical flow between two consecutive frames is calculated based on a function of the current scene depth and ego-motion. From the difference between the predicted and measured flow fields, large non-zero regions are classified as potential moving objects. Although this motion detection scheme provides dense results, the system may be prone to produce a large number of false positives or miss detections due to the noise involved in the perception task. Two other improved approaches have been developed in [43] and [16] to remove some false detections by considering the uncertainties of 3D scene flow [43] and 2D real optical flow [16] respectively. However, they just roughly model the uncertainty of the ego-motion obtained from other sensors (GPS or IMU). In fact, the camera ego-motion has a global influence on the predicted optical flow, therefore, its uncertainty should be well considered to improve the detection performances.

Based on the related works [42, 43, 16] mentioned above, we proposed an improved MOD approach which is only based on two consecutive stereo images, where no other sensor information is required. Furthermore, we detail how the ego-motion uncertainty may be taken into account to improve the predicted optical flow computation. Then the ego-motion and disparity uncertainties are incorporated into a probabilistic framework to compute the motion likelihood for each pixel. Finally, a graph-cut framework is applied to segment the moving objects globally by using the depth information and the motion likelihood.

2.1.2 Chapter Outline

Fig. (2.2) gives the outline of the proposed moving objects detection and segmentation approach. This proposed approach can be divided into three main steps, which have been shown in three different rectangles. The first step is presented in Section 2.2, where motion likelihood for each pixel is computed by considering the uncertainties in ego-motion estimation and disparity computation. Then a graph-cut based objects segmentation is presented in Section 2.3. Third, bounding box surrounding each object is generated in Section 2.4. Real experimental results on

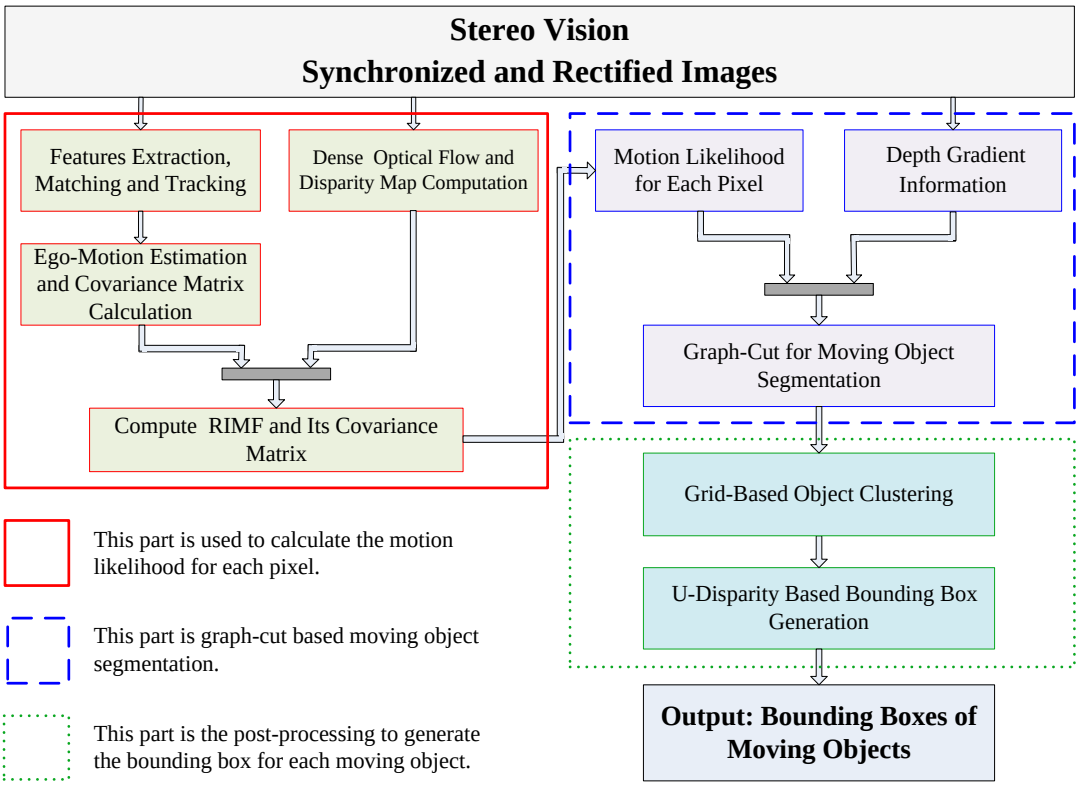


Figure 2.2: *Framework of moving object detection and segmentation*

public image sequences are presented in Section 2.5. Finally, the chapter ends with a short conclusion.

2.2 Vision-Based Moving Pixels Detection

Moving objects detection using moving cameras is still an open problem in the field of robotics and intelligent vehicles. The main difficulty is caused by the motion of the cameras. Due to the change of camera relative pose, each pixels value in the image sequences evolves in time. As said previously, the optical flow is generated by both the camera motion and the real 3D objects's motion. In order to obtain the real objects motion, an intuitive idea is to compensate the camera motion first. In order to describe the problem clearly, three motion flow based definitions are given:

- The *Measured Optical Flow* (MOF) represents the optical flow estimated using image processing techniques [44, 45]. Although the resulting flow is an estimation of the real apparent motion, we consider it in the thesis as the real optical flow given a confidence map on the obtained values.

- The *Global Image Motion Flow* (GIMF) represents the image pixel changes

caused by the relative camera motion only. This global image motion flow can be calculated for each image pixel using the relative camera motion and the depth information from the disparity map.

– The *Residual Image Motion Flow* (RIMF) is used to measure the difference between MOF and GIMF.

The RIMF can be used to distinguish between moving and non-moving pixels. In order to calculate the RIMF, the MOF and GIMF should be computed first. One can notice that the computation of the GIMF needs both the camera motion (ego-motion) and depth value of the pixel. The computation of the dense optical flow [44] and disparity map [46] are not involved in this work and we just used the results from the existing methods. Detailed introduction of dense optical flow and disparity is present in Appendix. A. The ego-motion of the camera has also to be estimated from the images because no other sensors are used. In the following section, we will introduce the estimation of the camera motion from two consecutive stereo pairs.

2.2.1 Ego-Motion Estimation

The stereo images are recorded by a SVS located on a mobile platform (vehicle or robot). The whole SVS is considered to be fully calibrated. After the stereo images rectification process [47], the left and right images are coplanar with only a translation in the X axis of b value, known as the baseline. Additionally, the left and right rectified images have identical focal length f and principal point coordinates as $p_0 = (u_0, v_0)$. As depicted in Fig. (2.3), the world coordinate system origin is assumed to be coincident with the left camera coordinate system origin. The Z-axis coincides with the left camera optical axis and is pointing forward, the X- axis is pointing right and Y-axis is pointing down. All the coordinate systems are right handed.

At each time step, the two images are synchronously obtained from the left and right cameras and two successive stereo image pairs (Fig. (2.3)) from the previous frame (at time $t-1$) and the current frame (at time t) are considered. The left image $\mathbf{I}_{t-1,l}$ in the previous frame is considered as the reference image. The right image in previous frame and the left and right image in current frame are represented as $\mathbf{I}_{t-1,r}$, $\mathbf{I}_{t,l}$ and $\mathbf{I}_{t,r}$ respectively. A 3D point $\mathbf{P}$ in the previous and current frames is noted $\mathbf{P}(X_{t-1}, Y_{t-1}, Z_{t-1})$ and $\mathbf{P}(X_t, Y_t, Z_t)$ respectively. Then we define $(u_{t-1,l}, v_{t-1,l})$, $(u_{t-1,r}, v_{t-1,r})$, $(u_{t,l}, v_{t,l})$ and $(u_{t,r}, v_{t,r})$ as the corresponding image points in the previous and current stereo frames. We assume that the

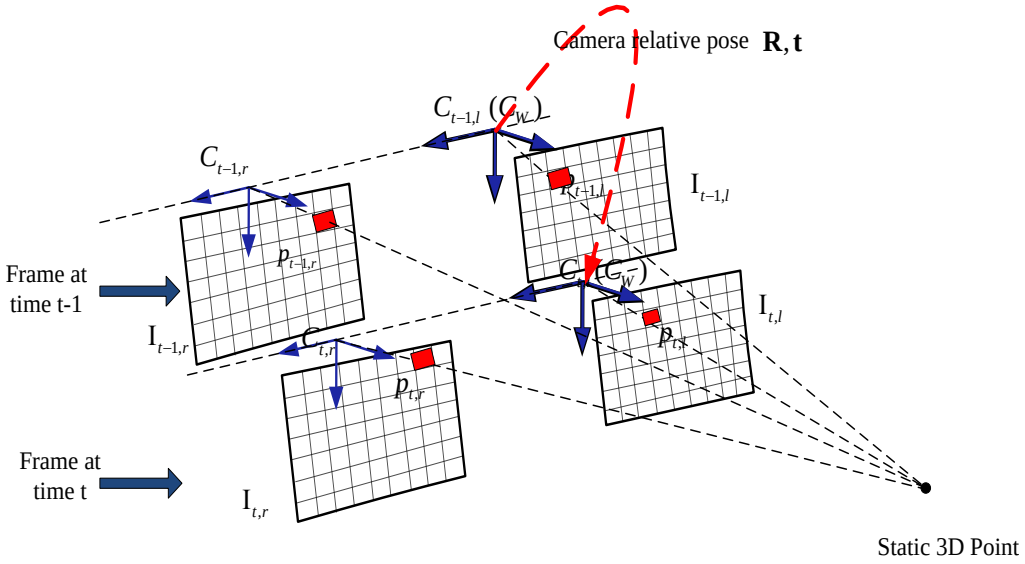


Figure 2.3: *Coordinate frames of the stereo-vision system*

stereo rig has undergone an unconstrained motion $\Theta = (r_x, r_y, r_z, t_z, t_y, t_z)^T$ between the two successive frames, where r_x, r_y and r_z are the rotational components and $\mathbf{t} = (t_x, t_y, t_z)^T$ is the translational component. Usually, r_x, r_y and r_z are also known as pitch, yaw and roll angles. This motion can also be written in the form of matrix $(\mathbf{R}|\mathbf{t})$, where $\mathbf{R}(\mathbf{r}) = \mathbf{R}_x(r_x)\mathbf{R}_y(r_y)\mathbf{R}_z(r_z)$ is the rotation matrix. Because no other sensors than cameras are considered, the relative camera pose has to be estimated based on the tracked and matched corresponding image points in the four images. The techniques for tracking and matching key points in stereo rig are introduced in the following section.

2.2.1.1 Tracking and Matching Key Points in Stereo Rig

As described in Fig. (2.4), the whole tracking and matching process in two consecutive stereo pairs can be divided into four main steps:

1. Extract key point $(u_{t-1,l}, v_{t-1,l})$ in reference image $\mathbf{I}_{t-1,l}$ based on the Alg. (B.1) in Appendix B;

2. Search the matched point $(u_{t-1,r}, v_{t-1,r})$ for $(u_{t-1,l}, v_{t-1,l})$ in image $\mathbf{I}_{t-1,r}$ using epipolar constraint (Detailed steps are presented in Alg. (B.2) in Appendix B) ;

3. Find the corresponding point $(u_{t,l}, v_{t,l})$ for $(u_{t-1,l}, v_{t-1,l})$ in image $\mathbf{I}_{t,l}$ by Lucas-Kanade [48] tracking method and find the corresponding point $(u^1_{t,r}, v^1_{t,r})$ for $(u_{t-1,r}, v_{t-1,r})$ in image $\mathbf{I}_{t,r}$ by Lucas-Kanade [48] tracking method;

4. Search the matched point $(u^2_{t,r}, v^2_{t,r})$ for $(u_{t,l}, v_{t,l})$ in image $\mathbf{I}_{t,r}$ using epipolar constraint (Detailed steps are presented in Alg. (B.2) of Appendix B) ;

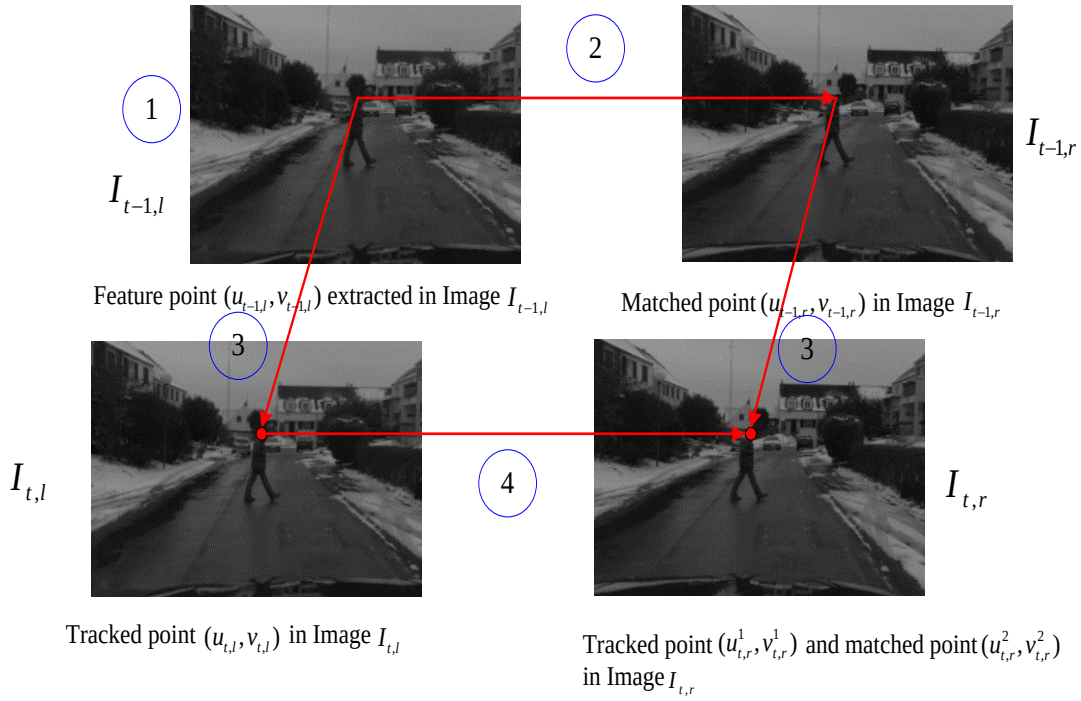


Figure 2.4: *Tracking and matching feature point in stereo rig*

Under ideal conditions, the tracked point $(u^1_{t,r}, v^1_{t,r})$ and the matched point $(u^2_{t,r}, v^2_{t,r})$ should be identical if the tracking and matching processes are correct. However, the uncertainty can not be avoided in both the tracking and stereo matching procedures. Here, we propose to use the Euclidean distance between $(u^1_{t,r}, v^1_{t,r})$ and $(u^2_{t,r}, v^2_{t,r})$ to measure the matching quality. Finally, only the points whose distance are below 0.5 pixel are accepted as correctly matched point and are packaged in $\mathbf{p}$, where $\mathbf{p}^i = (u^i_{t-1,l}, v^i_{t-1,l}, u^i_{t-1,r}, v^i_{t-1,r}, u^i_{t,l}, v^i_{t,l}, u^i_{t,r}, v^i_{t,r})^T$.

2.2.1.2 Ego-Motion Estimation

Once the key points are tracked and matched in the four views, the relative pose of the camera can be estimated by minimizing the sum of the reprojection errors of these key points. To do so, the 3D position of the feature point i in the previous frame are first computed using triangulation and the camera intrinsic parameters:

$$\begin{pmatrix} X^i_{t-1} \\ Y^i_{t-1} \\ Z^i_{t-1} \end{pmatrix} = \frac{b}{d_i} \begin{pmatrix} u^i_{t-1,l} - u_0 \\ v^i_{t-1,l} - v_0 \\ f \end{pmatrix}, \tag{2.1}$$

where $d_i = u^i_{t-1,l} - u^i_{t-1,r}$ is the disparity value of point i. Then, the 3D point can be transformed into the camera coordinate system at time t by applying the relative motion of stereo rig as below:

$$\begin{pmatrix} X^i_t \\ Y^i_t \\ Z^i_t \end{pmatrix} = \mathbf{R} \begin{pmatrix} X^i_{t-1} \\ Y^i_{t-1} \\ Z^i_{t-1} \end{pmatrix} + \mathbf{t}. \tag{2.2}$$

Finally, the 3D points are reprojected into the current left image using the camera intrinsic parameters and the perspective camera model [30]:

$$\lambda \begin{pmatrix} \hat{u}^i_{t,l} \\ \hat{v}^i_{t,l} \\ 1 \end{pmatrix} = \mathbf{K}\,[\mathbf{R}|\mathbf{t}] \begin{pmatrix} X^i_{t-1} \\ Y^i_{t-1} \\ Z^i_{t-1} \\ 1 \end{pmatrix} = \frac{b}{d_i}\mathbf{K}\,[\mathbf{R}|\mathbf{t}] \begin{pmatrix} u^i_{t-1,l} - u_0 \\ v^i_{t-1,l} - v_0 \\ f \\ \frac{d_i}{b} \end{pmatrix} \tag{2.3}$$

where, $\mathbf{K} = \begin{pmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{pmatrix}$ is the camera intrinsic parameters. In the same way, the 3D points are also reprojected into the current right image frame based on the right camera projection matrix and their image coordinates are defined as $(\hat{u}^i_{t,r}, \hat{v}^i_{t,r})$. In order to simplify the expression in the following, we use Pr^l and Pr^r to represent the reprojection procedure of left and right image points (nonhomogeneous coordinate form) described in Eq. (2.3) respectively. From Eq. (2.3), we can see that the image points $(\hat{u}^i_{t,l}, \hat{v}^i_{t,l})$ and $(\hat{u}^i_{t,r}, \hat{v}^i_{t,r})$ in the current frame can be predicted by the image point $(u^i_{t-1,l}, v^i_{t-1,l})$, $(u^i_{t-1,r}, v^i_{t-1,r})$ and $\mathbf{K}$ if we know the camera motion ($\mathbf{R}$ | $\mathbf{t}$). Rewriting Eq. (2.3) as a non-linear vectorial function $\mathbf{f}(.)$ as below:

$$\hat{\mathbf{x}}^i_t = \mathbf{f}(\Theta, \mathbf{x}^i_{t-1}) = \begin{bmatrix} Pr^l(\mathbf{K}, \mathbf{R}, \mathbf{t}, \mathbf{x}^i_{t-1}) \\ Pr^r(\mathbf{K}, \mathbf{R}, \mathbf{t}, \mathbf{x}^i_{t-1}) \end{bmatrix}, \tag{2.4}$$

where $\hat{\mathbf{x}}^i_t = (\hat{u}^i_{t,l}, \hat{v}^i_{t,l}, \hat{u}^i_{t,r}, \hat{v}^i_{t,r})^T$ are the predicted image points in current frame and $\mathbf{x}^i_{t-1} = (u^i_{t-1,l}, v^i_{t-1,l}, u^i_{t-1,r}, v^i_{t-1,r})^T$ are the detected image points at previous frames. The measured image points in current frame $\mathbf{x}^i_t$ have also been obtained from tracking and matching strategies which have been discussed in Sec. (2.2.1.1). In general, optimal camera motion vector $\hat{\Theta}$ can be obtained by minimizing the weighted squared error of measurements and predictions as:

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} F(\Theta, \mathbf{x}) = \underset{\Theta}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathbf{x}_t^i - \mathbf{f}(\Theta, \mathbf{x}_{t-1}^i)\|_{\Sigma}^2, \forall i = 1 \cdots N. \tag{2.5}$$

where $\mathbf{x}_t^i = (u_{t,l}^i, v_{t,l}^i, u_{t,r}^i, v_{t,r}^i)^T$ are the matched points in the current frame and $\| \cdot \|_{\Sigma}^2$ stands for the squared Mahalanobis distance according to the covariance matrix Σ.

2.2.2 Uncertainty Propagation

Although the optimal motion vector $\hat{\Theta}$ can be obtained from the minimization of Eq. (2.5), its accuracy also depends on the precision of the matched and tracked features' locations in the images. From the Eq. (2.4), the feature noise in $\mathbf{x}_{t-1}$ has been propagated to predicted image points $\hat{\mathbf{x}}_t$ in current frame. From Eq. (2.5), we can see that the uncertainty of the ego-motion parameters implicitly comes from the noise of both the predicted and the matched image points ($\hat{\mathbf{x}}_t$ and $\mathbf{x}_t$) in the current frame.

2.2.2.1 Statistical Model

Let's define $\mathbf{x} = [\mathbf{x}_{t-1}, \mathbf{x}_t] \in \mathbb{R}^{8N}$ that represents all the key points and $\mathbf{x}_t \in \mathbb{R}^{4N}$, $\mathbf{x}_{t-1} \in \mathbb{R}^{4N}$ that represent the image points at current and previous time instants, respectively. To be robust against outliers (mismatched features or features on moving objects), a RANSAC (RANdom SAmple Consensus) strategy is applied to estimate the relative pose between two successive frames. All inliers (whose squared error of measurements and predictions is less than 1 pixel) are used for the further estimation to get the final parameters of the ego-motion Θ. By assuming that all inliers considered in the final optimization are good matched pixels features with only additive Gaussian noise, the features follow a Gaussian probability density function:

$$\mathbf{x} \sim N\left(\begin{bmatrix} \mu_{\mathbf{x}_{t-1}} \\ \mu_{\mathbf{x}_t} \end{bmatrix} , \begin{bmatrix} \Sigma_{\mathbf{x}_{t-1}} & 0 \\ 0 & \Sigma_{\mathbf{x}_t} \end{bmatrix} \right), \tag{2.6}$$

where μ and Σ are the mean and the covariance of the features at the current and past time instants.

The Gauss-Newton optimization of Eq. (2.5) can converge rapidly if the starting point is close to the optimal point and the final optimal result is close to the true results. However, the optimal estimates always differ from the true ones (real camera

motion) due to the noise in the observed values. For a real vision-based system, we need both a robust estimation of the camera motion and a measurement of the uncertainty associated with this solution. To estimate the parameters uncertainties, the most classical approach is the sampling [49], such as Monte-Carlo approaches. However, these methods are very slow, which is obviously a drawback for real-time systems. Therefore, approximate approaches have been proposed to find a balance between accuracy and efficiency.

2.2.2.2 Approximate Approach

In [50] and [49], the authors proposed a derivation of the covariance matrix using the following model:

$$\Sigma_\Theta = \left(\frac{\partial g}{\partial \Theta}\right)^{-1} \left(\frac{\partial g}{\partial \mathbf{x}}\right)^{T} \Sigma_{\mathbf{x}} \left(\frac{\partial g}{\partial \mathbf{x}}\right) \left(\frac{\partial g}{\partial \Theta}\right)^{-T} \tag{2.7}$$

where $g(\mathbf{x}, \Theta) = \frac{\partial F(\mathbf{X},\Theta)}{\partial \Theta}$ is the gradient vector of $F(\Theta, \mathbf{x})$ respect to Θ and $\Sigma_{\mathbf{x}}$, which has been defined in Eq. (2.6), is the covariance matrix of the measured features at current and previous frames. The partial derivatives, $\frac{\partial g}{\partial \Theta}$ and $\frac{\partial g}{\partial \mathbf{x}}$ of $g(\Theta, \mathbf{x})$ are presented as below:

$$\frac{\partial g}{\partial \Theta} = \frac{\partial^2 \mathbf{F}}{\partial \Theta^2} = 2\left(\frac{\partial \mathbf{f}}{\partial \Theta}\right)^{T} \Sigma_{\mathbf{x}_t}^{-1} \frac{\partial \mathbf{f}}{\partial \Theta} - 2\frac{\partial^2 \mathbf{f}}{\partial \Theta^2} \Sigma_{\mathbf{x}_t}^{-1} \left(\mathbf{x}_t - \mathbf{f}\left(\Theta, \mathbf{x}_{t-1}\right)\right) \tag{2.8}$$

$$\frac{\partial g}{\partial \mathbf{x}} = \frac{\partial^2 \mathbf{F}}{\partial \Theta \partial \mathbf{x}} = \begin{pmatrix} \frac{\partial^2 \mathbf{F}}{\partial \Theta \partial \mathbf{x}_t} \\ \frac{\partial^2 \mathbf{F}}{\partial \Theta \partial \mathbf{x}_{t-1}} \end{pmatrix} = \begin{pmatrix} -2\Sigma_{\mathbf{x}_t}^{-1} \frac{\partial \mathbf{f}}{\partial \Theta} \\ \frac{\partial^2 \mathbf{F}}{\partial \Theta \partial \mathbf{x}_{t-1}} \end{pmatrix} \tag{2.9}$$

$$\frac{\partial^2 \mathbf{F}}{\partial \Theta \partial \mathbf{x}_{t-1}} = 2\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}_{t-1}}\right)^{T} \Sigma_{\mathbf{x}_t}^{-1} \frac{\partial \mathbf{f}}{\partial \Theta} - 2\frac{\partial^2 \mathbf{f}}{\partial \Theta \partial \mathbf{x}_{t-1}} \Sigma_{\mathbf{X_t}}^{-1} \left(\mathbf{x}_t - \mathbf{f}\left(\Theta, \mathbf{x}_{t-1}\right)\right). \tag{2.10}$$

The detailed calculation process is introduced in Appendix C.

First-Order Covariance Propagation

Equations (2.8) and (2.10) are both a sum of two terms, a first part with only first-order derivatives and a second part which is the product of second-order derivatives and the residual part $\mathbf{x}_t - \mathbf{f}(\Theta, \mathbf{x}_{t-1})$. If the final solution of Eq. (2.5) has a zero residual or very small one, the last term from Eq. (2.8)-(2.10) can be removed to get a first-order estimate as below:

$$\Sigma_\Theta = \left(\frac{\partial \mathbf{f}}{\partial \Theta}^T \Sigma_{\mathbf{x}_t}^{-1} \frac{\partial \mathbf{f}}{\partial \Theta} \right)^{-1} + A^T \Sigma_{\mathbf{x}_{t-1}} A, \tag{2.11}$$

where $A = \frac{1}{2} \left(\frac{\partial \mathbf{f}}{\partial \Theta}^T \Sigma_{\mathbf{x}_t}^{-1} \frac{\partial \mathbf{f}}{\partial \Theta} \right)^{-1} \left(\frac{\partial \mathbf{f}}{\partial \Theta} \right)^T \Sigma_{\mathbf{x}_t}^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{x}_{t-1}}$. The first right-hand term of Eq. (2.11) is the classical backward propagation approach [30]. Here, we call it *partially measured method* because only the current frame feature noise has been considered in it.

Second-Order Covariance Propagation

However, due to the noise of the measured features in the previous and the current frames, the residuals of the final solution of Eq. (2.5) may not be zero. So a second-order error propagation model which considers these residuals in the error propagation will be more suitable. A second-order error propagation result can be obtained by substituting Eq. (2.8, 2.10) into Eq. (2.7).

A general outline of the ego-motion estimation and its uncertainty computation can be found in Alg. (2.1) and the detailed estimation steps can be found in Appendix C.

2.2.2.3 Simulation Experiments

In section 2.2.2, several methods have been introduced to estimate the covariance matrix of ego-motion. In order to test their performances, we design the following simulation experiments.

Monte Carlo Experiments In the simulation experiments, both the intrinsic and extrinsic parameters of the stereo rig are known. The relative pose of the stereo cameras between the previous and the current frame is fixed before generating the image features. We first generate 3D space points according to a uniform distribution, those are then projected into the four images using the corresponding projection matrices. Considering the previous left camera as the reference coordinate system, the four projection matrices can be expressed respectively as: $\mathbf{P}_{t-1,l} = \mathbf{K}[\mathbf{I}|\mathbf{0}]$, $\mathbf{P}_{t-1,r} = \mathbf{K}[\mathbf{I}|\mathbf{s}]$, $\mathbf{P}_{t,l} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$, $\mathbf{P}_{t,r} = \mathbf{K}[\mathbf{R}|\mathbf{t}+\mathbf{s}]$, where $\mathbf{s} = (-b, 0, 0)^T$ is the translation vector between the left and right cameras. The features that appear in all the four camera images are kept. Then, a bucketing technique is applied to make the features distributed uniformly in the image plane. Finally, five points in each block are randomly selected to form the final measured features.

Algorithm 2.1 Ego-motion extraction and error propagation

Input: - Stereo image pairs $\mathbf{I}_{t-1,l}$,$\mathbf{I}_{t-1,r}$, $\mathbf{I}_{t,l}$ and $\mathbf{I}_{t,r}$ at previous and current frames;
- RANSAC iteration number N;
- Gauss-Newton iterative end criterion ξ;
- Covariance matrix of Matched features;

Output: - Camera relative pose parameters Θ (or $\mathbf{R}$ and $\mathbf{t}$) and its covariance Σ_Θ;

1: ▶ Compute the 3D point at previous frame using Eq. (2.3);
▷ RANSAC process to remove outliers;
2: **for** $i = 1$ **do** N ▷ N is maximum RANSAC times
3: ▶ Randomly select 3 matched features pairs;
4: ▶ iter = 0;
5: **while** iter< 20 || Gauss-Newton increment $> \xi$ **do**
6: ▶ Compute Jacobian matrix and residual matrix;
7: ▶ Update Θ using Gaussian-Newton iteration approach ;
8: **end while**
9: ▶ Record Θ and inliers indexes if we have more inliers than before;
10: **end for**
11: ▶ Refine the final parameters using all the inliers;
▷ Compute covariance matrix Σ_Θ for Θ;
12: ▶ Compute second partial derivatives of $F(\Theta, \mathbf{x})$ w.r.t. Θ using Eq. (2.8);
13: ▶ Compute partial derivative of $F(\Theta, \mathbf{x})$ w.r.t. Θ and $\mathbf{x}$ using Eq. (2.9, 2.10);
14: ▶ Compute the covariance matrix Σ_Θ using Eq. (2.7);
15: ▶ return Θ and Σ_Θ

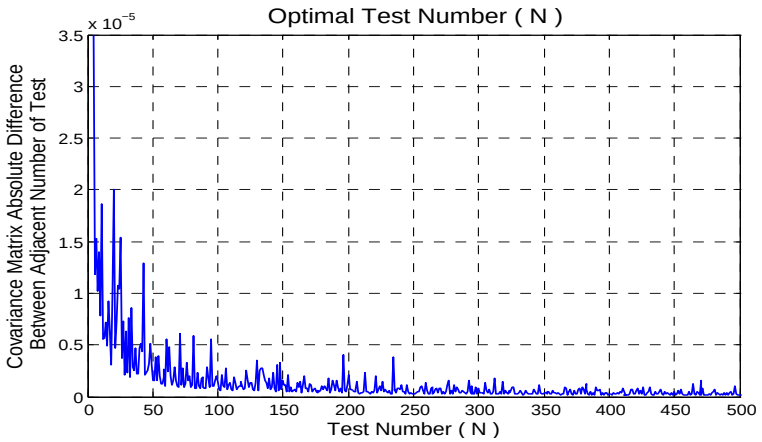


Figure 2.5: *Optimal test number.*

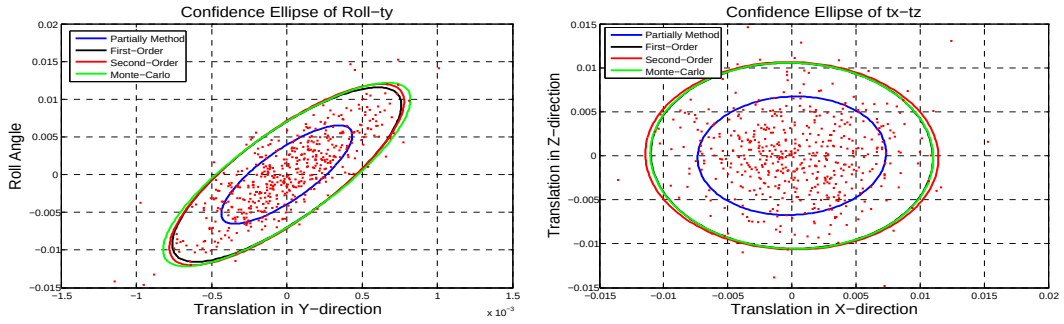
Monte-Carlo experiments are usually used to obtain the distribution of an unknown probabilistic entity by repeatedly running simulations many times. When the number N of times tends to infinity, the estimated parameter distribution converges to the true one. However, a large number of experiments is time-consuming and few experiments may be not enough to reflect the real distribution. Generally, the covariance matrix will gradually converge to a constant value when N increases. Here, we use the following absolute distance between two covariance matrices C^N and C^{N-1} to determine whether the Monte-Carlo experiments is more or less converging to the constant value. Here, C^{N-1} and C^N are the covariances computed from the $N-1$ and N experiments respectively.

$$AbsDis = \max_{i,j=1:m} |C^N_{i,j} - C^{N-1}_{ij}|, \tag{2.12}$$

where i and j are the element's subscripts of covariance matrix $C_{m\times m}$. Fig. (2.5) depicts the absolute distance changes with the increasing of the experiments number. From Fig. (2.5), we can see that the covariance matrix has a large fluctuation when N is small and this fluctuation decreases rapidly when N is large enough. Here we choose $N = 500$ for the following experiments because $AbsDis$ value is very close to 0 there.

A Monte-Carlo experiment is used to obtain an estimate of the covariance matrix as the ground truth and the detailed procedure of Monte-Carlo experiment are introduced in Appendix C. At each time, the measured features are generated using Eq. (2.6) and used as inputs in Eq. (2.5) to obtain the optimal parameters in Θ. Covariance matrices using the Monte-Carlo method can be calculated from N independent estimates of Θ. The way to obtain the minimum number N is described

in Fig. (2.5).



(a) *Confidence ellipse of* r_x-t_y **(b)** *Confidence ellipse of* $t_x - t_z$

Figure 2.6: *Performance comparison with ground truth*

Performances Comparison Simulation experiments have been considered to compare the performances of four different approaches described in Section 2.2.2. A qualitative comparison of these covariance estimates is shown in Fig. (2.6). Fig. (2.6) displayed 95% confidence ellipses for covariance between r_x and t_y and covariance between t_y and t_z in (a) and (b) respectively. In Fig. (2.6), the Monte-Carlo experiment and partially measured method results are represented by the green (dotted line) and blue (solid line) ellipses, while the black (dashed line) and red (dash-dot line) curves represent the results from the first-order and second-order methods respectively. The red dots are independent estimates of Θ from the Monte-Carlo experiment. Fig. (2.6) clearly shows that the first and second-order methods perform better than the partially measured technique. Second-order method performs slightly superior to the first-order method and both perform almost as well as the Monte-Carlo approach.

The covariances have also been quantitatively evaluated by counting the number of samples from the true distribution (red dots in Fig. (2.6)) that lie within the 1, 2, and 3-σ probability contours. Ideally, this fraction is approximately equal to the fraction of samples contained in the Monte-Carlo probability contours. In Fig. (2.7), the y - axis represents the percentage of samples contained in the confidence regions. From Fig. (2.7) we can also clearly see that first and second-order methods give similar results as Monte-Carlo experiment while significantly better than classical partially measured technique. From the simulation experiments above, we found that the first and second-order methods can achieve as good results as the Monte-Carlo approach. We used first-order method in all our experiments because first-order method is much more efficient than second-order method with similar results.

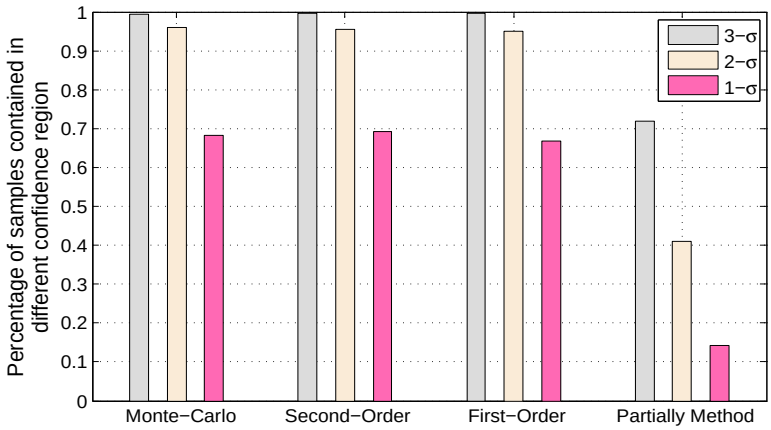


Figure 2.7: *Performance evaluation*

2.2.3 Moving Pixel Detection

In the beginning of Section 2.2, RIMF has been proposed to detect moving pixel. In this section we will explicitly present how to calculate RIMF and how it can be used for detection. In order to compute the RIMF, GIMF should be estimated first. In addition, the uncertainty of RIMF can also be computed from the uncertainties of the ego-motion and disparity.

2.2.3.1 Global Image Motion Flow

GIMF is used to represent the image motion flow caused by the camera motion. Given a pixel position $\mathbf{p}_{t-1} = (u_{t-1}, v_{t-1}, 1)^T$ in the previous image frame, we can predict its image location $\mathbf{p}_t = (u_t, v_t, 1)^T$ in the current frame [30]:

$$\mathbf{p}_t = \mathbf{KRK}^{-1}\mathbf{p}_{t-1} + \frac{\mathbf{Kt}}{z_{t-1}}. \tag{2.13}$$

Theoretically, we can predict the image location correspondences of the 3D static points in the current frame using the depth information at previous frame and the relative motion information of the camera only. However, this prediction is only true when the 3D point comes from the static objects; it does not hold for the moving objects. Finally, the GIMF $\mathbf{g} = (g_u, g_v)^T$ for point (u, v) caused by the camera motion can be expressed as:

$$\mathbf{g} = \begin{pmatrix} g_u \\ g_v \end{pmatrix} = \begin{pmatrix} u_t - u_{t-1} \\ v_t - v_{t-1} \end{pmatrix}. \tag{2.14}$$

2.2.3.2 Residual Image Motion Flow

Then, assuming that the MOF estimated between the previous and current frame at point (u, v) is $\mathbf{m} = (m_u, m_v)^T$, the RIMF $\mathbf{q} = (q_u, q_v)^T$ is computed as:

$$\mathbf{q} = \mathbf{g} - \mathbf{m} = \begin{pmatrix} g_u - m_u \\ g_v - m_v \end{pmatrix}. \tag{2.15}$$

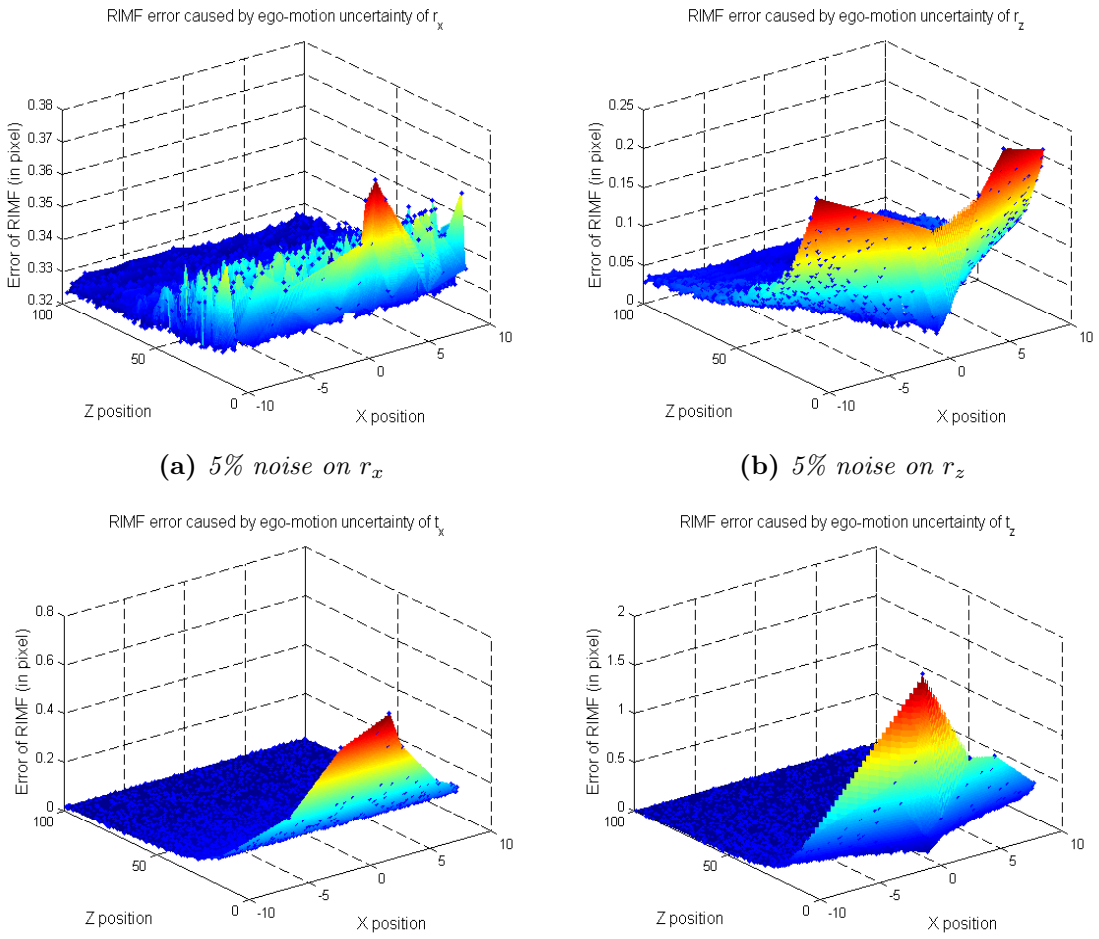
Ideally, the RIMF should be zero for a static point, while it should be greater than zero for moving points. The RIMF can be used as a cue to distinguish between moving and non moving pixels. Simply comparing the RIMF absolute difference to a fixed threshold, does not lead to a satisfying results to differentiate moving pixels from static ones since points with different 3D world locations have different image motions. Moreover, the estimated uncertainty, e.g. camera motion and pixel depth, have different influences on the image points. Ignoring these uncertainties could lead to a large number of false positive detections.

The uncertainty of the RIMF mainly comes from three parts. The first and the most important one is the uncertainty from the camera motion estimation. It is crucial because it has a global influence on each image pixel according to the Eq. (2.13). In addition, it affects differently the pixel at different locations. Moreover, different camera motions can give different influences on the RIMF results. For example, if the camera undergoes a pure translation, all the optical flow for the static objects will converge at a single point which is called the focus of expansion. The second influence part is the error of the depth estimation and the last one is the pixel location noise which results directly from the image noise (digital image quantization, image rectification, etc).

In order to know how the uncertainty of the ego-motion estimation affects the final RIMF calculation, a simulation experiment has been designed. First, we randomly set the ego-motion value and a 5%[2] Gaussian noise is added on each parameter. Then the GIMF and the RIMF are calculated using Eq. (2.13) and Eq. (2.15). The simulation results are displayed in Fig. (2.8), where the ego-motion vector is set as $\Theta = \begin{pmatrix} -0.009, & 0.007, & -0.008, & 0.052, & 0.09, & -0.20 \end{pmatrix}$. The sub-figures (a), (b), (c) and (d) show the RIMF uncertainties resulting from the noise of the different motion parameters. In order to simplify the simulation process, we assume that the uncertainty only comes from one parameter at each time and the others are accurate. The RIMF is computed based on Eq. (2.15). From these figures, we

[2]Here, we choose 5% because we want to prove that small uncertainty in the ego-motion estimation can cause big error in RIMF results.

can clearly see that the motion noise has more influence on the near points than the far points. It also has more influence on the pixels at the borders than at the center of the image when the camera undergoes a big motion in Z-axis between two consecutive frames.



(a) *5% noise on* r_x **(b)** *5% noise on* r_z

(c) *5% noise on* t_x **(d)** *5% noise on* t_z

Figure 2.8: *RIMF uncertainty generated by the ego-motion noise.*

2.2.3.3 Motion Likelihood Image

As mentioned above, the noise of RIMF is different at changing image location, and a fixed threshold does not lead to a satisfying solution to detect the moving pixels. In order to handle this problem, the uncertainty in the RIMF is propagated from the sensors to the final estimation using a first order Gaussian approximation. As in Eq. (2.15), the RIMF is a function of camera motion Θ, the pixel location (u, v) at previous frame, the disparity d and the measured optical flow (m_u, m_v).

Here, the uncertainty of the measured optical flow is not considered in this work because it only affects the detection result locally. Based on the forward covariance propagation framework in [30], the RIMF covariance can be calculated up to a first-order approximation as below:

$$\Sigma_{RIMF} = \mathbf{J}\Sigma\mathbf{J}^T,$$

where $\mathbf{J}$ represents the Jacobian matrix with respect to each input variable (e.g., the camera motion Θ, pixel position (u, v) and the disparity value d in the previous frame) and Σ is the covariance matrix of all the input variables.

$$\mathbf{J} = \begin{pmatrix} \frac{\partial q_u}{\partial r_x} & \frac{\partial q_u}{\partial r_y} & \frac{\partial q_u}{\partial r_z} & \frac{\partial p_u}{\partial t_x} & \frac{\partial q_u}{\partial t_y} & \frac{\partial q_u}{\partial t_z} & \frac{\partial q_u}{\partial u} & \frac{\partial q_u}{\partial v} & \frac{\partial q_u}{\partial d} \\ \frac{\partial q_v}{\partial r_x} & \frac{\partial q_v}{\partial r_y} & \frac{\partial q_v}{\partial r_z} & \frac{\partial q_v}{\partial t_x} & \frac{\partial q_v}{\partial t_y} & \frac{\partial q_v}{\partial t_z} & \frac{\partial q_v}{\partial u} & \frac{\partial q_v}{\partial v} & \frac{\partial q_v}{\partial d} \end{pmatrix} \tag{2.16}$$

$$\Sigma = \begin{pmatrix} \Sigma_\Theta & \mathbf{0}_{6\times3} \\ \mathbf{0}_{3\times6} & \Sigma_o \end{pmatrix},$$

where Σ_Θ is the covariance matrix of the motion parameters which has been estimated in the Section 2.2.2. Detailed information about the computation of $\mathbf{J}$ can be found in Appendix C. Here, $\Sigma_o = diag\begin{pmatrix} \sigma_u^2, & \sigma_v^2, & \sigma_d^2 \end{pmatrix}$, where σ_u and σ_v are the variances which are used to describe the pixel quantization error of the camera and σ_d describes the variance of the disparity value in its estimation process. In [43], the authors proposed that the uncertainty of the disparity map could also be considered as an approximate standard Gaussian Distribution and its variance can be linearly approximated by:

$$\sigma_d(u, v) = \sigma_0 + \gamma U_d(u, v), \tag{2.17}$$

where σ_0 and γ are two constant parameters and $U_d(u, v)$ is the uncertainty on the disparity value at position (u, v). Here, the matching cost is used as a confidence measure of the disparity value (further details can be found in [51]). Compared to the variance of each parameter in Σ, the covariance among the ego-motion parameters, position and the disparity are negligible and the estimation process will not be an easy task.

Based on the Σ_{RIMF} estimated above, we can compute the likelihood of a flow vector to be moving. Assuming a stationarity world and a Gaussian error propagation, a flow vector is assumed to follow a Gaussian distribution with zero mean and covariance matrix Σ_{RIMF}. Deviations from this assumption can be found by testing this null hypothesis or the goodness of fit. At the same time, this testing can be

done by evaluating the Mahalanobis distance [52] associated to the RIMF vector:

$$\mu_{\mathbf{q}} = \sqrt{\mathbf{q}^T \Sigma_{RIMF}^{-1} \mathbf{q}}, \tag{2.18}$$

where $\mathbf{q}$ is the RIMF vector at a certain image location defined in Eq. (2.15). Since $\mu_{\mathbf{q}}^2$ is χ^2 distributed, the RIMF motion likelihood $\xi(m)$ of RIMF vector can be computed according to its $\mu_{\mathbf{q}}$ value.

In Fig. (2.9), the sub-figures (a),(b),(c) and (d) are the motion likelihood images which comes from the Mahalanobis distance $\mu_{\mathbf{q}}$. Green pixels are detected as static and red as moving. In (a), two cyclists come from the opposite direction of the host vehicle and a pedestrian moves in the same direction as the vehicle and three of them have been well detected as moving. The shadow of the moving car in the glass window has also been detected. In (b) and (c), all the moving pedestrians have been detected but there are some false alarms on the ground which come from the MOF errors in them. The results of (d) is good due to the small camera motion and all the moving objects have been well detected.

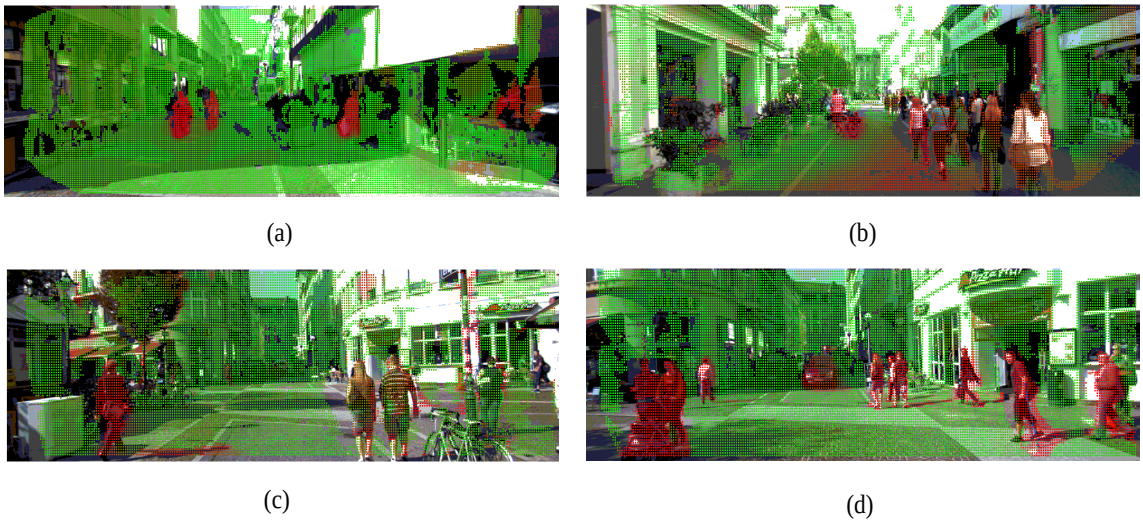


Figure 2.9: *Motion likelihood calculation based on the RIMF.*

2.3 Moving Objects Segmentation

In the previous section, a motion likelihood for each pixel has been obtained based on the RIMF and its covariance matrix. A likelihood threshold can be simply chosen to distinguish moving pixels from the static ones. However, some detection noises also exist because of the imperfect MOF. Fig. (2.10) shows some detection results using

different fixed thresholds. For example, the motion likelihood image at frame 16 is good and all the moving objects have been well detected, no matter which thresholds are used. Despite that the motion likelihood image at frame 535 is well estimated, it still has some noise on the edge of static objects due to the estimated optical flow errors. A lower threshold results in both high true positives and high false positives, oppositely a higher threshold may result in poor detection rate. Furthermore, we can not choose an optimal threshold that fits to all situations.

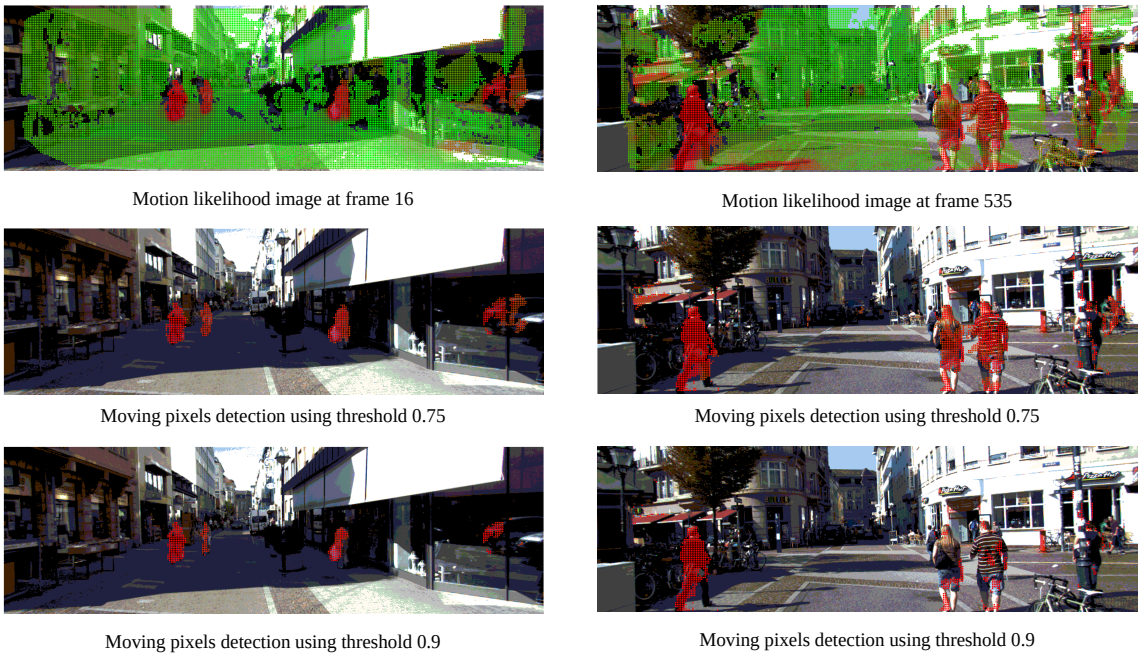


Figure 2.10: *Moving pixels detection using different thresholds*

2.3.1 Segmentation Approach

In order to effectively separate the motion foreground from the background, a segmentation step is required to consider both the motion information and the image appearance information. Usually, the segmentation of image into moving and stationary parts can be considered as a problem of assigning binary labels to each pixel,

$$l(\mathbf{x}) = \begin{cases} 1 & \text{, if pixel } \mathbf{x} \text{ is moving} \\ 0 & \text{, otherwise} \end{cases} . \tag{2.19}$$

It aims to find an optimal solution to assign each pixel to moving or non-moving. Several constraints should be considered for segmentation. Above of all, pixels with

high motion likelihood should be assigned as moving. Secondly, adjacent pixels with similar appearance and distance may belong to the same objects with the same label, otherwise their labels should be different. By considering all the constraints, the energy function can be built as

$$E(L) = E_r(L) + \lambda E_b(L) \tag{2.20}$$

where $L = \{l_1, l_2, \cdots, l_p\}$ is a binary vector, p is the number of the pixels in the image, l_i is defined in Eq. (2.19). The variables E_r and E_b are respectively called region and boundary terms. The parameter λ is used to weight the influence of both of them.

2.3.1.1 Region Term

The region term E_r captures the likelihood that the pixels belong to the moving foreground or static background. In sub-section 2.2.3, the motion likelihood of each pixel has been obtained, which can be used directly to build the region term of the energy function:

$$E_r = -\sum_{\mathbf{x}\in\Omega}\{l(\mathbf{x})\xi_m(\mathbf{x}) + (1 - l(\mathbf{x}))\xi_s(\mathbf{x})\}, \tag{2.21}$$

where Ω represents the image domain, ξ_m is the motion likelihood and ξ_s is a fixed prior likelihood to describe the point being static. If some prior motion information of the scene is available, ξ_s could be set distinctively for different image regions. This prior motion information may come from the detection results in the previous frame or categories information on the objects (e.g., ground, building and trees will be stationary while pedestrians, vehicles may be moving). It is common to assume that all the image pixels share the same stationary likelihood ξ_s if no prior is available.

2.3.1.2 Boundary Term

The boundary term is used to encourage similar neighboring pixels to be assigned to the same label. In [53, 54, 43], color is used to measure the similarity between neighboring pixels and give good performances when the foreground has different colors with the background. In [55, 56], an approach fusing color and depth is used for objects segmentation which succeeds when foreground has similar color with the background. Intuitively, the depth information is very useful for objects

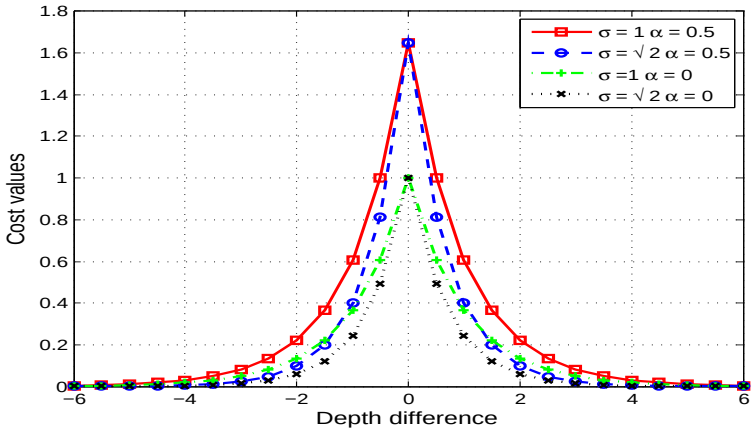


Figure 2.11: *Boundary cost function*

segmentation. The moving objects usually have important differences with their lateral background. However, the disparity map can not be directly used because the disparity values are not uniform for all pixels (it is large for the close objects and small for those that are far away). For those latter, even if the distance to the background is large, the difference in disparity may be not distinguishable. If an objects is at 50 m from the camera, and the background is at 5 m behind the objects, the values in the disparity map are 6.8 pixels and 6.18 pixels, respectively (we assume the baseline and the focal length of the stereo vision to be $b = 0.5\ m$ and $f = 680$ pixels). It is hard to get a good segmentation for such small differences. On the other hand, it will become much easier for remote objects if we use the reciprocal of disparity (in other words, the depth $z = \frac{bf}{d}$, with d being the disparity value).

Therefore, the boundary similarity can be defined as:

$$B(\mathbf{x}_i, \mathbf{x}_j) = exp(-\sigma(|z(\mathbf{x}_i) - z(\mathbf{x}_j)|) + \alpha), \tag{2.22}$$

where $z(x_i)$ and $z(x_j)$ represent the depth value at the point x_i and x_j. The function $B(\ .\ )$ is a positive, monotonically decreasing according to the depth difference between the neighboring pixels, in which σ and α are two parameters to control the descent speed and peak value respectively. In Fig. (2.11), a bigger α value gives a higher penalization cost to the depth difference, while the σ value controls the change of cost with the increase of absolute depth difference. Here we use value $\alpha = 0$ because the cost value equals to 1 when the depth difference is zeros. Meanwhile, we empirically set $\sigma = \sqrt{2}$ for all the sequences. So the energy for the boundary term can be expressed as below:

edge	weight(cost)
s-link: $n_s \rightarrow n(\mathbf{x})$	$-\xi_m(\mathbf{x})$
t-link: $n(\mathbf{x}) \rightarrow n_t$	$-\xi_s(\mathbf{x})$
neighborhood: $n(\hat{\mathbf{x}}) \leftrightarrow n(\mathbf{x})$, in which $\hat{\mathbf{x}} \in N_4(\mathbf{x})$	$\sum_{\hat{\mathbf{x}} \in N_4(\mathbf{x})} \exp(-\sqrt{2}(\lvert z(\hat{\mathbf{x}}) - z(\mathbf{x})) \rvert l(\hat{\mathbf{x}}) - l(\mathbf{x})\rvert$

Table 2.1: *Weights of Edges in* $G(n, e)$

$$E_b = \sum_{\Omega} \sum_{\hat{\mathbf{x}} \in N(\mathbf{x})} B(\hat{\mathbf{x}}, \mathbf{x}) |l(\hat{\mathbf{x}}) - l(\mathbf{x})|, \tag{2.23}$$

where $N(\mathbf{x})$ is the neighborhood pixels of $\mathbf{x}$. In order to reduce the computation time, here we use the 4-connectivity including the upper, lower, left and right neighborhoods of $\mathbf{x}$.

2.3.2 Graph-Cut Based Motion Segmentation

Let's rewrite the energy function 2.20 as below:

$$E(L) = \sum_{\mathbf{x} \in \Omega} \left\{ -l(\mathbf{x})\xi_m(\mathbf{x}) - (1 - l(\mathbf{x}))\xi_s(\mathbf{x}) + \lambda \sum_{\hat{\mathbf{x}} \in N_4(\mathbf{x})} exp(-\sqrt{2}(|z(\hat{\mathbf{x}}) - z(\mathbf{x}))) |l(\hat{\mathbf{x}}) - l(\mathbf{x})| \right\}. \tag{2.24}$$

The minimization of this energy function can be realized using the graph-cut framework. Usually, the original image is represented as a graph $G(n, e)$, where, n is defined as a set of vertices and e is the edge graph which connects two neighbour vertices. The n vertices contain two different kinds of nodes: common nodes and terminal nodes. A common node $n(\mathbf{x})$ corresponds to each image pixel $\mathbf{x}$; terminal nodes are either the source n_s or the sink n_t. This kind of graph is also called $s - t$ graph. For the motion segmentation here, s represents the stationary background while t represents foreground moving objects. Two kinds of edges exist: n-links and t-links. The n-links connect one pixel to any other neighboring pixels, using either 4-connectivity or 8-connectivity. The t-links connect all the pixels nodes to the terminal nodes. Each edge is assigned with a non-negative weight or cost. According to the energy function of Eq. (2.24), individual edge costs are defined in Table 2.1.

A subset of edges $c \in e$ is called a cut, if the terminal nodes are completely separated after removing all the edges. The cost of the cut is denoted as $|c|$, which is measured by summing up the costs of the cut (removed) edges. A minimum cut is called *min-cut*, which is the cut with the minimum cost that can be computed using min-cut/max-flow algorithms [57, 53].

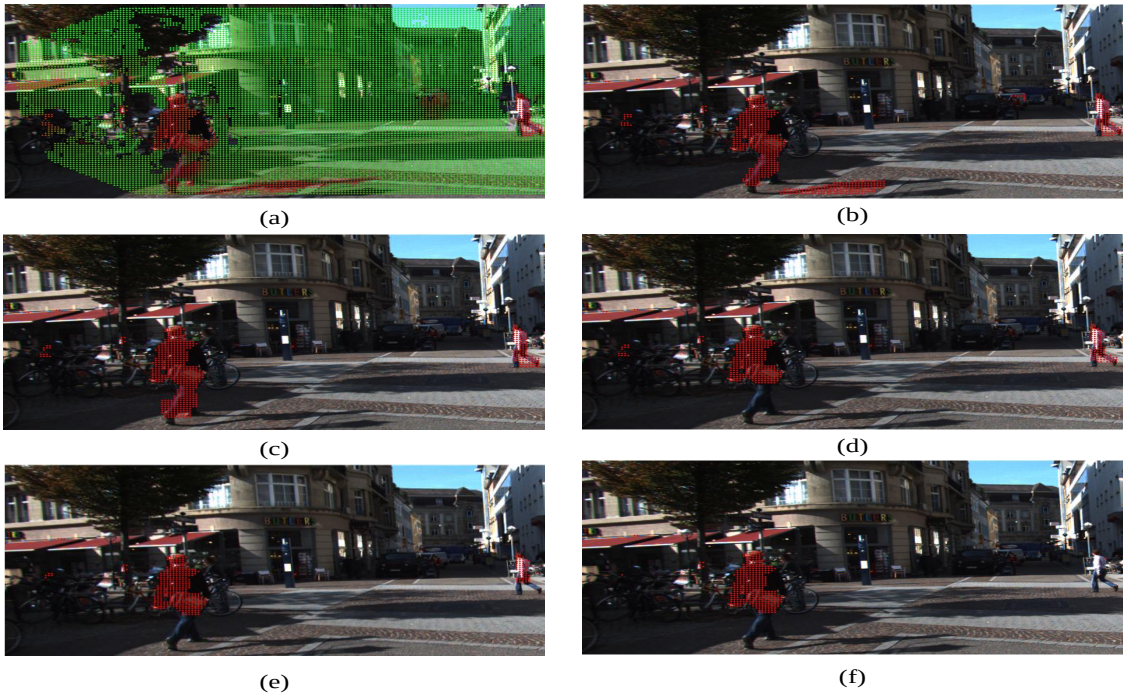
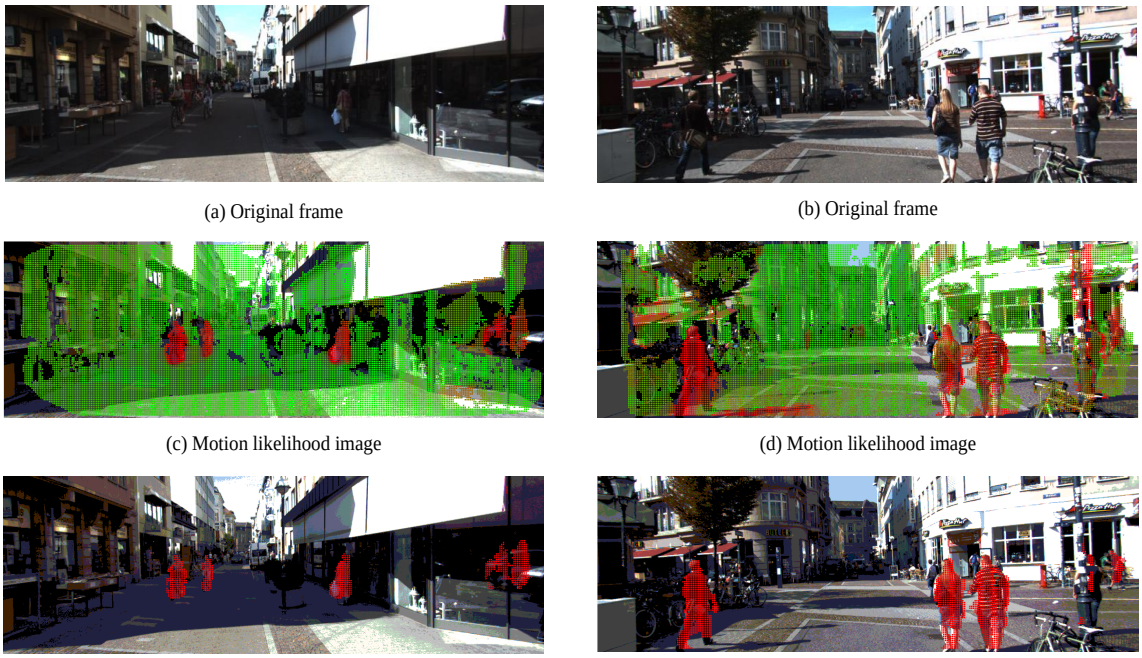


Figure 2.12: *Segmentation results from different λ values. Sub-figure (a) is the motion likelihood image and sub-figure (b) - (f) are the segmentation results when λ equals to $\{0.25, 0.5, 0.75, 2.0, 5.0\}$ respectively.*

In Eq. (2.20), λ is used to balance the influence between the region and boundary terms. Clearly, the segmentation results on the edges heavily depend on the weight parameter λ. For a low value of λ, the segmentation is mainly on the motion likelihood of a single pixel whereas a high value of λ value results in only small or no segment at all. Figure (2.12) shows the segmentation results using different λ values for the moving pedestrians. From Fig. (2.12), we can see that small edge costs result in some error detections (as in (b)), while high edge costs result in small regions (such as in (d), (e) and (f)). We choose $\lambda = 0.5$ for our experiments for two reasons. First, the segmentation results should rely more on region term (motion likelihood). When $\lambda = 0.5$, the largest boundary cost is only 0.5 for each pixel. Second, the experimental results in Fig. (2.12) also show that $\lambda = 0.5$ gives the best segmentation results.

In order to save computer memory and to improve the processing speed in the graph-cut algorithm, a down-sampling technique is used. We take one pixel for each four pixels in both row and column. Fig. (2.13) displays some segmentation results using our proposed approach.



(a) Original frame (b) Original frame

(c) Motion likelihood image (d) Motion likelihood image

(e) Segmentation result (f) Segmentation result

Figure 2.13: *Graph-cut based moving objects segmentation results in two different frames. Sub-figure (a) and (b) are the original images; (c) and (d) display the motion likelihood for each pixel; (e) and (f) are the segmentation results when* $\lambda = 0.5$.

2.4 Regions of Interest Generation

After the segmentation step, ROI should be generated for each objects for the next recognition process. Because disparity values are available for each image pixel, the spatial information could be constructed using Eq. (2.1) and then 3D ROI can be generated for each potential moving objects. Additionally, some erroneously detected pixels (e.g., shadows) can be eliminated by using the disparity value.

2.4.1 Detection Objects in 3D World Space

For our system, we mainly focus on a cubic detection space of 30 m (longitudinal), 20 m (lateral) and 3 m (height) in front of the vehicle. In this limited subspace, a density map is constructed by projecting all the detected 3D moving points on the xOz plane. The density map is associated with an accumulation buffer. A cell in the accumulation buffer covers an area of 50 mm $\times$ 50 mm on the xOz plane. The weights that the points add to the density map have a Gaussian repartition, with the maximum at the center cell and decreasing in the neighboring cells. Because points become sparser as we move away from the camera, the diameter of the patch augments gradually with the increase of the distance. The size of the patch p is

defined by the following strategy (as shown in Fig. (2.14)):

$$p = \begin{cases} 1 \times 1 \text{ cell} & z < 10m \\ 2 \times 2 \text{ cells} & 10\text{m} < z < 15\text{m} \\ 4 \times 4 \text{ cells} & 15\text{m} < z < 25\text{m} \\ 6 \times 6 \text{ cells} & 25\text{m} < z < 31\text{m} \end{cases} . \tag{2.25}$$

After obtaining the density map, an empirical threshold is chosen so as to remove sparse points, that could be mis-detected image pixels, e.g., shadow or objects borders. Here, a patch will be emptied if its point number is below a certain value. Here we set 50 as the threshold empirically. The false alarms at objects boundary are usually due to the error of the measured optical flow (smoothing constraint). Fig. (2.15) shows some ROI generation results relying on the grid-based method. Based on this approach, the shadow can be easily removed, such as in (c). In Fig. (2.15)-(c), each color corresponds to one rough clustering in the disparity map.

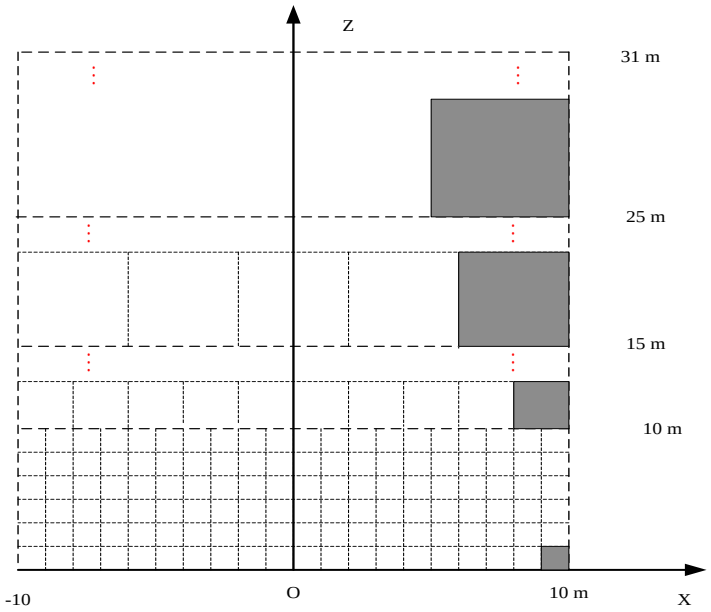


Figure 2.14: *Grid map drawing in the XoZ plane*

2.4.2 U-Disparity Map Based ROI Generation

In each cluster, the bounding box can be generated for every moving objects for the next recognition step. Region growing is used to remove the redundant detection and to integrate part detection using the dense disparity map. U-V disparity maps [58, 59], which are two variants of the classical disparity map, are often used for road

and obstacle detection. The U-disparity map has the same width than the original image, which is formed by recording the number of the pixels who share the same disparity value along each image column [59].

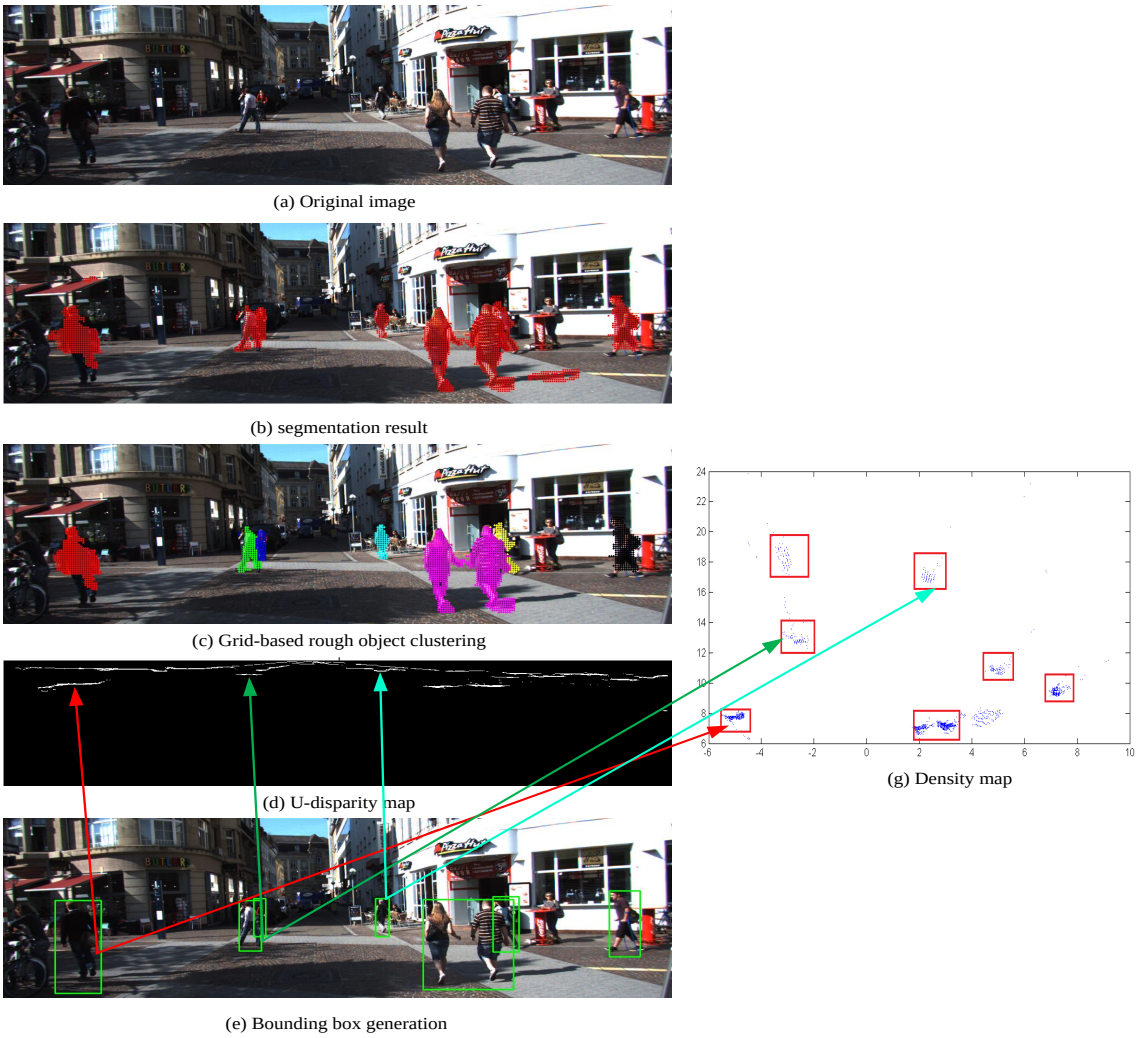


Figure 2.15: *Bounding boxes generation from moving pixels segmentation*

In the U-disparity map, an upright objects will form a horizontal line because of the same disparity value. Each white horizontal line represents a corresponding upright objects. This information can be effectively used to determine the width of the objects. After getting the width of the bounding box, region growing is applied to the neighborhood of the clustering group pixels based on the disparity value. The pixels whose disparity values are between the minimum and maximum disparity value of each cluster are considered to belong to the same objects. The final bounding boxes of the moving objects are shown in Fig. (2.15)-(e).

2.4.3 V-Disparity Map Based Clutter Reduction

According to [60], the real world height of the objects could be estimated as:

$$h_i = h_c + \frac{(y_i - y_0) z \cos\theta}{f} \tag{2.26}$$

Here, h_i and h_c are the height of objects i and height of the camera respectively in the world coordinate frame; the variable θ is the camera tilt angle and f is the camera focal length; z is the depth of the objects; y_0 and y_i are the horizon position and top of the objects in the image coordinate. Assuming that moving objects are not higher than 3 meters, some obvious false positives may be filtered. For this purpose, the horizontal position is first computed using the V-disparity map. Then, the actual height of the objects h_i is calculated using Eq. (2.26). Finally we retain only the objects whose height is between 0.75m and 3m because the height of most moving objects is in this range. Detailed steps can be found in Alg. (2.2).

Algorithm 2.2 Bounding Box Generation and Clutter Reduction

Input: - objects Bounding box;
- Camera height h_c,camera tilt angle θ and camera focal length f;
- The distance of objects to the camera z;
- Horizon position y_0;

Output: - Real world height of the objects h_i ;

1: ▶ Compute the U- and V- disparity maps;
2: ▶ According to its disparity value, each moving pixel could be assigned to different upright objects using U-disparity map;
3: ▶ Estimate horizontal line y_0 and camera tilt θ from V-disparity map;
4: ▶ Calculate real world height h_i of the objects using horizontal line y_0, camera height h_c and tilt angle θ as Eq. (2.26);
5: ▶ Keep the detection result for whose h_i is between 0.75 and 3m ;

2.5 Experimental Results on Real Data

Several KITTI video sequences[3] have been chosen to test the moving objects detection and the segmentation approach. More details about the sensor setup and data information can be found in [61, 62]. The actual objects labels and locations have been provided in some of these sequences, which can be used to evaluate our moving objects detection algorithm. In order to test the effectiveness of our approach, different environments of the sequences have been chosen for the evaluation.

[3] http://www.cvlibs.net/datasets/kitti/

2.5.1 Moving Objects Detection Evaluation

In the KITTI datasets, the video sequences are acquired from a SVS installed on the roof of a vehicle. Five different video sequences (10 frames per second) acquired in different road situations were used to test our moving objects detection algorithm. General information about these sequences are introduced below:

1. Four Inner city sequences have been selected: quiet city street, road intersections and crowd city street. Some thumbnails can be found in Fig. (2.19)-(a),(b),(c) and (d) respectively. In the inner city sequence, the host vehicle was driven at a low speed (about 15 km/h) because of complex road conditions.

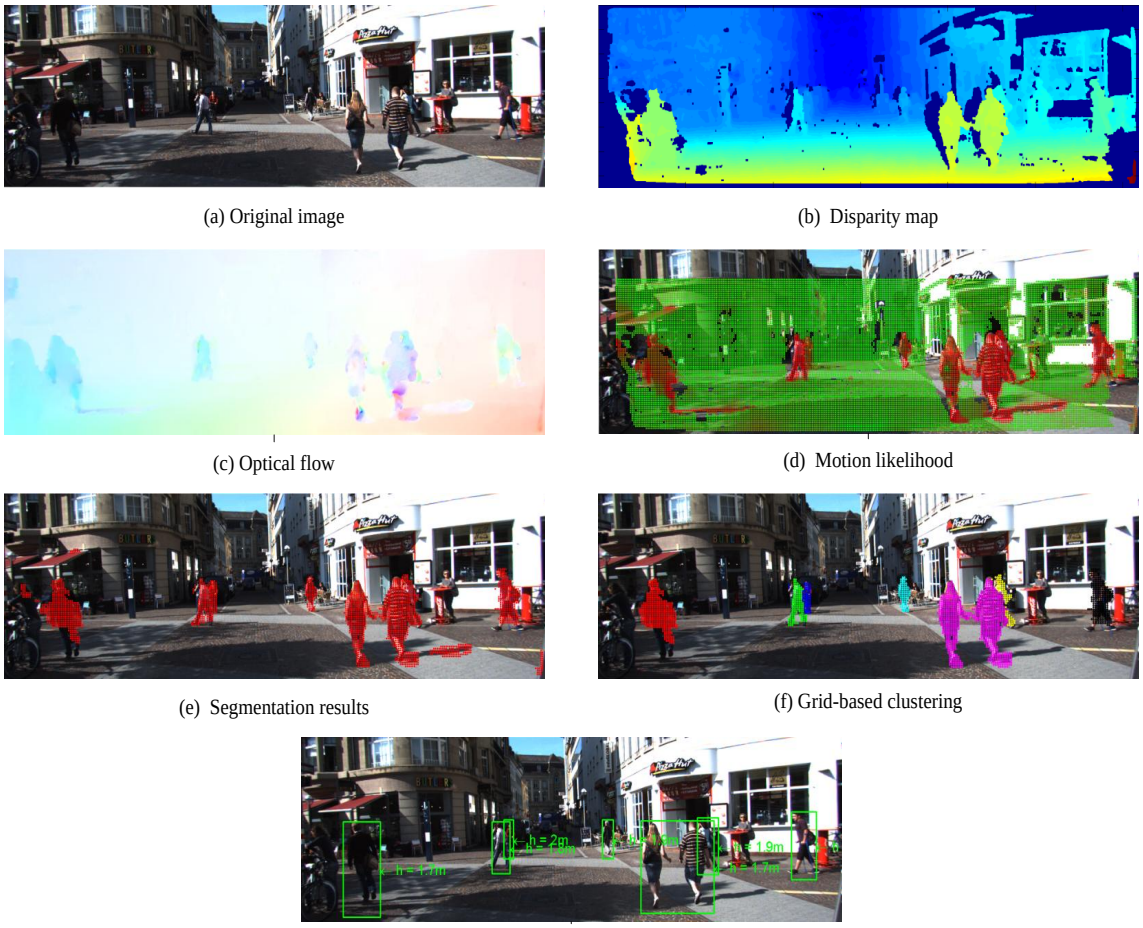
2. A campus sequence was used to test the performance of the approach when the camera undergoes a big rotation around the Y- axis.

3. A suburban road sequence was used to test our sparse feature based moving objects detection algorithm when the host vehicle drove at high speed.

Before presenting the experimental results, we review our detection algorithm as in Fig. (2.16). First, the stereo disparity map [46] and optical flow (dense [63] or sparse [64]) are computed before the moving objects detection steps. At the same time, the relative camera pose between two consecutive frames and its covariance are estimated as mentioned in sub-section (2.2.1) and (2.2.2). The standard deviation of the features in Eq. (2.6) is empirically set to $\Sigma = \begin{bmatrix} 0.75 & 0 \\ 0 & 0.75 \end{bmatrix}$ pixel. Ideally this value should be changed depending on the situation. Generally speaking, Σ should get a high value when the vehicle has a high speed, and otherwise a lower value should be set. In order to compute the variance of disparity in Eq. (2.17), we set $\sigma_0 = 0.25$ and $\gamma = 0.075$ empirically. In addition, the experimental results show that the variance of disparity has limited influence on motion detection results.

2.5.1.1 Detection Performances

Fig. (2.16)-(a) displays the original image. Fig. (2.16)-(b) and (c) show the disparity map and dense optical flow. Fig. (2.16)-(d) is the motion likelihood image, in which the stationary and moving parts are respectively displayed in green and red. Fig. (2.16)-(e) displays the segmentation results based on the graph-cut approach. Fig. (2.16)-(f) shows the grid based moving objects clustering results and different colors represent different point groups. Fig. (2.16)-(g) gives the bounding box generation result, where the height of the objects can also be given.



(a) Original image

(b) Disparity map

(c) Optical flow

(d) Motion likelihood

(e) Segmentation results

(f) Grid-based clustering

(g) Bounding box generation

Figure 2.16: *Moving objects detection steps*

Fig. (2.17) shows the detection results in the campus sequence. During this sequence, the camera turned from left to right at a high speed. The vehicle direction changes nearly of 90 degrees in 4.3 seconds. The experimental results show that our algorithm can work well in this situation. The cyclists behind the trees far from the camera can also be detected. In Fig. (2.17), the red ellipse highlights the missing detection objects. This cyclist has not been detected by our algorithm because it does not appear in the right camera and the 3D points can not be reconstructed in the disparity map. Two pedestrians at the left boundary of the second image have been included in one rectangle because they are inseparable in the disparity space.

We also tested our algorithm on a suburban highway sequence and the detection results are displayed in Fig. (2.18). On the highway, both the ego-vehicle and the objects vehicles move at a high speed, about 60 km/h. The frame rate of image sequence is 10 frames per seconds. In this case, the dense optical flow approach does not work well because of the high changes between the two successive frames.

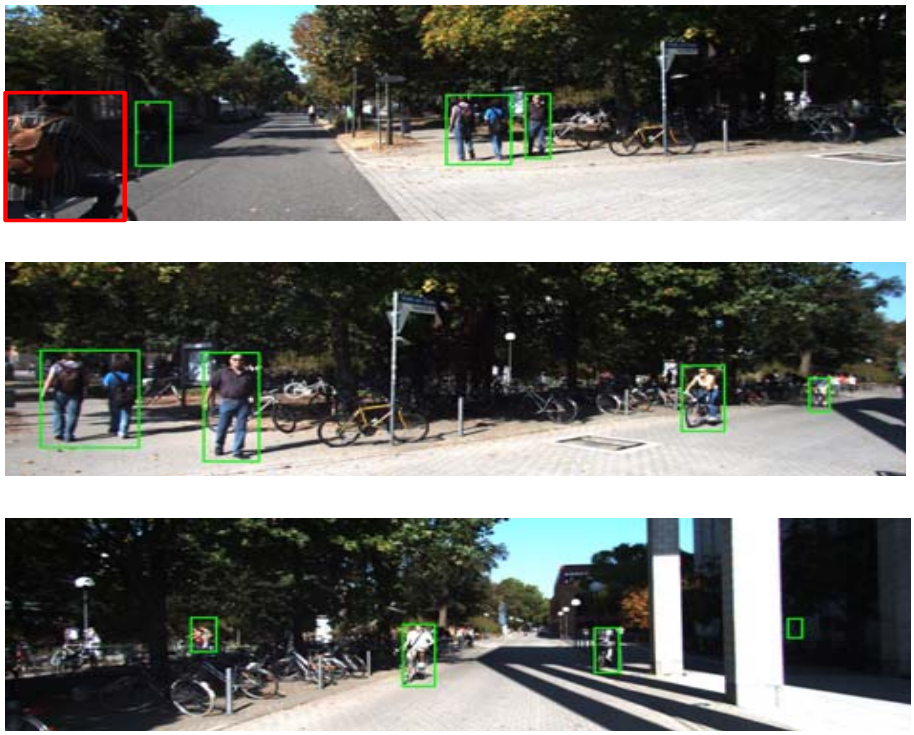


Figure 2.17: *Detection results on a campus sequence.*

Hierarchical based sparse feature tracking approach [48] can be used to handle this problem. Features tracking and matching between the two stereo frames can be realized by Alg. (B.2). A lower threshold is set in the feature extraction step to make sure that we can obtain enough features on the moving objects. The opposite direction driving vehicles were detected at a range of 40m, which remains sufficient for an appropriate reaction of the driver. The white car moving in front of the camera, was also properly detected even if it moves in the same direction as the ego vehicle.

Fig. (2.19) displays the results obtained on four different inner city sequences. In Fig. (2.19)-(a), cyclist sequence is captured around the corner of a quiet city street. In this sequence, the moving van and the cyclist appear during all the frames. Both of them have been detected by our approach in most of this sequence. The good performance of our approach benefits from the low ego vehicle speed and the relative simple street environment.

In Tab. (2.2), we give the evaluation results of our detection algorithm compared to the ground truth. The cyclist sequence includes 154 frames and the ground truth of the moving objects is given in Tab. (2.2). Here, only the moving objects whose

Road Sequence 16

Figure 2.18: *Detection results on a suburban road.*

distance is less than 30m are considered: thus the van is only considered in the first 87 frames. The evaluation method used in this thesis is briefly introduced below.

Let BB_g be the ground truth for the bounding box and BB_d be the bounding box for the detected objects. A detected $BB_d\{i\}$ and a ground truth $BB_g\{j\}$ form a potential match if they sufficiently overlap. Specifically, we employ the PASCAL challenge [65] measure to evaluate the detection results:

$$\alpha = \frac{\text{area}(BB_g\{i\} \cap BB_d\{j\})}{\text{area}(BB_g\{i\} \cup BB_d\{j\})} \tag{2.27}$$

Usually a threshold α is chosen to determine whether the detection objects is matched with the ground truth. Here we set $\alpha = 0.5$ as in PASCAL challenge [65]. In table (2.3), true positive represents the number of real moving objects bounding boxes detected in the whole sequence. False positives mean static objects that have been mis-detected as moving and false negatives are the moving objects that have not been detected. The true static objects are not taken into account because our algorithm focuses on detecting the moving objects only. The precision and recall have also been computed to measure the performance of the algorithm.

(a) Sequence 05

(b) Sequence 11

(c) Sequence 17

(d) Sequence 71

Figure 2.19: *Detection results on four different inner city sequences. (a) Cyclist sequence. (b) Crossroad 1. (3) Crossroad 2. (4) Crowded street*

It obtained a precision of 82.7% along with a recall of 93.1% in the cyclist sequence.

objects	Start frame	End frame	Number of frames [4]
Van	1	87	87
Cyclist	1	154	154
Pedestrian	1	5	5

Table 2.2: *Ground truth of cyclist sequence*

Objects	True positive	False negative	False positive[5]	Recall	Precision
Van	76	11	/	/	/
Cyclist	151	3	/	/	/
Pedestrian	2	3	/	/	/
Total	229	17	48	0.931	0.827

Table 2.3: *Detection results of cyclist sequence*

[4]The number of frames that the objects appears in.

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}} = 0.931 \tag{2.28}$$

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive} + \text{False positive}} = 0.827 \tag{2.29}$$

Fig. (2.19)- (b) and (c) show some detection results in two crossroads. The red rectangle shows a false positive detection. The detections in these two sequences are relatively easy because the ego vehicle is nearly stationary, waiting for the green lights. Especially in crossroad 2, all the moving objects have been well detected by our algorithm. The results of crossroad 1 and 2 are displayed in table (2.4) - (2.7).

Objects	Start frame	End frame	Number of frames
Car1	56	60	5
Car2	95	118	24
Car3	104	138	35
Car4	115	157	43
Car5	127	139	13
Van	135	172	38
Bus	159	209	51

Table 2.4: *Ground truth of crossroad 1*

Objects	True positive	False negative	False positive	Recall	Precision
Cars	100	20	/	/	/
Van	36	2	/	/	/
Bus	51	0	/	/	/
Total	187	22	79	0.895	0.703

Table 2.5: *Detection results of crossroad 1*

Objects	Start frame	End frame	Number of frames
Car1	1	7	7
Car2	1	23	23
Car3	19	43	25
Car4	35	58	24

Table 2.6: *Ground truth of crossroad 2*

[5]Here, we only record the number of false positive detections in the whole sequence. It is hard to say one false positive detection corresponding to a moving cyclist or to a moving van when both of them appear in the image.

Objects	True positive	False negative	False positive	Recall	Precision
Cars	79	0	/	/	/
Total	79	0	0	1.00	1.00

Table 2.7: *Detection results of crossroad 2*

The last sequence we tested is taken in a crowded street. The host vehicle moves slowly, which makes detecting moving objects easier. Slowly moving objects can well be detected by our approach, even when they move on the epipolar plane. Note that the algorithm also detected partially occluded objects because we use a dense disparity and dense optical flow maps. Some false negative and false positive detections happen in the real image sequences, as displayed in Fig. (2.19)- (d) (red bounding box) due to reflections on windows in the scene. In this sequence, the actual bounding box as of the objects have not been provided . We have built the ground truth for the moving objects from frame 500 to frame 700. The evaluation results are shown in Table (2.8)-(2.9).

Objects	Start frame	End frame	Number of frames
Pedestrian1	530	700	171
Pedestrian2	560	640	81
Pedestrian3	500	580	81
Pedestrian4	500	506	7
Pedestrian5	500	670	171
Pedestrian6	540	610	71
Pedestrian7	550	590	41
Pedestrian8	550	640	91
Pedestrian9	590	660	71
Pedestrian10	630	700	71
Pedestrian11	672	700	29
Car1	630	700	71

Table 2.8: *Ground truth of crowded street*

Objects	True positive	False negative	False positive	Recall	Precision
Total	923	33	83	0.965	0.917

Table 2.9: *Detection results of crowded street*

2.5.1.2 Computation Time

All the experiments have been realized on a standard laptop (Intel i7, 4 Core) with Matlab R2014a processing environment. When the dense optical flow is used,

the total average computational time is about 30 seconds for each frame. The dense optical flow calculation step takes about 15 seconds. Around 10 seconds are spent on the motion likelihood computation, 4 seconds on the graph-cut based segmentation and 1 seconds on the bounding boxes generation. Computing ego-motion and estimating the uncertainty only takes about 0.25 seconds. Although our Matlab implementation is not real-time, it improves a lot when compared to [66] (7 minutes per frame) and further accelerations could be achieved by C/C++ implementation with parallel/GPU computing.

2.6 Summary

In this chapter, an approach has been proposed to detect moving objects from two consecutive stereo frames. The ego-motion uncertainty is estimated through a first-order error propagation model that is used to obtain the motion likelihood for each pixel. Pixels with a high motion likelihood and a similar depth are detected as moving based on a graph-cut motion segmentation approach. Additionally, a fast recognition of moving objects becomes possible based on the segmentation results. Detection results in several different real video sequences show that our proposed algorithm is robust with respect to global (camera motion) and local (optical flow) noise. Furthermore, our approach works with all image pixels and arbitrarily moving objects (including partially occluded) can be detected. Without any tracking strategies, our detection approach gives a high recall rate and also exhibits an acceptable precision rate in several public sequences.

However, much time consumption is a big problem of the proposed method due to computation of the motion likelihood for every image pixel and the segmentation with the graph-cut algorithm. In addition, the performance of MOD highly relies on the results of dense optical flow and disparity maps. However, their estimation in a complex dynamic environment (including other moving objects) often becomes very difficult.

Chapter 3

Pedestrian Recognition

Contents

3.1 Introduction

In Chapter 2, the bounding boxes of the moving objects have been generated, and further information about these bounding boxes (such as categories, etc.) should be provided for the ADAS. Here, we only consider whether there is or not a pedestrian in the bounding box. Compared to the classical pedestrian detection problem [67, 68, 69, 70], the difference here is that we do not need to search for ROI

(the potential image patch of pedestrian) from the whole image. What we need to do is to classify the features extracted from the bounding boxes.

3.1.1 Motivation

Figure 3.1: *Pedestrian detection challenges*

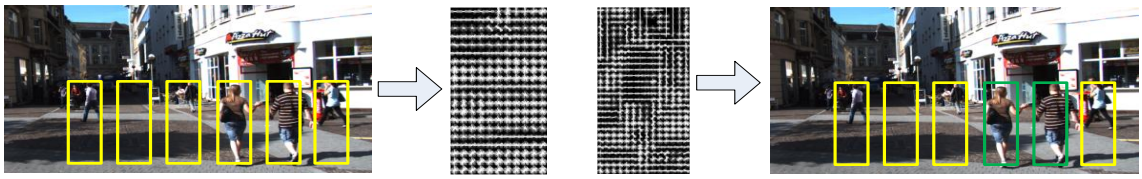
Pedestrian detection (or recognition) problem has been defined ten years ago, and more than 40 methods [71] have been proposed to handle this task. During the last decade, great progresses have been achieved due to improvement on both the computer vision techniques and machine learning strategies. Although notable achievements have been obtained by the contributions of many researchers, pedestrian detection is still an open problem in the computer vision community due to different challenges. The first challenge is the dynamic background and the variable appearances of the pedestrian. Detection in static scenes (e.g., video surveillance) is easier than in dynamic scenes since the background and the view-point do not change in time. This problem becomes much more complex when the cameras move, especially in high cluttered urban environments. The high variance in appearance, occlusions, and different poses, view points and distances present difficult problems in pedestrian detection. Some examples[1] are shown in Fig. (3.1). In real world applications, environmental conditions are also a crucial factor for the pedestrian detection and recognition systems. The image quality will be greatly reduced in bad weather conditions, such as rain or fog (the first two images in Fig. (3.2)). Under

[1] All the images come from the KITTI dataset.

Figure 3.2: *Challenges in pedestrian detection*

low illumination conditions, images tend to become noisy and blurry induced by long camera exposure times.

Most of the pedestrian detection systems can be divided into three main steps (as shown in Fig. (3.3)): ROI generation, feature extraction and pedestrian classification. Here, the ROI from the MOD step can be used for classification (recognition) directly. After obtaining the ROI, the features, such as Histogram of oriented gradients (HOG), are extracted from it to represent its image information. Then a classifier is trained offline with a large amount of labeled (positive and negative) instances based on these features. Usually, the performances of the classifiers highly depends on the training data they used. Several public pedestrian datasets such as MIT dataset [72], INRIA pedestrian dataset [67], Caltech data [18], CVC pedestrian dataset [73, 74] and KITTI [61] have been widely used for the pedestrian detection. Commonly, classifiers can achieve remarkable results if the training and testing instances come from the same dataset. However, their performances are bad when these examples come from different datasets. That is because the characteristic of images from one sequence to another are generally different. These differences (such as different image resolution, various background textures, etc) result in the features extraction difference as well. For a specific traffic scene, it is usually boring or even impossible to build a new specific dataset by labeling a lot of training instances manually. Semi-supervised learning approaches address this problem by using a small number of labeled instances and many unlabeled instances because they are cheap to obtain. Then, the class label for the unlabeled instances can be constructed by fusing the outputs of several existed object detectors. In [75, 76], the authors proposed to fuse the classification results from different experts based on the belief function theory by using some combination rules. In this chapter, we aim to develop a state-of-the-art semi-supervised pedestrian classification approach in dynamic real-world environments.



(a) ROI Generation (b) Feature Extraction (c) Classification

Figure 3.3: *Outline of pedestrian detection*

3.1.2 Chapter Outline

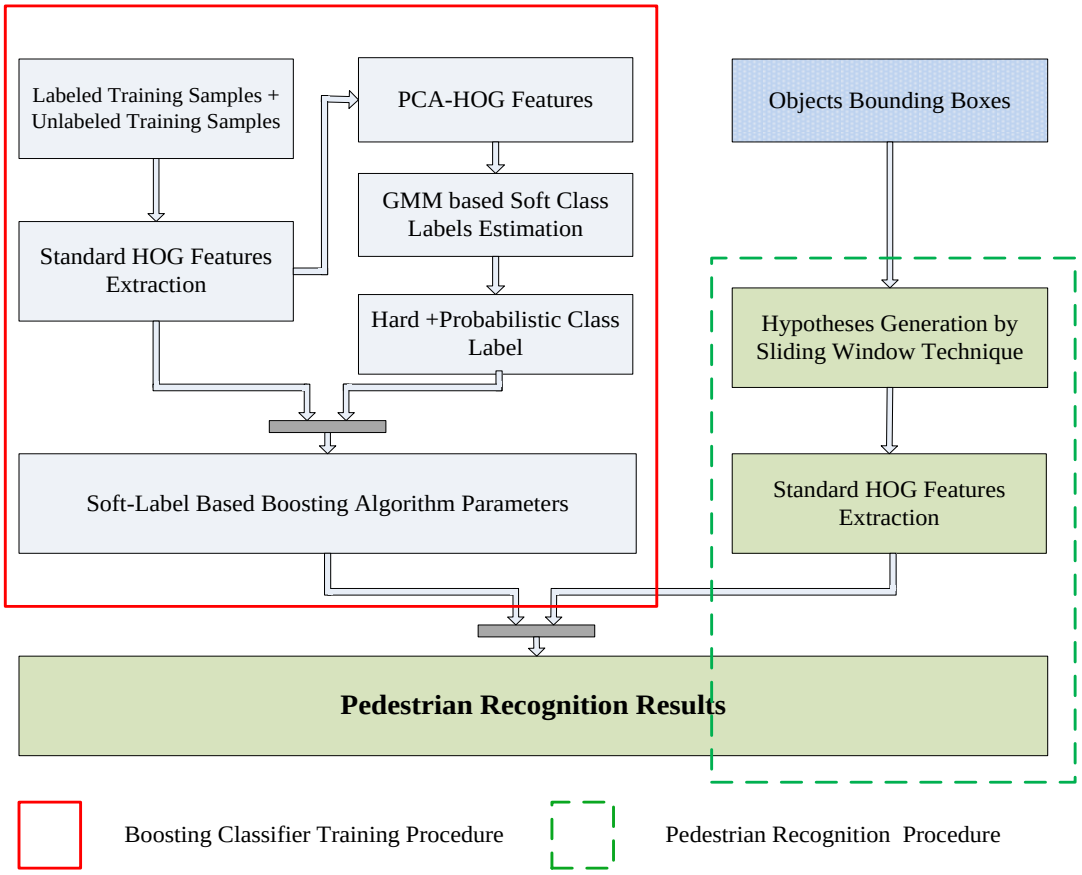


Figure 3.4: *Outline of soft-labeled based semi-supervised boosting for pedestrian recognition*

In this chapter, we describe in detail the recognition part of our moving object detection system, which is illustrated by Fig. (3.3). The left box is the semi-supervised boosting classifier training procedure and the right box gives the process of pedestrian recognition part. The structure of this chapter is organized as follows. In the beginning of Section 3.2, we first present the well known feature sets for pedestrian detection, especially the HOG features. Then PCA-HOG (Principal Component

Analysis-HOG) features which are used for soft label estimation are introduced in the remaining section. In Section 3.3, a general review of classifiers for pedestrian detection is presented first, then the proposed soft-label based boosting algorithm is explained in details. Several classification and recognition experiments are designed to test the effectiveness of the approach in Section 3.4. Finally, we give a general conclusion on our pedestrian recognition work in Section 3.5.

3.2 Features Extraction

3.2.1 Motivation

The image features are extracted for the classification step in each object bounding box. To achieve good classification performances, the image features should be invariant to changes in illumination, differences in viewpoint and shifts in object contours. Instead of using raw pixel intensities directly, one often uses some form of more advanced local image descriptors from the image part corresponding to the bounding box. The use of features has several advantages. First, features usually have more discriminative information than image intensities, such as object edges, motion, etc. Second, this strategy can be time saving because of the lower dimension of the features compared to the original image pixels. For these reasons, computing the most useful and efficient features for representing the pedestrians is crucial for pedestrian detection. A general overview of the main features used for pedestrian detection are introduced in the following paragraphs.

3.2.2 State of the Art

Local image representations have been widely used as feature sets for object detection and recognition. The basic idea is that the local image descriptors are extracted around at a sparse set of salient image points, usually called points of interest or key points. Generally, the descriptors are assembled together to construct a descriptor vocabulary or codebook for a certain object category. Then, the object detection or recognition is realized by matching new test images against this visual vocabulary. SIFT (Scale Invariant Feature Transformation) [77], which can be considered as one of the most famous local image gradient based descriptors, has been successfully applied for image stitching [78] and object recognition [79]. In the SIFT descriptor, first a local scale and a dominant orientation are obtained from the key-point detector. Then they are used to vote into orientation histograms with weighting strategy

according to their gradient magnitudes.

Stimulated by the SIFT descriptor and shape contexts [80], HOG features [67] have been proposed, which have been developed for pedestrian detection. HOG features collect gradient information in local cells[2] into histograms using tri-linear interpolation, and normalize overlapping blocks (a block is squared image area which includes four cells) composed of neighboring cells. Interpolation, local normalization and histogram binning make the representation robust to changes in lighting conditions and small variations in pose. Due to their remarkable performances for pedestrian detection, several HOG based features have been proposed. In [81], HOG and Local Binary Pattern have been combined together to form new feature sets which are capable to handle partial occlusions. In this approach, global and local detectors have been trained from the training data using linear SVM, where global detectors are used for scanning the whole window, while part detectors are applied for some special local regions. Evaluation results show visible improvements over standard HOG on the INRIA Person data set [67].

Besides the appearance, other information are also used to built the features for the pedestrian detection. In [68], the authors combine the flow and appearance of oriented histogram features together to obtain better results than only appearance based features, especially for the moving pedestrians. Furthermore, the disparity [17] in stereo vision is also an important information that can be used for pedestrian detection, because the foreground objects usually share different depth with the background. Based on this idea, a new feature is proposed based on binocular disparity. Besides the pedestrian itself, the context in its margin neighborhood can also offer helpful information. In [82], local features of the pedestrian and the neighborhood context are combined to construct a context descriptor; then, a so-called context-boost iterative classification algorithm is applied for the classification process. 3D geometric context [83], local pixel context [69] and shape context [84] are all considered to form the pedestrian descriptors.

Although most of feature descriptors are computed by based on local information, they can also be classified into global or part-based approaches depending on how they combine the underlying features. Global approaches describe a people in a fixed size window and the model is learned by sliding image windows from the images and computing feature descriptors on these windows. Note that multiple local regions (legs, shoulder, head, etc) have been included in the features of global descriptors (e.g., HOG), however, the position of these regions is fixed and cannot

[2]small squared areas uniformly divided from the instance image.

change between object instances. Usually, it is easy to train a classifier on the global features due to their less complex models. A binary classifier can usually achieve good detection performances for this kind of features, e.g., AdaBoost [85].

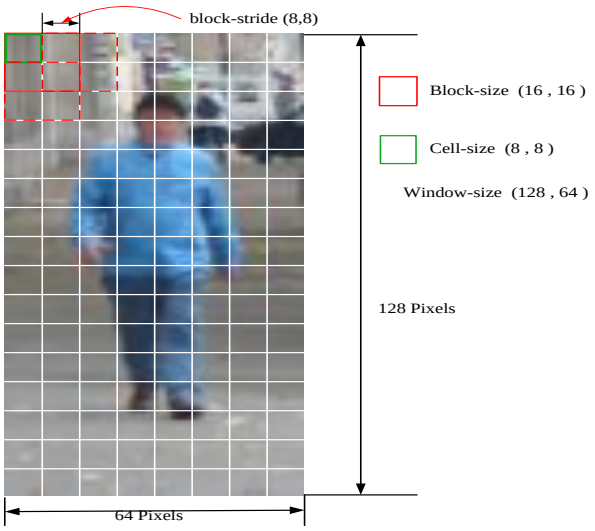
However, global approaches work poorly when the people are partially occluded, overlap or some uncommon postures, such as sitting, squat or running. In this case, part-based approaches which describe the people with different parts achieve better results. Local parts may correspond to human limbs (semantic parts), head or shoulder, typically discriminative image regions (e.g., [86]). Then, a detector or weak classifier is trained for each local parts and their outputs are combined to get the final decision. If an object is described into several parts, their spatial location-relation should also be considered. A fixed spatial layout model is easy to handle [87], however, a flexible spatial model [88, 89, 90] is much suitable for more complex situations.

3.2.3 PCA-HOG Features

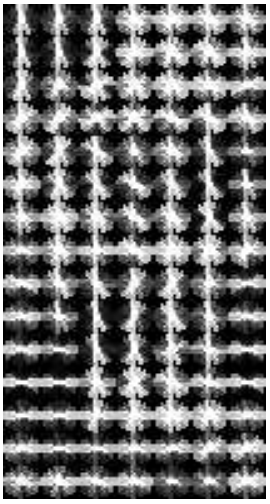
In this thesis, we adopt HOG features for the pedestrian recognition system due to two reasons. First, they have been widely used for pedestrian detection and their effectiveness have been well proved in different applications. Second, the objective of this chapter is to train a semi-supervised classifier rather than to build new features descriptors for pedestrian recognition. In order to exploit the latent information of unlabeled instances, we estimate soft class labels for them based on the labeled instances. However, it is difficult to estimate their labels using high dimension HOG descriptors directly, thus PCA is applied as a tool to project them into a lower dimension feature space.

3.2.3.1 Extraction of HOG

HOG features are calculated by taking well-normalized local histograms of image gradient orientation in a dense grid. As shown in Fig. (3.5), we will introduce how to compute the standard HOG descriptors [67] in a 64×128 pedestrian image. First, the image window is divided into cells of size of 8×8 and for each cell a local 1-D histogram is accumulated by quantizing gradient directions or edge orientations over all the pixels of the cell into 9 directions. In order to achieve invariance to illumination, all the local histograms should be normalized in larger spatial regions or "blocks" of size of 16×16 pixels. The normalization is processed by accumulating a measure of the local histogram over a block and using the results to normalize all



(a) *Cell, block, and window in HOG extraction*



(b) *Displayed HOG descriptors*

Figure 3.5: *HOG features extraction*

of the cells in it. In each block, 4 cells are included. Typically each cell is shared between several blocks (as shown in Fig. (3.5)), but their normalizations happened in different blocks. Thus, one cell will appear several times in the final output vector with different normalizations. This process seems redundant but the detection results show that this improves the performances. The finally HOG descriptor is the accumulation of the features from all the blocks of a dense overlapping grid covering the whole detection window. A final HOG features of a 64×128 window is a vector of 3780 elements.

3.2.3.2 PCA-Feature Selection

According to [91], the extracted HOG features include redundant information and using lower dimensional features leads to a model with fewer parameters and speeds up the learning and detection algorithms. This issue of dimensionality reduction (sometimes refereed to manifold learning) is also a significant problem across a wide variety of information processing fields such as pattern recognition, data machine learning, and data visualization. Various approaches have been proposed in the past years, a comprehensive review of them can be found in [92]. Among all the dimensionality reduction methods, PCA [93] is one of the most well known unsupervised techniques and it has successfully been applied to numerous computer vision problems [94]. PCA-HOG features have been used in [95] and [96] for pedestrian

detection and people counting.

Principal Components Learning Because PCA is an unsupervised approach, both labeled and unlabeled instances are used together to find the principal components. Let the training data be represented in a $n \times D$ matrix $\mathbf{X}$ consisting in n data-vectors $\mathbf{x}_i, i = 1, \cdots, n$ with dimensionality D. Vector $\mathbf{x}_i$ is the HOG descriptors of instance i. Typically, PCA wants to find a linear mapping to project the high-dimensional data into a low-dimensional data, while this mapping should maximize the amount of variance in the data. In mathematical terms, PCA aims at finding a mapping Γ which maximizes $\mathbf{\Gamma}^T \text{cov}(\mathbf{X})\mathbf{\Gamma}$, where $\text{cov}(\mathbf{X})$ is the covariance matrix of data $\mathbf{X}$. This mapping Γ can be simply computed the d principal eigenvectors (i.e., principal components) of the covariance matrix. So Γ can be obtained by solving the eigen equations of covariance matrix

$$\text{cov}(\mathbf{X})\Gamma = \lambda\Gamma \tag{3.1}$$

where, λ is the d principal eigenvalues of $\text{cov}(\mathbf{X})$. Then the low-dimensional data representation $\mathbf{y}_i$ of data $\mathbf{x}_i$ are computed by mapping Γ as

$$\mathbf{Y} = \Gamma^T(\mathbf{X} - \bar{\mathbf{X}}) \tag{3.2}$$

The low-dimensional features are called PCA-HOG features in the rest of the manuscript.

3.3 Classification

In Section 3.2, we have introduced how to extract features from an image window. Then we should learn the parameters of a discriminant model (or decision boundary) between the positive and the negative classes from the two classes training examples.

3.3.1 Motivation

Fully supervised methods always require a lot of labeled instances to reveal the true interpretation of the data. They aim at learning a mapping from the input data to the outputs. Generally, the performances of a supervised algorithm highly depends on the supervisor (labeled instances). In many applications, obtaining sufficient labeled instances for the fully supervised learning methods is difficult or even impossible. However, collecting a large set of unlabeled data instances is usually

easy, or in many applications cheap to obtain. Therefore, it is desirable to have methods which require only a minimal supervision and are able to learn from both labeled and unlabeled data. Semi-supervised learning deals with developing algorithms which can use both labeled samples and unlabeled samples. Both supervised and semi-supervised methods have achieved great success for classification problems. A general reviews of the state of the art approaches is presented in following section.

3.3.2 State of the Art

3.3.2.1 Supervised Classifiers

Boosting The boosting algorithm [97, 85] is a widely used supervised classifier for object detection. The main idea of the boosting algorithm is to built a strong classifier by combining several weak learners $f_t(x)$. The output of the final decision is given by

$$F(x) = sign(\sum_{t=1}^{M} \alpha_t f_t(x)) \tag{3.3}$$

in which, the weak learner $f_t(x)$ corresponds to a very simple classifier (for instance a stump, that is a decision tree with only one split); α_t is the weight (confidence) of tth weak learner, which is determined by the training error of this classifier. The weak classifier is achieved sequentially by changing the weight distribution of the training instances. Each training instance is associated with a weight. If this instance is misclassified by the current weak learner, its weight will be increased in the next iteration, otherwise the weight will be decreased. Thus, the weak learner will be trained with a re-weighted data set and will focus more on the instances that are hard to classify.

The original AdaBoost is also known as discrete AdaBoost because its outputs are only discrete values in $\{+1, -1\}$. A variant has been called Real AdaBoost [98] because the output of the classifiers could be a real value. This real value is the probability that a given input instance belongs to a class, considering the current weight distribution for the training set. Other modified versions, such as gentle AdaBoost [99], LogitBoost [99] and FloatBoost [100] were also proposed and a general overview of them can be found in [101].

In [102], the AdaBoost algorithm is trained by using the Haar-like features for the face detection which gives not only robust detections but also real-time performances. Based on their work, another AdaBoost approach has been proposed for

pedestrian detection [103]. In their work, a real-time detection system has been proposed by using the integral array representation [102] and boosted detector cascades technique [104]. AdaBoost cascades [70, 105] are slow in the training process to select feature encoding, but they obtain good performances in the run-time of the final detectors.

Support Vector Machine SVM are proved to be a very powerful tool for the pattern recognition (classification) in the last decades [106]. This approach attempts to separate positive and negative instances with an optimal hyperplane, i.e., an hyperplane maximizes the margin (gap) between both sets of instances in the feature space. In [67], a linear SVM trained using HOG features, was successfully applied for pedestrian detection. Other related pedestrian detection working with linear SVM can be found in [107, 108, 109].

It was extended to nonlinear classification by applying the kernel trick [110] and thus the data are mapped in a high-dimensional space using a kernel, in which their margin are linearly separated. Compared to the linear SVM, non-linear version reaches better performances. However, the computation cost and memory consumption also increase rapidly. Owning to the development of the hardware, non-linear SVMs has been also used for pedestrian classification in [111, 112, 113]. A recent work presents efficient versions of non-linear SVMs by applying a specific class of kernels [114].

Neural Networks The classification problem can be considered as a nonlinear mapping process that maps an input features vector to output object classes' space. Neural networks with a back propagation learning algorithm are well known for supervised classification and have been proved as an effective approach in pedestrian recognition system. In [115], the authors use a mix of unsupervised and supervised training to create a Convolutional Neural Network by extracting features directly from raw pixel values. The results show good detection performances on INRIA, ETH and TUD-Brussels datasets. Other works, such as [116, 117, 118], focus on using deep architectures for learning the visibility relationship among overlapping parts at multiple layers to handle the occlusion problem. A Switchable Deep Network (SDN) is proposed in [119] for pedestrian detections. By using SDN, hierarchical features, salience maps, and mixture representations of different body parts can be learned automatically together.

3.3.2.2 Semi-Supervised Learning

In supervised learning, training instances are provided together with their corresponding class labels. This entirely differs from unsupervised learning, where training instances are provided without class information. Semi-supervised learning (SSL) is a trade-off between both algorithms [120, 121, 122]. In SSL, a classifier is trained from both labeled and unlabeled instances, the amount of which is usually much higher than that of the labeled instances. In general, SSL methods can be categorized into transductive and inductive approaches. The goal of transductive learning is to infer the correct labels for the given unlabeled training data only, while an inductive learning approach aims to infer the correct mapping from data's feature space to the class label. However, the terms inductive and transductive are frequently mixed up in recent literature, such as transductive support vector machines (TSVM) [123] which share the property of inductive learning.

In SSL, one wants to improve a classifier by introducing large amounts of unlabeled instances instead of using only a small number of labeled instances. However, since there is no class label information for unlabeled data, most approaches attempt to assign a pseudo class label for the unlabeled instances by using some underlying structure assumptions, such as: the smooth assumption which assumes that data should be classified to the same class if they are very close to each other. Under this assumption, the decision boundary between the classes must pass through low density regions; the cluster assumption, which assumes that data tend to be separated into different discrete clusters and data instances in the same cluster must share the same label; the manifold assumption, which assumes that the marginal distribution is supported on a Riemannian manifold and in other words, it means that even if the data is observed in a d-dimensional feature space, the data really lies on a lower-dimensional manifold governed by only a few degrees of freedom.

However, as mentioned in [121], additional unlabeled instances are not always helpful to improve the classifier performances. Bad matching of problem structure with model assumption can lead to a degradation of classifier performances [121]. Several works that theoretically revealed this issue [124, 125]. Since SSL has been introduced, a lot of related approaches have been proposed. In the following, we will give some of the most popular SSL approaches.

Boosting and SSL Several approaches have been proposed to extend boosting to the semi-supervised setting, e.g., [126, 127, 128, 129]. Most of these approaches try to add an unsupervised regularization term to the supervised loss function of

boosting that penalizes decision boundaries passing through high density regions. Different kinds of loss functions are built for the labeled and unlabeled instances respectively. Some typical supervised boosting algorithms are introduced shortly below.

Boosting with Regularization. SemiBoost [128] and semi-supervised regularized boosting [130, 129] are two typical semi-supervised approaches. The former approach uses the manifold and clustering assumptions to assign the pseudo class labels to unlabeled instances. More precisely, the inconsistency between labeled and unlabeled instances is used to construct the loss function for the unlabeled instances. In [130, 129], the pseudo class label of an unlabeled instance is given under the margin cost functional framework [131]. Both of these approaches have been applied in the field of computer vision for objects detection [132] and facial expression recognition [129].

Boosting with Prior Information. Other semi-supervised boosting methods are based on using of prior information [133, 134, 135]. In some cases, the class labels are assigned by the expert according to some background knowledge. This kind of labeling is sometimes called soft labeling, and correspond to probability, belief degree or outputs from other classifiers. The usual error loss functions cannot be used to handle this kind of soft label. In this case, the relative entropy or Kullback-Leibler divergence can be applied to measure the difference between the output of the weak classifier and the soft class labels [126, 127, 133].

Self-Training, Co-Training and Multiview Learning Self-training is a commonly used approach for semi-supervised learning. In self-training, a classifier is first trained with a small amount of labeled data using a supervised method. Then this classifier is used to classify the unlabeled data. Finally, the unlabeled instances which were classified with a high degree of confidence are added into training set, together with their predicted labels.

Co-training [136] is a semi-supervised learning technique that requires two views of the data. In the co-training algorithm, each example is assumed to be able to be described by two different feature sets and either of them can provide different, but complementary information about the instance. Initially two separate classifiers are trained with the labeled data, on the two features sets respectively. Each classifier then classifies the unlabeled data, and "teaches" the other classifier with the few unlabeled examples (and the predicted labels) they feel the most confident with. Each classifier is retrained with the additional training examples given by the other

classifiers, and the process is repeated.

If more than two classifiers are used, the learning approach is referred to multi-views learning. For example, tri-training [137] is an extension of co-training that uses three classifiers instead of two. In [137], the authors have shown both theoretically and empirically that tri-training requires weaker conditions in order to converge, which makes it more applicable in practice.

3.3.3 Contributions

We present here our strategy for performing boosting from instances associated with probabilitics labels. First, we present how decision trees can be induced from such data. Then, we proceed with our soft label boosting procedure.

3.3.3.1 Boosting from Soft-Labeled Data

Soft-Label Decision Trees. In this section, we present how CART (Classification And Regression Tree) [138] can be extended to handle training instances associated with probabilistic class labels [139]. A decision tree is constructed by recursively splitting the parents nodes into children nodes until one or several terminal conditions are satisfied, such as instances falling into one of the children nodes are less than a certain number. The Gini index is usually employed to quantify the quality of a split and thus to find the optimal split values for constructing the decision tree [140]. The Gini index measures the node impurity by taking the diversity within the class probability estimates for a node. For a tree node λ_t , the Gini index $G(\lambda_t)$ is computed as below:

$$G(\lambda_t) = \sum_{l \neq k} p_l(\lambda_t) p_k(\lambda_t) = 1 - \sum_{k=1}^{K} p_k^2(\lambda_t) \tag{3.4}$$

in which $p_k(\lambda_t)$, $k = 1 \cdots K$ (K is the number of classes) is the class probability for current node λ_t. For a classical decision tree, this probability can be simply estimated by the frequency of each class k:

$$p_k(\lambda_t) = \frac{n_{kt}}{n_t} \tag{3.5}$$

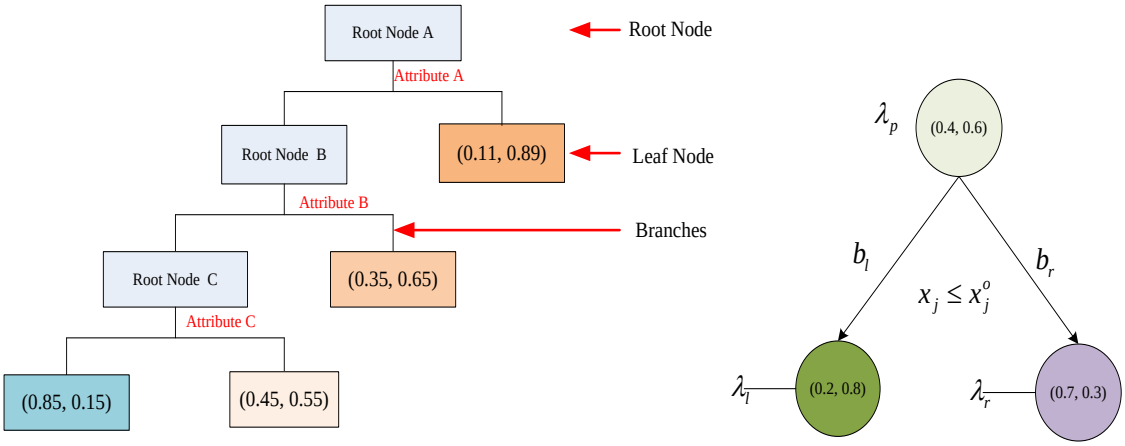
where n_{kt} is the number of instances of class k falling into node λ_t and $n_t = \sum_k n_{kt}$ is the total amount of instances falling into node λ_t. The Gini index in Eq. (3.4) depends upon the estimates of $p_k(\lambda_t)$. However, the computations of $p_k(\lambda_t)$ in

Eq. (3.5) are not appropriate for learning sets with probabilistic label because the class frequency within a node cannot be determined directly anymore, the class information being not sure. In this case, the class frequencies can be estimated from the training set by taking its expectation. Then, Eq. (3.5) can be replaced by

$$p_k(\lambda_t) = \frac{1}{n_t} \sum_{i=1}^{n_t} \pi_{i,k}, \tag{3.6}$$

where $\pi_{i,k}$ is the probability of x_i belongs to class k. The class probability estimates obtained from Eq. (3.6) can be used directly to compute the Gini index defined in Eq. (3.4).

Fig. (3.6)-(a) shows a decision tree built by soft labeled instances. The structure of the decision tree is the same as the classical decision tree, while a probability vector is assigned for each leaf node. Fig. (3.6)-(b) describes the process of splitting a parent node into two children nodes based on the optimal variable index j and value x_j^o. Some detailed information about Gini index based node splitting for soft training instances can be found in Algorithm 3.1.



(a) An Example of Soft Labeled Based Decision Tree for 2 Classes

(b) Gini Splitting Rule for Soft Label Based Decision Tree

Figure 3.6: *An simple example of soft label based decision tree for 2 classes*

Misclassification Error Estimation. After that the decision tree has been constructed, it can be used to classify other test instances. For a test instance $\mathbf{x}$, its prediction class label is determined according to the leaf node in which the instance falls. For a classical decision tree, each leaf node is associated with a hard class label. Considering $f(\mathbf{x})$ as the output of the classifier for $\mathbf{x}$, the 0-1 classification error for $\mathbf{x}$ is defined as below:

Algorithm 3.1 Gini rule based node splitting

Input: - Training instances in node λ_t;

Output: - Node split parameters, variable index j_o and value $x_{j_o}^k$.

1: ▶ Compute the Gini index $G(\lambda_t)$ for node λ_t according to Eq. (3.4).
2: ▶ Set $\triangle i_{max} = 0$. ▷ The maximum change of the impurity $\triangle i_{max}$
3: ▶ Search for best variable index j_o and best value $x_{j_o}^k$ through all variables of all instances.
4: **for** $j = 1$ **do** d ▷ d is the dimension of feature vector
5: **for** $k = 1$ **do** n_t ▷ n_t is the number training instance in node λ_t
6: ▶ Set $x_0 = x_j^k$, use the x_0 as the optimal value to split node λ_t into two parts.
7: ▶ Compute Gini index $G(\lambda_l)$, $G(\lambda_r)$ for the left and right nodes λ_l, λ_r according to Eq. (3.4) respectively;
8: ▶ Compute the change of the impurity $\triangle i$ from the parent node to the children nodes based on $\triangle i = G(\lambda_t) - p_l G(\lambda_l) - p_r G(\lambda_r)$, in which $p_l = \frac{n_l}{n_t}$, $p_r = \frac{n_r}{n_t}$, n_l and n_r are the number of instances falling in left and right nodes respectively.
9: **if** $\triangle i > \triangle i_{max}$ **then**
10: ▶ $\triangle i_{max} = \triangle i$; $x_{j_o}^k = x_0$, $j_o = j$;
11: **end if**
12: **end for**
13: **end for**
14: ▶ Split the node λ_t into left and right nodes based on the optimal index j_o and optimal value $x_{j_o}^k$.
15: ▶ Output optimal split parameters: optimal variable index j_o and value $x_{j_o}^k$.

$$L_{0-1}(y|f(x)) = \begin{cases} 1 & \text{if } \operatorname{sign}(f(\mathbf{x})) \neq y \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

where y is the ground truth class label of instance $\mathbf{x}$. However, the 0-1 classification error can not be used directly for the case of soft labeled decision tree because its output is a continuous value rather than a categorical class value. For binary classification problems, the cost-weighted misclassification error for the instance $\mathbf{x}$ can be expressed as below:

$$E_{Y|\mathbf{X}=\mathbf{x}} L_c(Y|f(\mathbf{x})) = \eta(\mathbf{x}) \mathrm{I}_{f(\mathbf{x})\leq 0.5} + (1-\eta(\mathbf{x})) \mathrm{I}_{f(\mathbf{x})>0.5} \tag{3.8}$$

where $\eta(\mathbf{x}) = P(Y = k|\mathbf{x})$ is considered as the posterior probability of class k given $\mathbf{x}$ and the output $f(\mathbf{x})$ is taken as an estimate of $\eta(\mathbf{x})$. $\mathrm{I}_{\text{condition}}$ stands for the indicator function (it is equal to 1 if the condition given is met, and 0 otherwise). The conjunction "$y = 1$ & $f(x) \leq 0.5$" describes the "false negatives" and "$y = 0$ & $f(x) > 0.5$" describes the "false positives".

Compared to the cost-weighted misclassification error, other error criteria, such as Expected Squared Error and Absolute Error:

$$E_{Y|\mathbf{X}=\mathbf{x}} L_s(Y|f(\mathbf{x})) = \eta(\mathbf{x})(1-f(\mathbf{x}))^2 + (1-\eta(\mathbf{x}))f^2(\mathbf{x}) \tag{3.9}$$

and

$$E_{Y|\mathbf{X}=\mathbf{x}} L_a(Y|f(\mathbf{x})) = \eta(\mathbf{x})(1-f(\mathbf{x})) + (1-\eta(\mathbf{x}))f(x) \tag{3.10}$$

also can be used for measuring the classification error. Fig. (3.7) shows the four different types of misclassification error for three different values of the posterior probability of class 1 (0.975, 0.75, 0.5). The black real line presents the L_{0-1} loss, whose values have been transferred by a logistic transform $p(\mathbf{x}) = \frac{e^{f(\mathbf{x})}}{e^{f(\mathbf{x})}+e^{-f(\mathbf{x})}}$. From the first subfigure of Fig. (3.7), we can see that the cost weighted error is close to 0-1 classification error when $\eta(\mathbf{x})$ has a high value. While the dotted green (absolute error) and dashed blue (squared error) lines decrease with the increase of $f(\mathbf{x})$ and reach their minimum value at $\eta(\mathbf{x})$, and then increase again. This variation trend can be seen more clearly in the second subfigure of the first row when $\eta(\mathbf{x})$ has a lower value. At the same time, the misclassification penalization for the cost weighted error has been reduced to $\eta(\mathbf{x})$ while the penalization for the right classification increases to $1-\eta(\mathbf{x})$. When $\eta(\mathbf{x}) = 0.5$, which means that the instance has an equal probability for class 1 and 2. In this case, the cost-weighted error criterion does not

work because it gives the same penalization on both false and right classifications. From the figure and analysis above, we can conclude that: cost weighted error works well if the $|\eta(\mathbf{x}) - 0.5|$ is large. The squared criterion and the absolute error are more suitable for the class probability estimation than for the classification problems. In the following, we will apply the cost-weighted misclassification error in our proposed boosting algorithm to handle the training data with soft class labels.

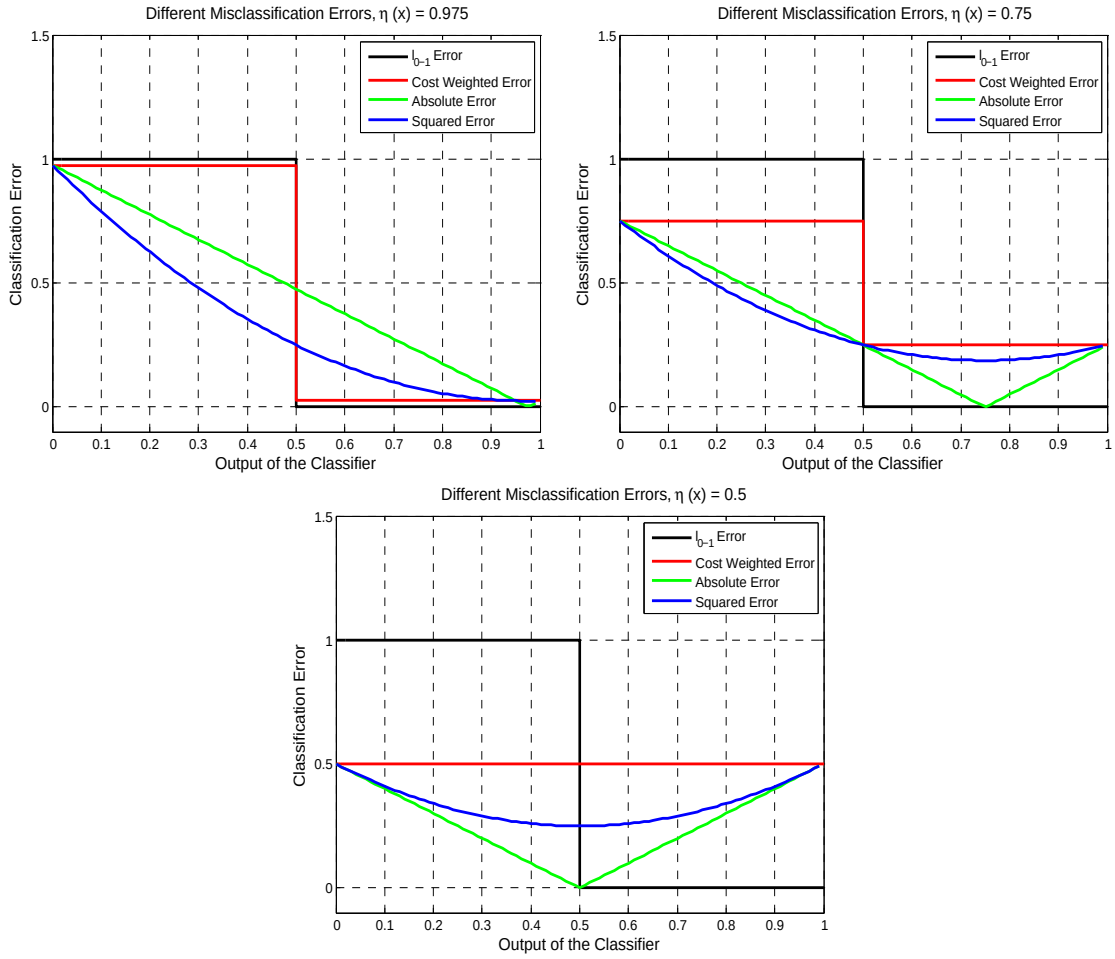


Figure 3.7: *Different misclassification error estimate*

We propose to boost the decision stumps trained using the soft labeled instances as follows. We use the classical boosting algorithm, except for the classification error of an instance $\mathbf{x}$ which is now quantified using Eq. (3.8) (with $f(x)$ being replaced by $h_t(\mathbf{x})$). The classification error attached to the weak learners can now be estimated by

$$\epsilon_t = \frac{1}{N} \sum_{i=1}^{N} w_i^t . \epsilon_{\mathbf{x}_i}.$$

Our soft label boosting approach is detailed in Alg. (3.2).

Algorithm 3.2 Soft label based Boosting algorithm

Input: - Training data $(\mathbf{x}_i, \boldsymbol{\pi}_i), i = 1, \cdots, N$, $\pi_{i,1} = P(y_i = 1|x_i)$;
- T, number of weak learners;
- Weak learners;

Output: - The final decision: $H(x_i) = \begin{cases} 1, & \text{if } \sum_{t=1}^{T} \alpha_t.h_t(x_i) \geq \frac{1}{2}\sum_{t=1}^{T}\alpha_t \\ 0, & \text{otherwise} \end{cases}$.

1: ▶ Initial instance weight vector: $w_i^1 = 2 * abs(\pi_{i,1} - 0.5)$.
2: **for** $t = 0$ **do** T ▷ T is the number of weak learners
3: ▶ Normalization of instance weights: $w_i^t = \frac{w_i^t}{\sum_{i=1}^{N} w_i^t}$;
4: ▶ Train a weak classifier h_t with distribution w_i^t , get back a hypothesis $h_t(.) : \mathcal{X} \rightarrow [0, 1]$;
5: ▶ Calculate the expected classification error for instance $\mathbf{x}_i$: $\epsilon_{\mathbf{x}_i} = \pi_{i,1}\mathrm{I}_{h_t(\mathbf{x}_i)\leq 0.5} + (1 - \pi_{i,1})\mathrm{I}_{h_t(\mathbf{x}_i)\geq 0.5}$
6: ▶ Calculate the expected error for the weak classifier h_t: $\varepsilon_t = \sum_{i=1}^{N} w_i^t.\epsilon_{\mathbf{x}_i}$
7: ▶ Set $\beta(t) = \frac{\varepsilon_t}{1-\varepsilon_t}$ and $\alpha_t = log\frac{1}{\beta(t)}$;
8: ▶ Update weight to each instance $\mathbf{x}_i$: $w_i^{t+1} = w_i^t.\beta(t)^{1-\epsilon_{\mathbf{x}_i}}$
9: **end for**
10: ▶ Output the final decision: $H(\mathbf{x}_i) = \begin{cases} 1, & \text{if } \sum_{t=1}^{T} \alpha_t.h_t(\mathbf{x}_i) \geq \frac{1}{2}\sum_{t=1}^{T}\alpha_t \\ 0, & \text{otherwise} \end{cases}$

Remarks

Our proposed soft label based boosting algorithm can take both hard and soft labeled instances as inputs. If all the instances are hard labeled, it boils down to the original AdaBoost algorithm. The soft label based decision tree can take both soft and hard training instances. Additionally, Eq. (3.8) degenerate to 0-1 classification error (which used in the classical AdaBoost algorithm) when the inputs are hard labeled instances.

Furthermore, an instance with a probabilistic label $\eta(\mathbf{x})$ close to the uniform probability distribution (0.5 in binary classification) will have a significant classification error even if it is well classified. As in Eq. (3.8), the classification error will be $\eta(\mathbf{x})$ (or $1-\eta(\mathbf{x})$) when $h_t(\mathbf{x}_i) \leq 0.5$ (or $h_t(\mathbf{x}_i) > 0.5$). This reflects the difficulty to learn a classifier from instances with high uncertain information. In order to let the classifier focus on the training instances with high confidence, we set the initial weights of the instances to $w_i^1 = 2 * abs(\pi_{i1} - 0.5)$. For a hard labeled instance, the initial weight is 1, while a 0 initial weight will be given to an instance who has equal classes probabilities.

3.3.3.2 Soft Class Label Estimation

The training data are expressed as $D = D^L + D^U$, where D^L and D^U are the labeled and unlabeled instances respectively. $D^L = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\} \subseteq \mathbf{X} \times \mathbf{Y}$, in which $\mathbf{x}_i \in \mathbb{R}^d$ stands for the d-dimensional feature of ith instance, $y_i \in \mathbf{Y}$ is the class label and n is the number of labeled instances. Here, we only focus on the binary classification problem, therefore $\mathbf{Y} = \{-1, +1\}$. The unlabeled instances are expressed as $D^U = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\} \subseteq \mathbf{X}$, in which m is the number of unlabeled instances. We also use $N = m + n$ to represent the number of all the training instances.

Gaussian Mixture Models. A GMM [141] is a parametric density estimation technique, in which the distribution of the data is supposed to be a mixture of g multivariate Gaussians:

$$p(\mathbf{x}|\mathbf{\Psi}) = \sum_{k=1}^{K} \theta_k \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma_k|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)\} \tag{3.11}$$

where, θ_k is the prior probability of the kth component of the model, μ_k a $d \times 1$ mean vector and Σ_k is a $d \times d$ covariance matrix to be estimated. The covariance matrices Σ_k can be of full or constrained to be diagonal, so that parameter estimation requires less training data and is faster. GMMs estimation using both diagonal and full covariance matrices has been investigated in this thesis. The detailed results are presented in Section 3.4.2.1.

GMMs are classically estimated from unlabeled instances: Maximum Likelihood Estimates (MLE) of the parameters are computed using the Expectation Maximization (EM) algorithm [142]. When labeled instances are available, they can be integrated in the parameter estimation process. The advantage of using such labeled instances is twofold. First, they can be used to compute (nontrivial) starting values for the parameters. Furthermore, they may guide the algorithm towards more accurate MLE parameters, since additional information is taken into account.

Probabilistic Class Label Estimation For a binary classification problem, we choose $K = 2$ to represent each class with a single Gaussian distribution. Therefore we have 5 groups of parameters to be estimated (since $\theta_1 + \theta_2 = 1$). Initial parameter values $\Psi^0 = \{\theta_k^0, \mu_k^0, \Sigma_k^0\}$ are obtained by computing the relative frequencies, the mean vector and the covariance matrices using the corresponding labeled instances. Then, MLE of the parameters are computed by applying the EM algorithm on all

the data. This makes it possible to obtain posterior probability estimates for the unlabeled data, that can then be used as soft labels in our boosting procedure.

3.4 Experimental Results

3.4.1 Benchmarks

In order to evaluate the effectiveness of the semi-supervised boosting algorithm, three different kinds of datasets have been used. The first datasets we used are the CVC [143] and INRIA [67] pedestrian datasets that have been widely used for training supervised classifiers. The CVC datasets have 3172 positive (pedestrian) and 15150 negative (non-pedestrian) instances while the INRIA datasets have 3542 positive and 4560 negative instances (randomly extracted from the negative images). Second, the binary classification datasets LibSVM repository [144] was also used in our experiments. Finally, we applied the approach on the real urban city sequences for pedestrian recognition.

In the three experiments, all the instances are randomly separated into labeled and unlabeled with a ratio value γ, which is defined as: $\gamma = m/(m+n)$, where, n and m represent the number of labeled and unlabeled instances respectively. For pedestrian recognition experiments, the HOG features [67] are computed for each instance first, then PCA is applied to reduce the feature dimensions. Then the PCA-HOG features are used to estimate soft class labels for unlabeled instances. And we use the original HOG features in classification process. We will introduce the experiments in detail in the following sections.

3.4.2 Classification on Classical Dataset

First, we test the performances of soft class labels estimated by GMM model on INRIA and CVC datasets. Then the soft class labels are used to evaluate the semi-supervised boosting algorithm.

3.4.2.1 Soft-Label Estimation

In order to evaluate the performance of the label estimation, we proceed as follows: each instance is assigned the class with the highest posterior probability. Then we compute three different error indicates: the false negative rate: F_neg=$\frac{m^+_{mis}}{m^+}$, the false positive rate: F_pos=$\frac{m^-_{mis}}{m^-}$ and the average false rate: F_ave=$\frac{m_{mis}}{m}$, where

$m^+$and m^- are the number of positive and negative instances, while m^+_{mis}, m^-_{mis} are the number of positive and negative instances that have been misclassified by the GMMs respectively. We use $m_{mis} = m^+_{mis} + m^-_{mis}$ to represent the number of all the misclassified instances. Details about the two different experiments are described in the following sections.

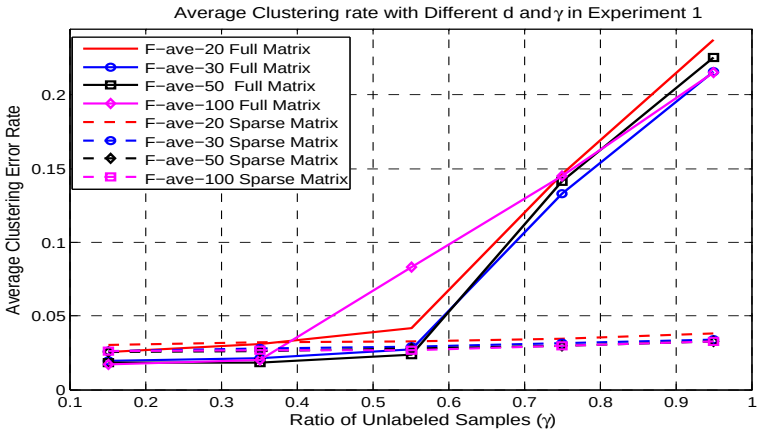
In Experiment 1, only the CVC dataset is used. The γ values range from 0.15 to 0.95. At each value, four different feature dimensions d have been tested. In Experiment 2, both the CVC and INRIA datasets are used in this experiment. We always consider the instances from the CVC dataset as the labeled data while the INRIA dataset is divided into unlabeled and labeled parts with a ratio γ.

In Experiment 1, all the instances come from the dataset while in Experiment 2 the instances comes from both the CVC and INRIA datasets. The experiment 2 is designed to test whether the GMMs can work if the labeled instances and the unlabeled instances come from different datasets. Fig. (3.8) and (3.9) describe three different error rates of the two experiments respectively. In the two figures, the solid and dotted lines represent the results with the full and diagonal covariance matrices in the GMM estimation process respectively.

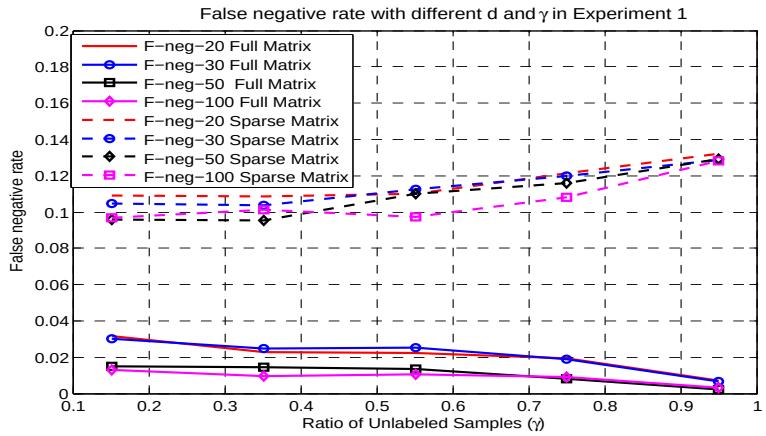
Analysis and Conclusions

Considering the average clustering error rate in Experiment 1 (Fig. (3.8)-(a)), the full matrix performs a little better than the diagonal matrix when there are enough labeled instances $\gamma < 0.5$, but it decreases rapidly when the amount of unlabeled instances increases. The performances of diagonal matrix keeps stable with the decrease of labeled instances. The reason is that more labeled instances are needed to accurately estimate the full matrices parameters; hence when the labeled instances are large enough, full matrices give better clustering results than using only diagonal ones. The performance of the full matrix drops rapidly with the decrease of the amount of labeled instances, while the performance of using the diagonal matrices is nearly unchanged. The average clustering error rate in Experiment 2 (Fig. (3.9)-(a)) has some differences compared to Experiment 1. The performances of the diagonal matrix highly depends on the amount of selected feature dimension when the number of unlabeled instances increases. The error rate obtained with full matrices only increases a little with the number of unlabeled instances.

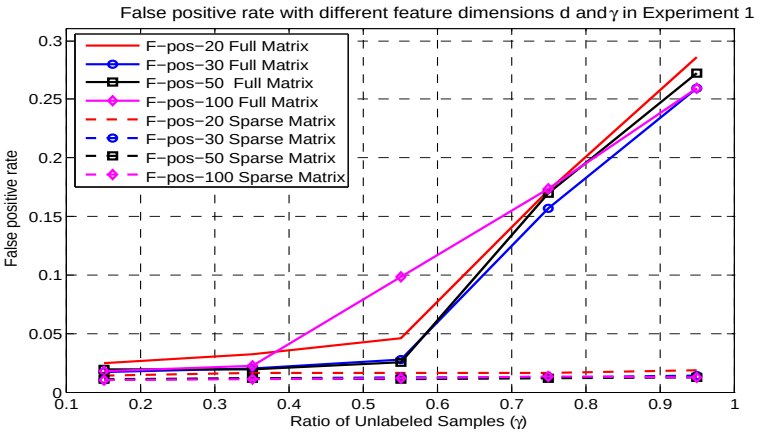
The second row of Fig. (3.8) and (3.9) give the false negative rates in Experiments 1 and 2. From these two figures, we can see that the accuracy when using full matrices (solid lines) is much higher than with diagonal matrices (dotted lines).



(a) *Average clustering error rate in experiment 1,*

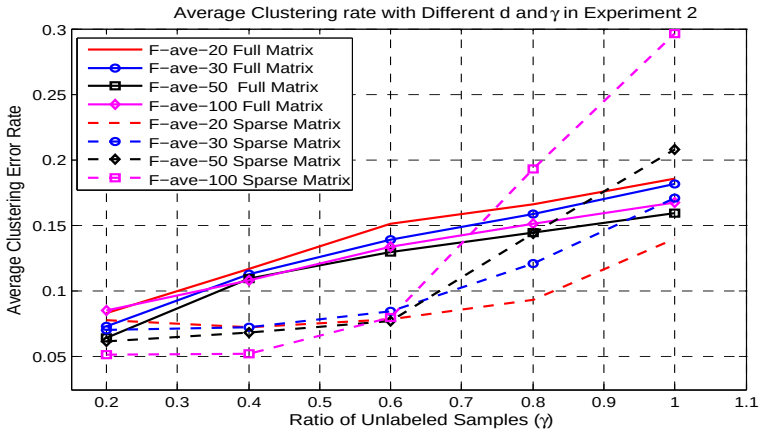


(b) *False negative rate in experiment 1*

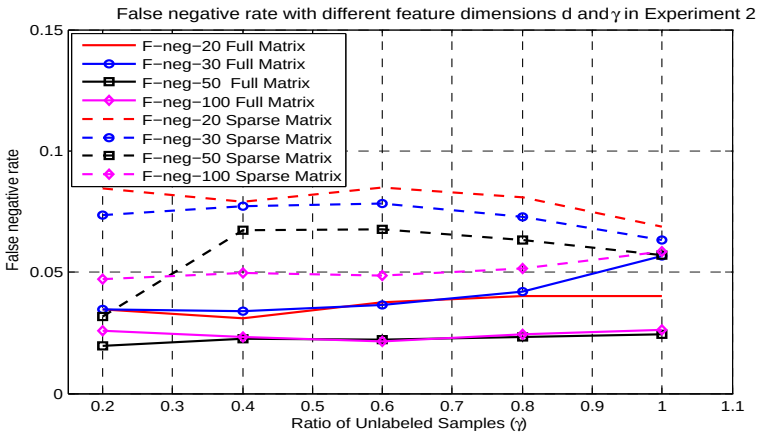


(c) *False positive rate in experiment 1*

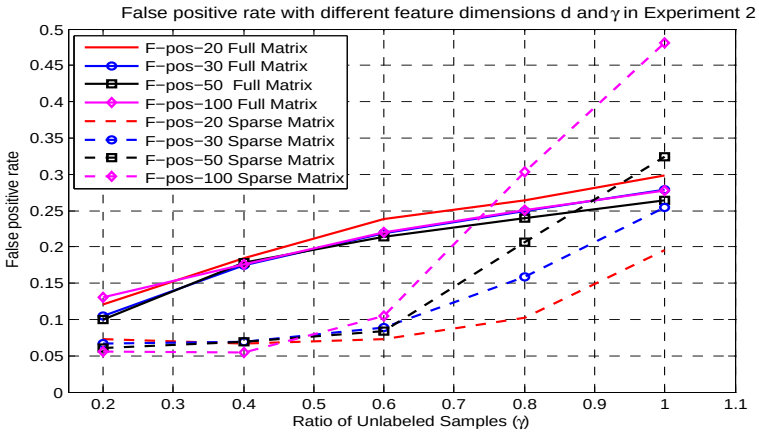
Figure 3.8: *Performances of GMMs based Probabilistic class labels estimation in experiment 1. In order to express succinctly, some abbreviations are used in figure legends: F-pos: false positive error rate; F-neg: false negative error rate; F-ave: false average error rate; {20,30,50,100} are the dimension of the PCA-HOG features.*



(a) *Average clustering error rate in experiment 2*



(b) *False negative rate in experiment 2*



(c) *False Positive Rate in Experiment 2*

Figure 3.9: *Performances of GMMs based probabilistic class labels estimation in experiment 2. In order to express succinctly, some abbreviations are used in figure legends: F-pos: false positive error rate; F-neg: false negative error rate; F-ave: false average error rate; {20,30,50,100} are the dimension of the PCA-HOG features.*

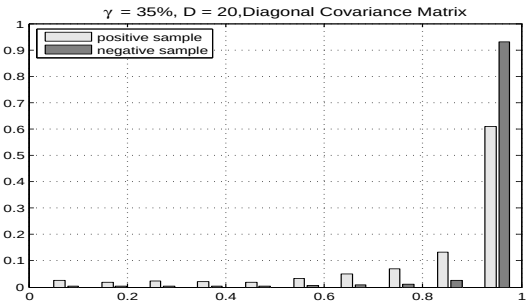
From the plots in the third row of Fig. (3.8) and (3.9), we can conclude that the diagonal matrices give much more stable results than the full matrices, especially when the labeled instances are not sufficient. Through the above analysis, we also find that full matrices are much more suitable for the distribution of the positive instances while the diagonal matrices are better for the ones of the negative instances.

This phenomenon can be explained by the fact that the features of the positive instances are likely concluded (because a pedestrian is included in each image instance), so a full covariance matrices will give a better description of these features. However, the negative instances are collected from various scenes that include different objects, such as building, trees, vehicles, road, sky etc. In this case, a general diagonal covariance matrix may be much better for the negative instances than for the positive ones.

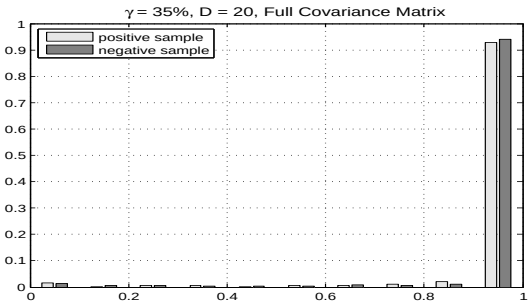
Usually, the number of negative instances is more than the number of positive instances for training pedestrian detectors. For example in CVC dataset, the number of positive is 3172 while the negative is 15150. The performances of soft label estimation on the negative instances are especially important. Therefore diagonal matrices with a lower feature dimension are chosen for the following experiments because they give better results than others.

The histograms in Fig. (3.10) to Fig. (3.13) show the distribution of estimated posterior probabilities for actual class of the instances (i.e., estimates of $P(\omega_1|\mathbf{x})$ for positive instances and $P(\omega_2|\mathbf{x})$ for negative instances), where light bars and dark bars represent the positive and negative classes respectively. In Experiment 1, we show results for rates of unlabeled instances of $\gamma = 0.35$ and $\gamma = 0.95$, with four different numbers of features kept for PCA (Fig. (3.10) and Fig. (3.11)). In Experiment 2, we consider $\gamma = 0.6$ and $\gamma = 1.0$, again for four different numbers of features kept for PCA (Fig. (3.12 and Fig. (3.13)).

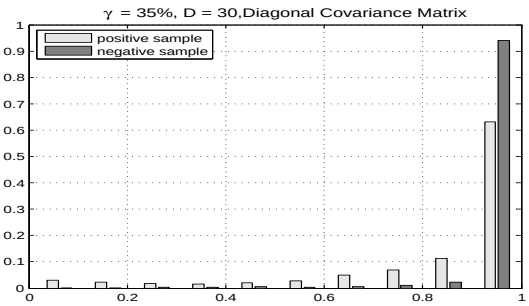
In these figures, we can found that most of the positive and negative instances have been assigned to a high class probability (rightmost column in each sub-figure), which means that our GMMs based approach is effective to estimate the soft class labels for most of the unlabeled instances. From the Fig. (3.10) and Fig. (3.11) of experiment 1, diagonal matrices give better results for negative instances than full matrices; this superiority is more obvious when $\gamma = 0.95$. Full matrices give to about 20% negative instances a high probability (more than 0.9) to the wrong class. For the positive instances, the full matrix performs better, which can be seen clearly in the rightmost light gray column in each sub-figures. However, only a small ratio of positive instances have been given a high probability to negative class for



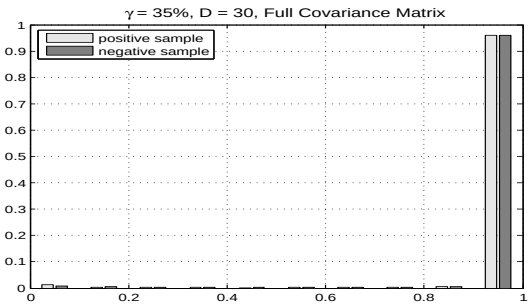
(a) *Diagonal Matrix in Experiment 1*



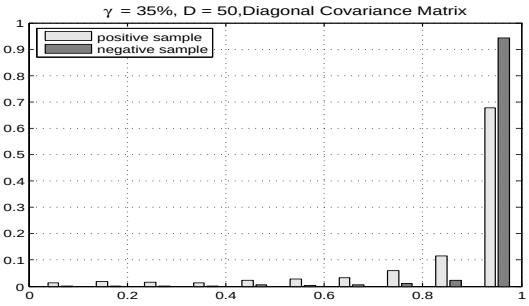
(b) *Full Matrix in Experiment 1*



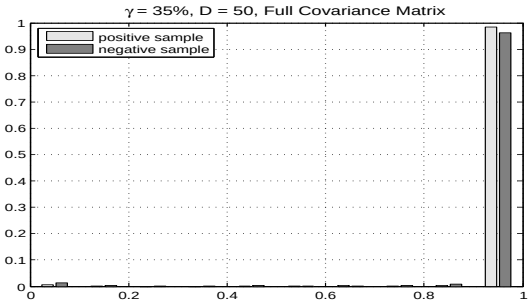
(c) *Diagonal Matrix in Experiment 1*



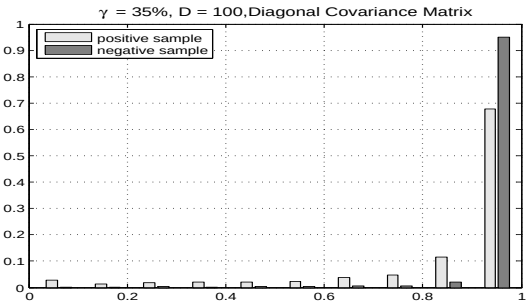
(d) *Full Matrix in Experiment 1*



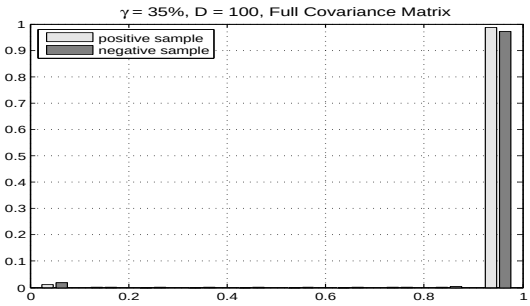
(e) *Diagonal Matrix in Experiment 1*



(f) *Full Matrix in Experiment 1*

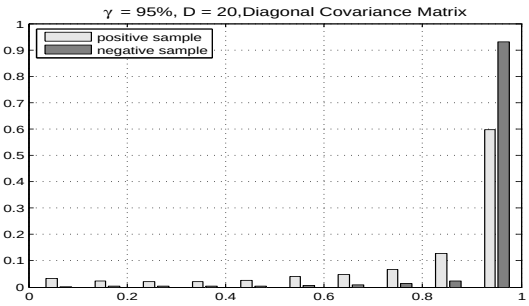


(g) *Diagonal Matrix in Experiment 1*

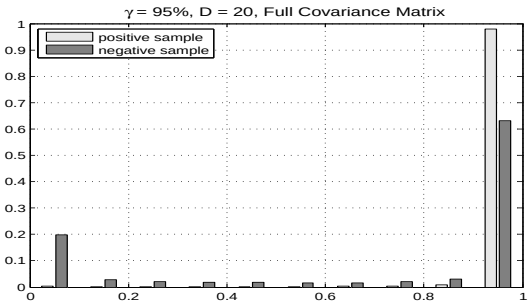


(h) *Full Matrix in Experiment 1*

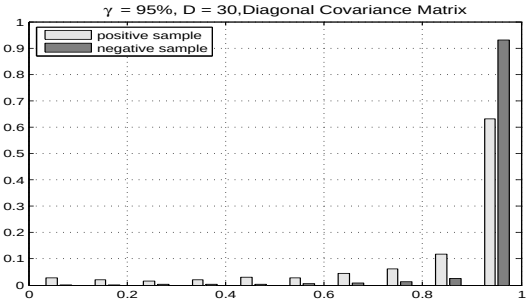
Figure 3.10: *Distribution of estimated probabilistic class labels in experiment 1.*



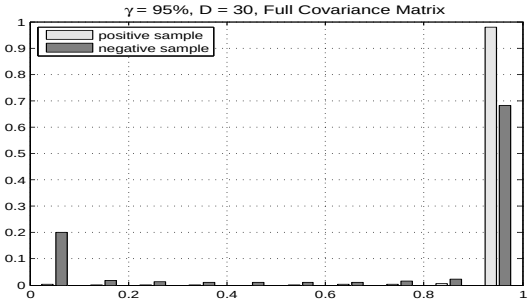
(a) *Diagonal Matrix in Experiment 1*



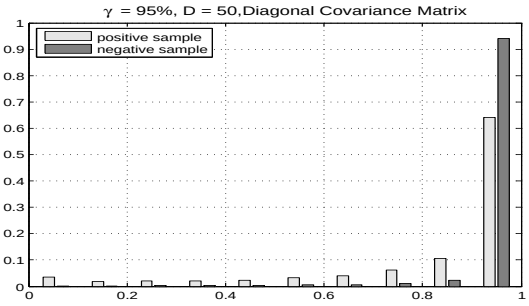
(b) *Full Matrix in Experiment 1*



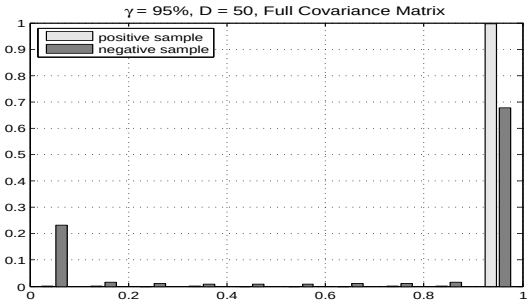
(c) *Diagonal Matrix in Experiment 1*



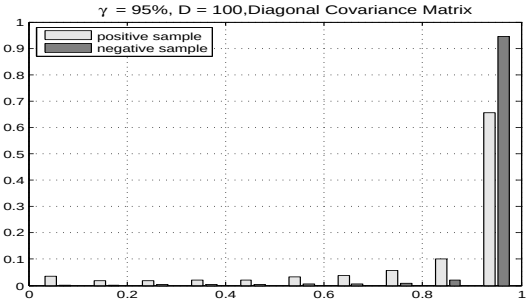
(d) *Full Matrix in Experiment 1*



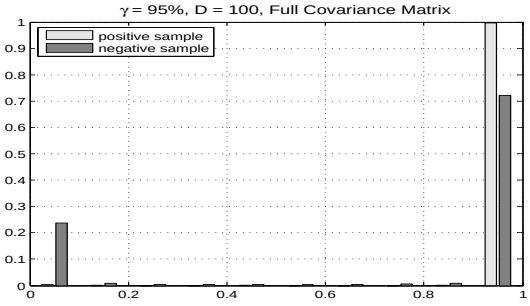
(e) *Diagonal Matrix in Experiment 1*



(f) *Full Matrix in Experiment 1*

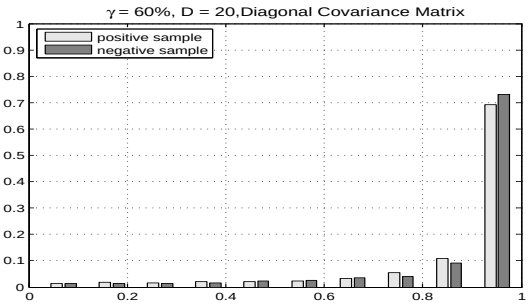


(g) *Diagonal Matrix in Experiment 1*

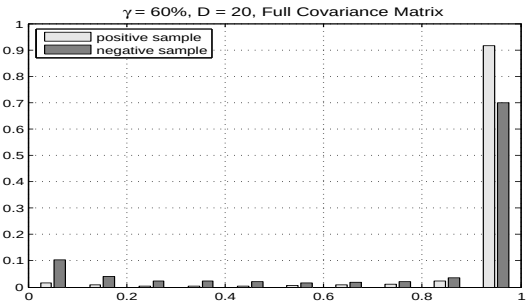


(h) *Full Matrix in Experiment 1*

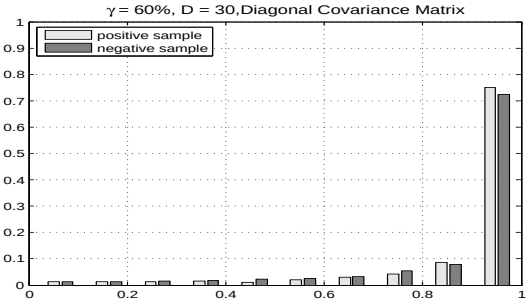
Figure 3.11: *Distribution of estimated probabilistic class labels in experiment 1.*



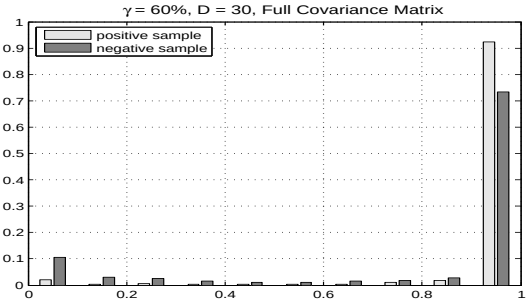
(a) *Diagonal Matrix in Experiment 2*



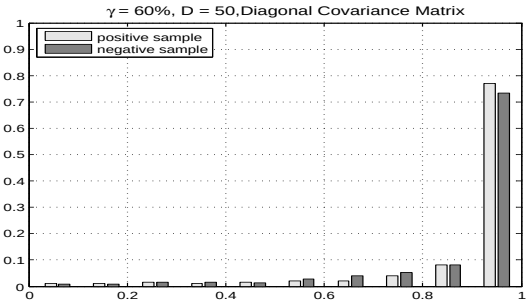
(b) *Full Matrix in Experiment 2*



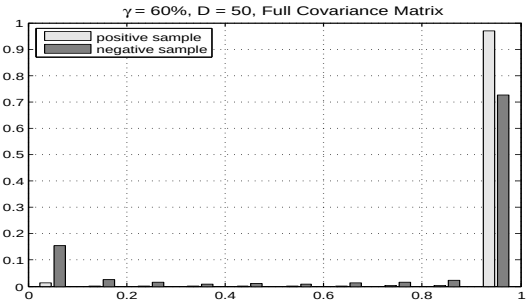
(c) *Diagonal Matrix in Experiment 2*



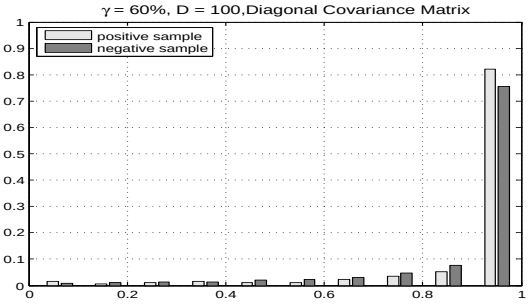
(d) *Full Matrix in Experiment 2*



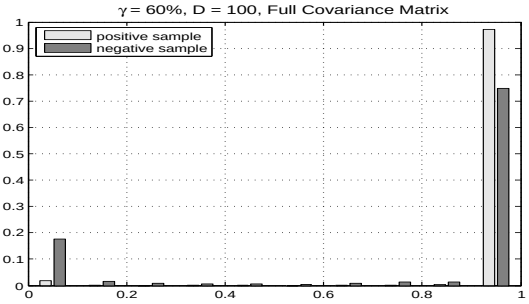
(e) *Diagonal Matrix in Experiment 2*



(f) *Full Matrix in Experiment 2*

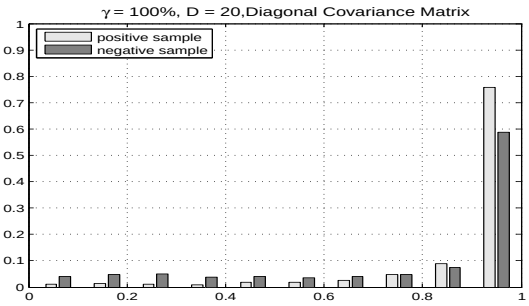


(g) *Diagonal Matrix in Experiment 2*

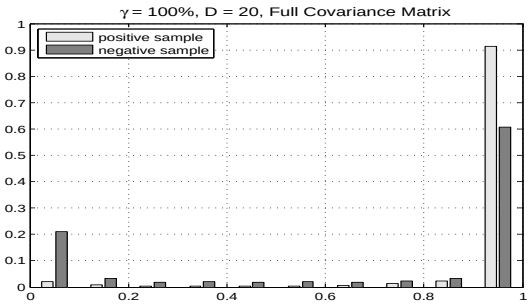


(h) *Full Matrix in Experiment 2*

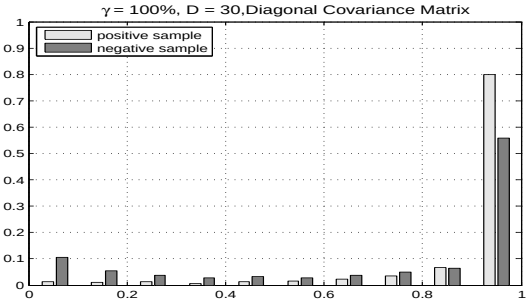
Figure 3.12: *Distribution of estimated probabilistic class labels in experiment 2.*



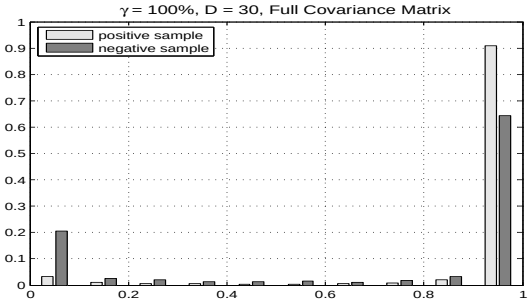
(a) *Diagonal Matrix in Experiment 2*



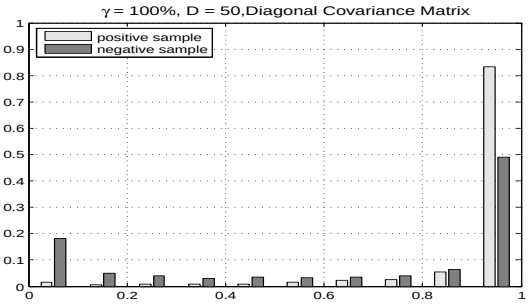
(b) *Full Matrix in Experiment 2*



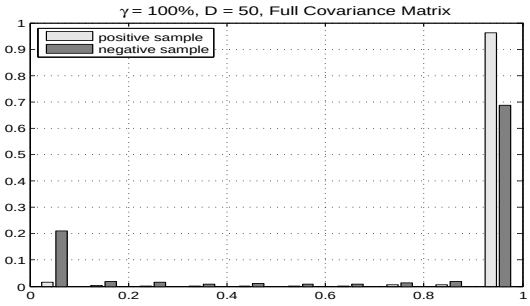
(c) *Diagonal Matrix in Experiment 2*



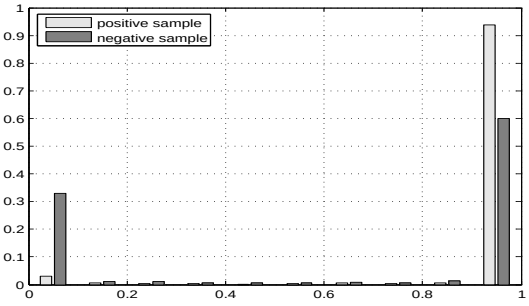
(d) *Full Matrix in Experiment 2*



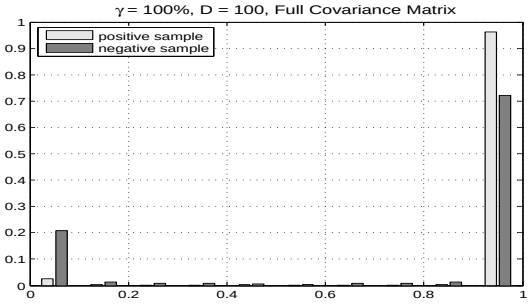
(e) *Diagonal Matrix in Experiment 2*



(f) *Full Matrix in Experiment 2*



(g) *Diagonal Matrix in Experiment 2*



(h) *Full Matrix in Experiment 2*

Figure 3.13: *Distribution of estimated probabilistic class labels in experiment 2.*

both diagonal and full matrices. The clustering error of full matrices for positive instances is smaller than that obtained with diagonal matrices.

The Fig. (3.12) and Fig. (3.13) give estimation results in the Experiment 2. Compared to Experiment 1, both the full and diagonal matrices give inferior performances for the negative instances because the negative instances in CVC dataset and INRIA dataset are quite different. So it is not easy to use the labeled instances in CVC to estimate the unlabeled instances in INRIA. Our proposed method also gives a high probability to the right class (the columns at the rightmost of each sub-figure). Similar with Experiment 1, full matrices perform better for the positive instances and the diagonal matrices give superior results for negative instances in experiment 2. The only difference is that the performances of diagonal matrices for negative samples decrease with the increase of the PCA-HOG features' dimensions in experiment 2. It gives even worse results than full matrices when features' dimensions equal 100. However, the performances of the diagonal matrix for positive instances improve with the increase of the feature dimension.

Based on the analysis of the two experiments above, we can conclude that the diagonal matrices give relatively better results than the full matrices when the feature dimensions are less than 50. Because it makes less error on the negative instances and usually we have more negative instances in the training data. Finally, the diagonal matrices give less average errors for all the instances (see in Fig. (3.8).

3.4.2.2 Pedestrian Classification

We took the INRIA pedestrian dataset to evaluate the algorithm for pedestrian recognition. First, we randomly selected 75% of the whole data as the training data and the rest was kept for testing. In our experiments, we let the amount of labeled instances vary to test the robustness of our proposed approach. We followed the method in [67] to represent each instance by a 3780-dimensional HOG feature. Then a PCA has been applied to the resulting vector to obtain a 20-dimensional feature for each. Guided by the labeled instances, the probabilistic class labels for unlabeled training instances are estimated by GMMs using the EM algorithm. Here, 20 features were kept to estimate the soft class labels. In the boosting algorithm, the original HOG features are used for recognition. The AdaBoost algorithm trained with few labeled instances is taken as the baseline. Four different kinds of classifiers have been designed in this experiment: classifier 1 is a classical AdaBoost classifier trained using only few labeled instances; classifier 2 is a GMMs classifier using both the labeled and unlabeled instances; classifier 3 is the proposed semi-supervised

boosting classifier trained using labeled instances and all the unlabeled instances with soft class label; classifier 4 is also the proposed semi-supervised boosting classifier trained using all labeled and some selected unlabeled instances who have reliable soft class labels. Classifier 1, 3 and 4 use stump decision trees as the weak learners. Hard label decision trees are used in Classifier 1, while soft decision trees are applied in both Classifier 3 and 4.

All the experiments have been repeated 5 times and we calculated the average recognition rate to draw the Fig. (3.14). From this figure, we can see that the recognition rate of Classifier 1 (blue line) increases with the increase of labeled training instances, however, the recognition rate of Classifier 2 does not grow with the increase of the labeled. The green line represents the performance of Classifier 3. Although it increases with the growth of labeled, it gives worse performances than Classifier 1 which has been trained using only few labeled ones. Through this experiment we found that the weak performances of Classifier 3 are mainly due to the training instances who have been assigned with wrong probabilistic class labels, which have a great influence on the recognition rate of Classifier 3.

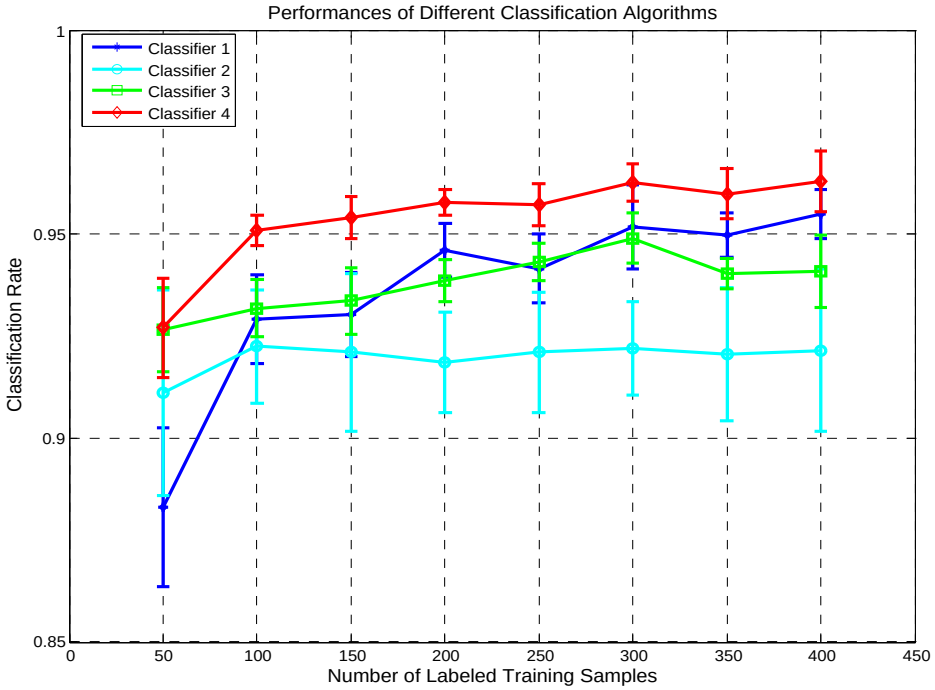


Figure 3.14: *Average recognition rates with variances of four different classifiers. We choose* $T = 500$ *for all boosting classifiers in this experiment.*

In order to reduce this negative effect, we should remove these unlabeled instances with these wrong probabilistic labels. Therefore, we considered a soft label

to be reliable if its predicted class label using Classifier 1 and 2 are consistent. Classifier 4 is trained using all the labeled instances and the instances with reliable probabilistic class labels. Although our strategy is not able to get rid of all the wrong probabilistic instances from the training data, it still improves the recognition rate significantly. This improvement can be seen from the red line of Fig. (3.14). Comparing the red and blue lines in Fig. (3.14), we can find that our proposed boosting algorithm with additional soft labeled instances gives a better recognition rate (with a small variance)than the AdaBoost trained with only labeled instances.

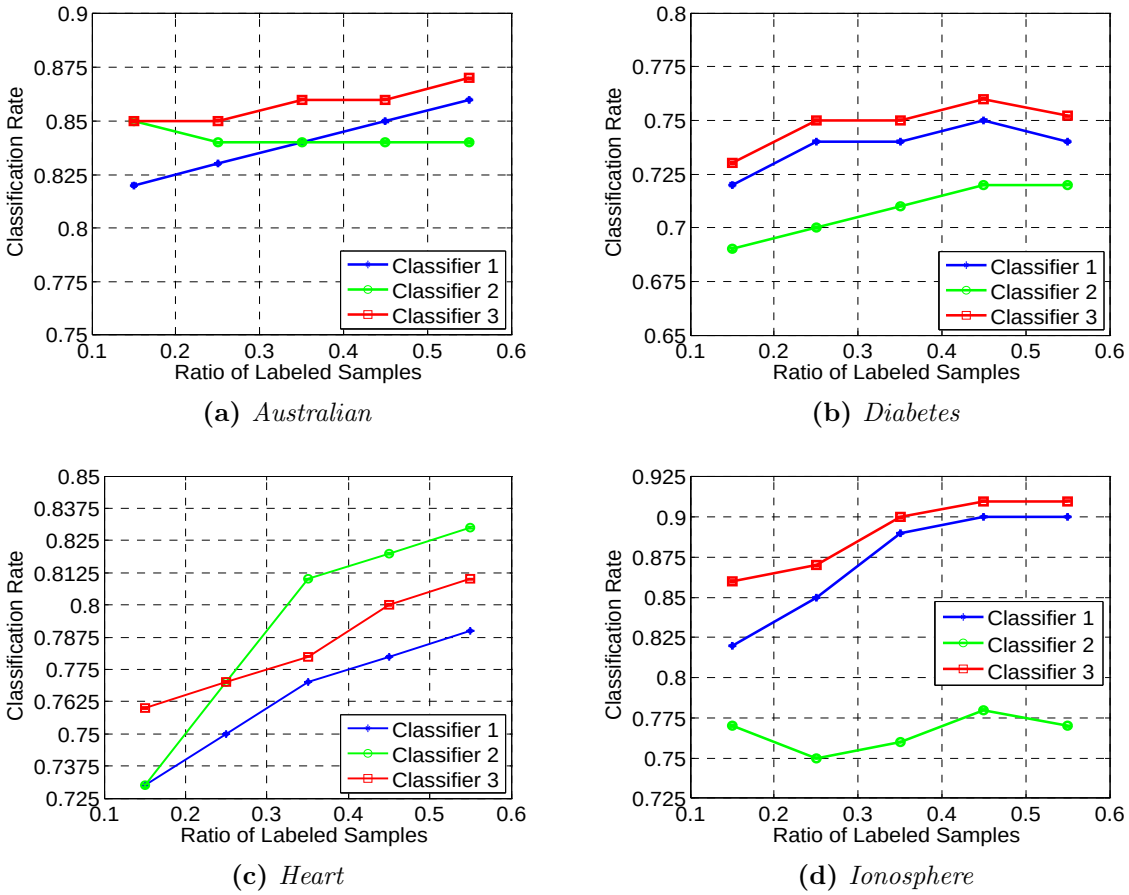
3.4.2.3 Data Classification

Four different dataset have been chosen to test our algorithm. Table 3.4.2.3 gives a general description of the data.

Dataset	Size of the data	Attribute dimension
Australian	690	14
Diabetes	768	8
Heart	270	13
Ionosphere	351	34

As in the previous section, all the data have been randomly divided into training and testing sets with a proportion 3:1. In the training data, a small part of the training samples are selected as labeled samples and the rest are considered as unlabeled samples. In this experiment, only three classifiers are designed: 1) classifier 1 is a classical AdaBoost classifier trained using only few labeled samples, 2) classifier 2 is a GMM classifier using both the labeled and unlabeled samples, 3) classifier 3 is the proposed semi-supervised boosting classifier trained using all the labeled and some selected unlabeled samples and the selection strategy is the same as in subsection 3.4.2.

For each dataset, the ratio of labeled samples changed from 0.15 to 0.55. Fig. (3.15), show the classification rate of the four different datasets in each sub-figure respectively. From the figure we can easily find that our proposed semi-supervised boosting algorithm (red line) has a higher classification rate than classical AdaBoost algorithm (blue line) most of the time. The additional unlabeled samples help to improve the classification rate in our semi-supervised boosting algorithm.



(a) *Australian* **(b)** *Diabetes*

(c) *Heart* **(d)** *Ionosphere*

Figure 3.15: *Classification rate of four different Dataset. All the experiments have been repeated 50 times. We choose* $T = 100$ *for all boosting classifiers in this experiment.*

3.4.3 Pedestrian Recognition in Real Urban City Sequences

Here, we apply our classification strategy to the bounding boxes obtained as explained in Section 2.4. The flow chart of our pedestrian recognition is reminded in Fig. (3.16).

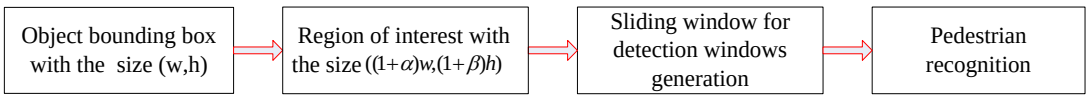
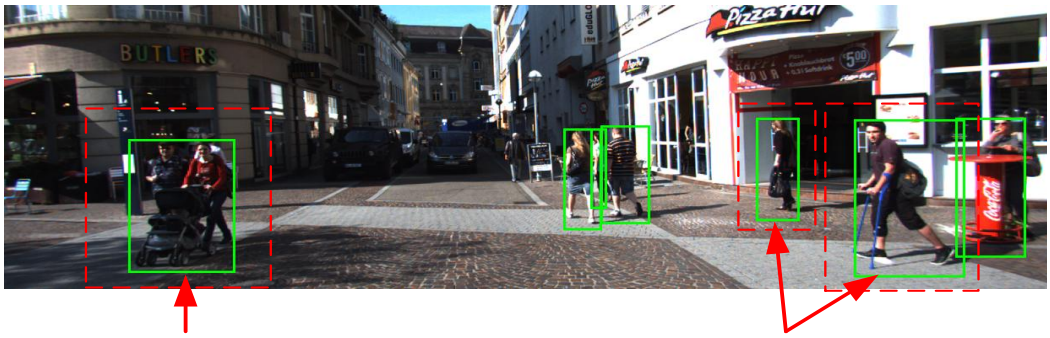


Figure 3.16: *Flow chart of pedestrian recognition*

3.4.3.1 Detection Windows Generation

As in [8], the HOG features are usually extracted in a 64*128 standard detection window with a 4 pixels margin in each side. In the training and testing procedures, an arbitrary size of instance is resized to a 64*128 standard window. Since the classifier is sensitive to the location of detection window, we can not obtain optimal recognition results if we extract the HOG features from the objects bounding boxes directly. The reasons for that are listed as below: first, the generated bounding boxes may be not accurate. Partial and redundant detections (as in Fig. (3.17)-(1)) may happen in the detection step. Second, more than one pedestrian may be included in a single bounding box. Third, the positive training instances in public dataset also include some background surrounding the pedestrians, which can be seen clearly in Fig. (3.17)-(3).



(3) Positive Training Samples in INRIA Dataset

Figure 3.17: *Detection window generation*

In order to reduce the influences of the third reason, an appropriate border will be added to each bounding box with ratios α and β in horizontal and vertical directions respectively. The red dashed boxes in first row of Fig. (3.17) are the enlarged bounding boxes with the size of $((1+2\alpha)w, (1+2\beta)h)$. In our experiments we choose $\alpha = 0.25$ and $\beta = 0.15$. Additionally, a sliding window strategy is

employed in each bounding box to improve the recognition rate. Moreover, we have two different sliding window strategies according to the type of bounding boxes: single pedestrian or grouped ones. We use a ratio $\rho = \frac{w}{h}$ (w and h are the width and height of the bounding box) to define the type of the bounding box. As the ratio ρ of training instances (64×128 in [67] or 48×96 in [107]), we choose 0.5 as a threshold. If $\rho \leq 0.5$, the bounding box is considered as a single pedestrian, otherwise, it is considered as a grouped bounding box. Sliding window strategies for single and grouped pedestrians bounding box are described below.

For a single pedestrian bounding box, it is resized to a image of size 80*144 first. Then the sliding detection windows (size 72*132) are obtained from this resized image for HOG features extraction. The starting point is $(1, 1)$, then detection windows slide in both directions of the resized image with a same sliding stride s. After the sliding window process, we have $n_d = \llcorner \frac{144-132}{s} + 1 \lrcorner * \llcorner \frac{80-72}{s} + 1 \lrcorner$ detection windows in each bounding box, where $\llcorner . \lrcorner$ is the nearest smallest integer value.

For a grouped pedestrians bounding box, the sliding window process has a little difference in the horizontal direction. First, the bounding box is resized with a ratio η, where $\eta = \frac{144}{h}$. Then the sliding detection windows (size 72*132) are obtained from this resized image for HOG features extraction. Different from the single pedestrian bounding box, the slide strides in horizontal and vertical directions are s_x and s_y respectively. The number of detection windows n_d for a grouped pedestrians bounding box is computed as $n_d = \llcorner \frac{144-132}{s_y} + 1 \lrcorner * \llcorner \frac{w*\eta-72}{s_x} + 1 \lrcorner$.

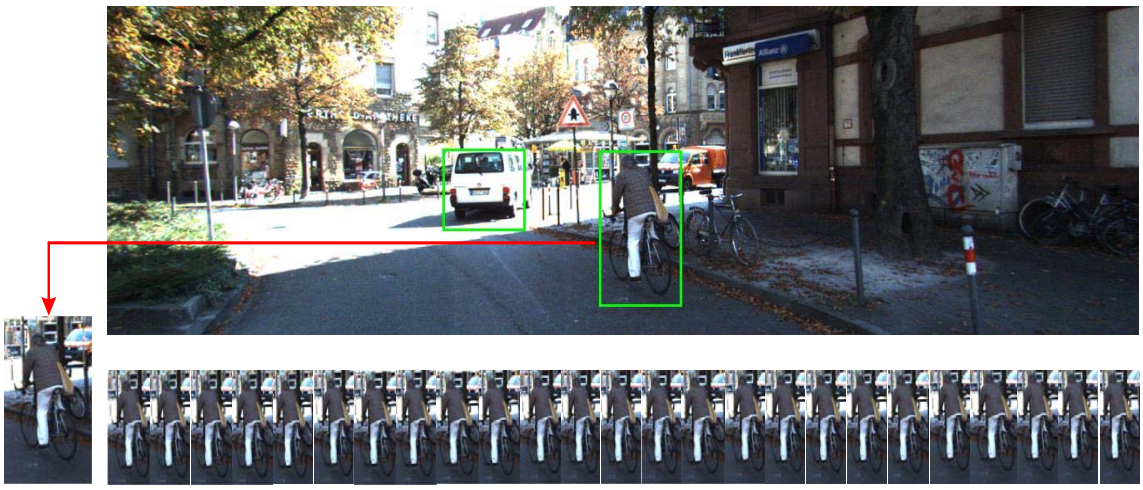


Figure 3.18: *Pedestrian recognition*

Fig. (3.18) gives the details of the pedestrian recognition process. The first row is a bounding box result of the moving object detection. The second row shows 25 detections windows generated from one of the bounding boxes using our sliding

Parameters	Description	Value
α	additional border ratio in horizontal direction	0.15
β	additional border ratio in vertical direction	0.25
ρ	ratio threshold for single or grouped bounding box	0.5
s	sliding stride for single bounding box	2 pixels
s_x	sliding stride in horizontal direction for grouped bounding box	4 pixels
s_y	sliding stride in vertical direction for grouped bounding box	2 pixels

Table 3.1: *Parameters used in our experiments*

window strategy. Tab. (3.1) gives the main parameters used in our experiments.

3.4.3.2 Pedestrian Recognition

Three different classifiers have been designed in this experiment for the pedestrian recognition task.

- Classifier 1 is a classical AdaBoost classifier trained using only 200 labeled instances.
- Classifier 2 is the classical AdaBoost classifier trained using all the 6076 labeled instances.
- Classifier 3 is the proposed semi-supervised boosting classifier trained using 200 labeled instances and some reliable soft labeled instances.

All three classifiers use 500 stumps as weak learners. The table in Fig. (3.18) shows the recognition results of the three classifiers in one frame, where the value 1 (or 2) in the table means that this instance is a pedestrian (or a non-pedestrian). From Fig. (3.18), we can see that the location of the detection window has a big influence on the recognition result. Our sliding window strategy can reduce this influence. A detection bounding box is considered as non-pedestrian if and only if all the detection windows are detected as non-pedestrian. In other words, one detection bounding box is considered to be a pedestrian as long as one detection window has been recognized as a pedestrian.

3.4.3.3 Experimental Results

We have tested our pedestrian recognition classifiers in different KITTI urban city sequences. Three detected bounding boxes are considered here, non-occlusion pedes-

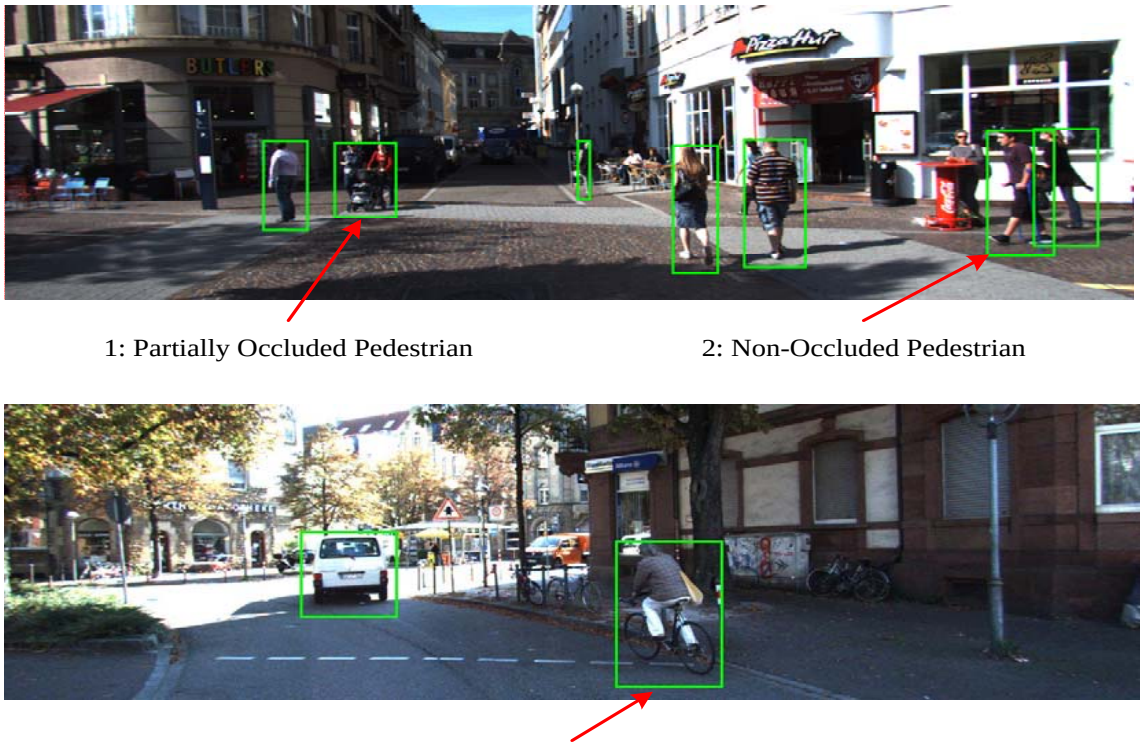


Figure 3.19: *Three different types of pedestrian in urban city*

trian, partially occluded pedestrian and cyclist. Some examples of bounding boxes are shown in Fig. (3.19).

In the first row of Fig. (3.19), the couple behind the baby carriage appears in about 90 frames of this sequence. However, they are correctly detected in 68 frames only by the moving object detection algorithm because they are far away from the camera. The man crossing the road with a crutch appears in about 149 frames of this sequence, while 144 bounding boxes have been generated for him in the detection step. The cyclist in the second row has been detected in 154 frames. At the same time, a van is also detected in 57 frames. Tab. (3.2) and (3.3) show the recognition results of the two sequences respectively. We only consider recognition results of the non-occluded people and the partially occluded couple in the first sequence, so we do not have the true negative objects in Tab. (3.2). In the second sequence, a van is taken as the true negative detections.

A bounding box is recorded as a true positive as long as one detection window is verified as a pedestrian. From Tab. (3.2), we can see that classifier 2 gives the best performances for both objects. Compared to the classifier 1, the classifier 3 performs a little better. Tab. (3.3) displays the recognition results in the second sequence.

Different Classifiers	Non-Occluded Pedestrian		Occluded Pedestrian	
	True Positive	False Negative	True Positive	False Negative
Classifier 1	116	28	54	14
Classifier 2	135	9	59	9
Classifier 3	122	22	55	13

Table 3.2: *Pedestrian recognition results in sequence 1*

Classifiers	True Positive	False Negative	True Negative	False Positive
Classifiers 1	143	4	53	4
Classifiers 2	147	0	57	0
Classifiers 3	147	0	57	0

Table 3.3: *Pedestrian recognition results in sequence 2*

In this sequence, 204 bounding boxes (147 cyclist and 57 van) are considered as the inputs for our recognition step. From the table, we can see that our approach performs as well as classifier 2. Compared to classifier 1, our classifier 3 gives a great improvement, with more true detections and less false detections.

Fig. (3.20) - Fig. (3.22) give the recognition results in detail. As mentioned above, 25 detection windows are generated for the single pedestrian bounding box by using the sliding window strategy. Fig. (3.20)-(a) records the number of the positive detection windows (n_p) which have been recognized as pedestrian by the three classifiers in each frame. For a fair comparison, the parameter n_p for grouped pedestrians bounding box should be multiplied by a coefficient $25/n_d$ (n_d is the number of detection window in this bounding box). Obviously, the more positive detection windows were made, the better the classifier is. In Fig. (3.20)-(a), the X-axis represents the frame number and the y-axis gives the number of positive detection windows n_p. The point with a 0 value at the y-axis means that the bounding box is considered as a non-pedestrian by the classifier in this frame. The diagrams of Fig. (3.20)-(b) record the percentage of frames (φ) whose n_p (positive detection window number) is over a given value τ. In other words, φ can also be considered as the right recognition rate if the τ is taken as the threshold of having pedestrian or non-pedestrian in the bounding box.

From these figures, we can generally conclude that our approach (classifier 3) gives better results than classifier 1, especially for the cyclist, which gives about 15% improvement on average. For the partially occluded pedestrians, the recognition rate of the proposed approach increases a lot as well when the τ value is not

large. Although the recognition rates drop rapidly when τ value is above 4, classifier 3 still has a higher recognition rate than classifier 1. In Fig. (3.20), we find that all the classifiers are inferior to recognize the non-occlusion pedestrians here. That is because the pedestrian is always sideways in this sequence and the sideways pedestrian is hard to be recognized. Even in this case, our approach performs better than classifier 1.

The pedestrian recognition is realized on a standard laptop (Intel i7, 4 Core) with Matlab R2014a processing environment. The approach can achieve 0.5 seconds per frame because only the ROIs generated from the MOD step are considered. Compared to the classical sliding window technique in the whole image (18s per frame), our approach has great improvement. Compared to the classical AdaBoost, the proposed semi-supervised boosting algorithm requires more time for training because of the increase of the unlabeled instances. However, the time increase in training process is sustainable because the classifiers is trained offline in advance. Furthermore, the time spent on recognition part of our proposed algorithm is similar with the classical AdaBoost algorithm.

3.5 Conclusion

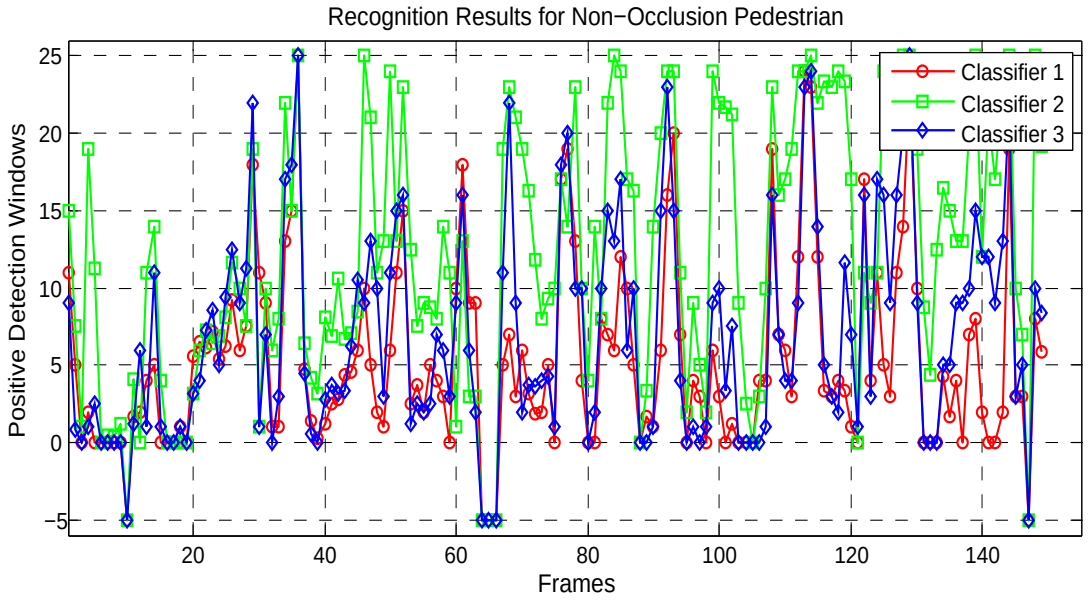
In this chapter, we proposed a semi-supervised boosting algorithm that uses few labeled instances and a large number of unlabeled instances for pedestrian recognition.

We have proposed a boosting framework which takes both hard and soft labeled instances together in training process. Compared to the classical boosting algorithm, several modifications appear in our method: first, soft labeled based decision trees have been employed as weak classifiers to replace the classical decision trees. Second, the cost-weighted classification error is applied to measure the classification rate of each weak classifier and this error is also used to update the distribution weight of each training instance. These modifications can properly handle the soft labeled instances in the training process.

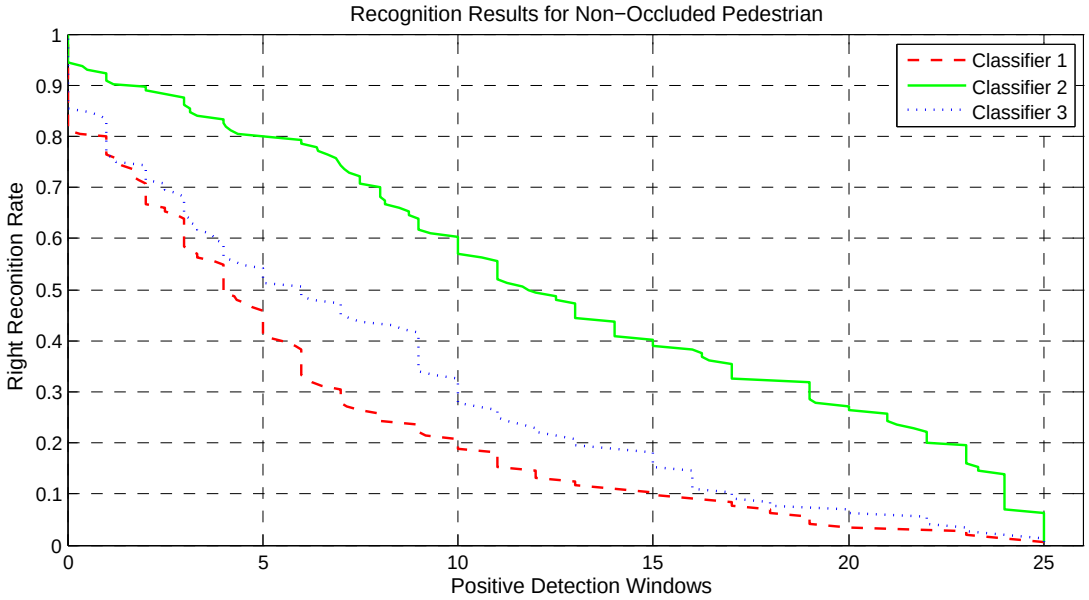
Compared to other semi-supervised learning methods (such as, self-training) which assign a pseudo hard label for each unlabeled instance, we give soft class labels to the unlabeled instances in our approach. The advantages of soft class labels are twofold: first, it can represent the uncertainty of instance such as partial object or redundant background in the instance; second, it can represent the uncertainty in the GMM based soft class label estimation process.

We also showed that PCA-HOG features dimension reduction has a little influence on the results of soft class labels estimation. This interesting phenomenon shows that it is possible to estimate soft class labels for pedestrian instances with fewer features than using standard high dimensional HOG. Additionally, GMMs can give right class labels for most of the unlabeled instances based on the reduced PCA-HOG features, especially for the positives instance in the pedestrian recognition experiments.

Finally, our proposed approach has been tested on several public datasets and the experimental results show that it can improve both the classification and pedestrian recognition rates by adding soft labeled instances. In the real pedestrian recognition experiments, non-occluded, partly occluded and cyclists have been successfully recognized respectively. Our proposed algorithm gives better recognition rates than classifier trained based on only labeled instances in all the three cases and this indicates that our approach has a good adaptation to various environments.

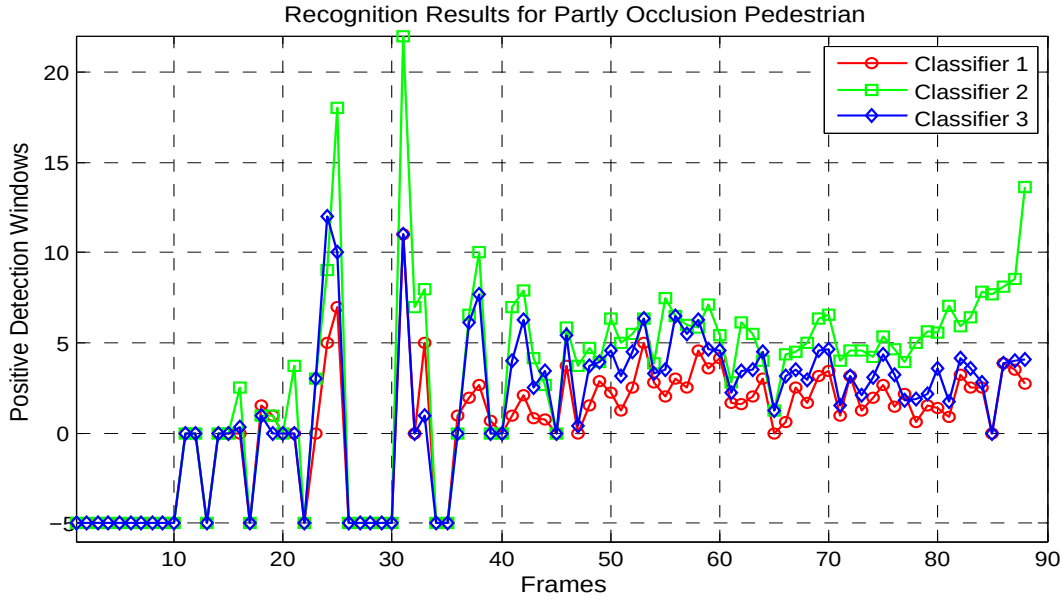


(a) *n_p for non-occluded pedestrian, $n_p = -5$ means that the object has not been detected in this frame*

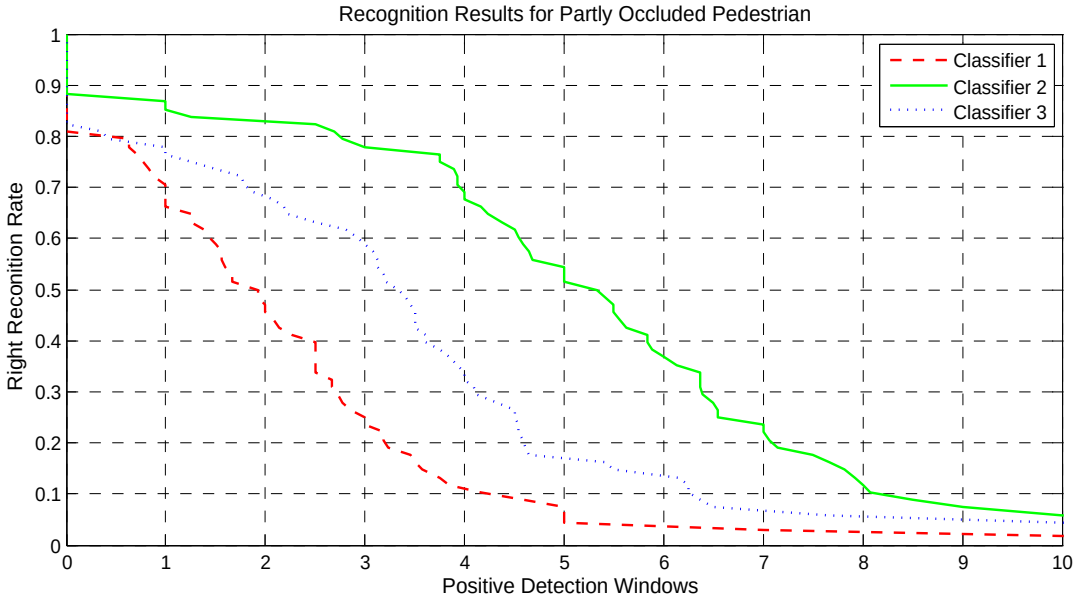


(b) *Recognition rate for non-occluded pedestrian*

Figure 3.20: *Recognition performance of three classifiers for non-occluded pedestrian. The point with values of -5 at y-axis means that the bounding box of the object is not detected in this frame.*

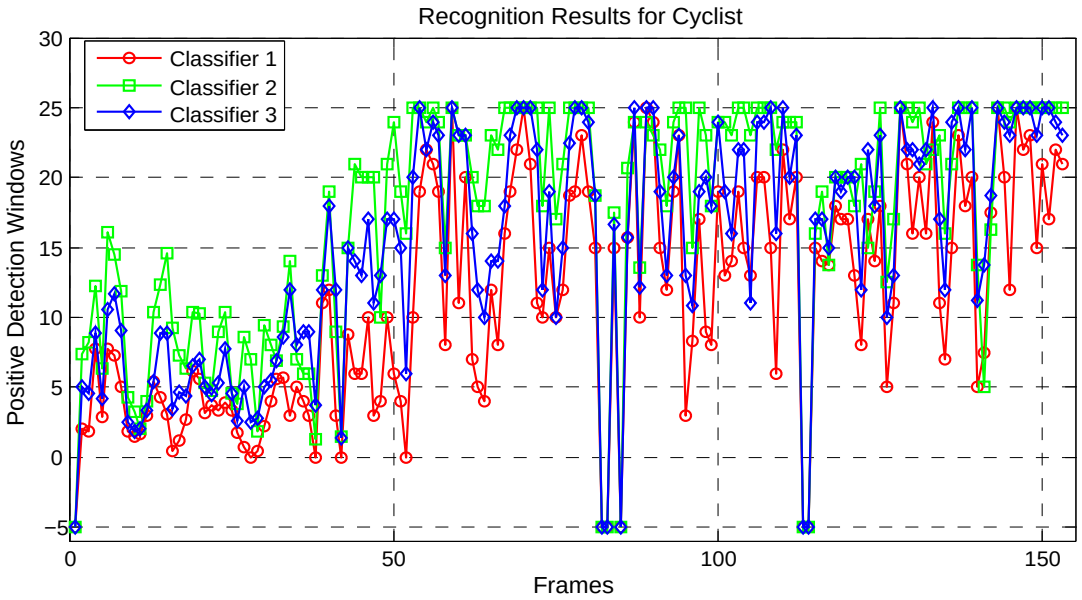


(a) *n_p for partly occluded pedestrian, $n_p = -5$ means that the object has not been detected in this frame*

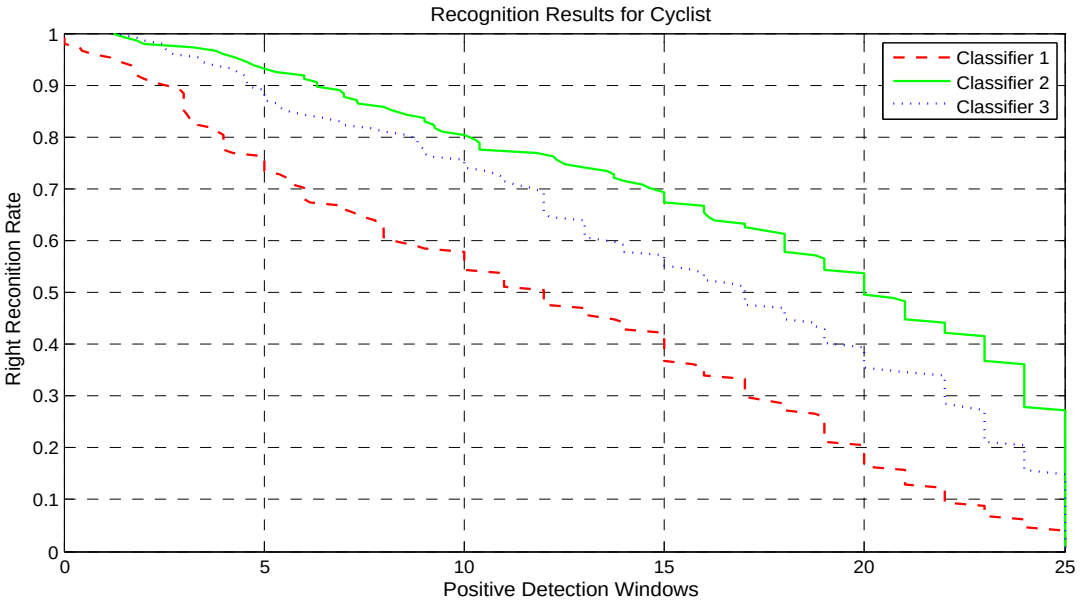


(b) *Recognition rate for partly occluded pedestrian*

Figure 3.21: *Recognition performance of three classifiers for partly-occluded pedestrian. The point with values of -5 at y-axis means that the bounding box of the object is not detected in this frame.*



(a) *n_p for cyclist, $n_p = -5$ means that the object has not been detected in this frame*



(b) *Recognition rate for cyclist*

Figure 3.22: *Recognition performance of three classifiers for cyclist. The point with values of -5 at y-axis means that the bounding box of the object is not detected in this frame.*

Chapter 4

Conclusions and Perspectives

Contents

4.1 Conclusions

THIS thesis addresses the problem of perceiving the surrounding environment of the host vehicle using stereo image sequences in various traffic scenes. The purpose work is included in the class of advanced driver assistance system (ADAS) for intelligent vehicles. Recently, ADAS are still a challenge due to the complexity of urban environments. Vision-based approaches play an important role in overcoming these difficulties because of the rich texture, color and depth information in the images. Two elements are crucial in ADAS: the state of the host vehicle and the state of the surrounding environment. The aims of this thesis is to detect the moving objects in the surrounding environment and to provide their categories and location information.

First of all, a general description of multi-modal based ADAS has been given at the beginning of this manuscript. Then we give some advantages of using vision sensors in ADAS compared to the use of Lidar sensors. In Chapter 2, our vision-based moving object detection algorithm has been introduced in details. The state of the art has been presented and an analysis of the advantages and drawbacks of the main methods has been proposed. Then, we have presented an effective moving object detection approach by modeling the uncertainties in ego-motion estimation and disparity computation. First, the residual image motion flow (RIMF) is computed to

distinguish the moving or not-moving pixels. Because of the non-uniform distribution of the RIMF over the whole image, some false positive alarms are generated if a simple threshold is applied. In order to solve this problem, we calculate the motion likelihood of the RIMF for each pixel by using its covariance matrix, computed using a Gaussian error propagation strategy. Then, the motion likelihood together with the depth derivative is incorporated into a graph-cut optimization framework to segment the moving objects. The effectiveness of our approach has been shown on several public datasets. The experimental results show that our algorithm can achieve satisfactory performances in different urban traffic scenes. Small moving objects, partially occluded moving objects and objects moving on the epipolar plane can be detected. For an image with 375*1200 resolution, the proposed approach can achieve about 30 seconds per frame.

In Chapter 3, the problem of pedestrian recognition has been introduced. We first present the challenges and motivations for pedestrian detection and recognition issues. A general summary of the latest pedestrian recognition approaches has been provided. Then we present our approach to address this issue. This approach can be decomposed into three parts. First, we select features via a PCA-based analysis of the HOG features. Then, the unlabeled training instances are clustered using a Gaussian mixture model in order to compute the soft class labels. Finally, a soft-label boosting algorithm is trained on the soft labeled training images. This algorithm consists in replacing the standard decision trees algorithm by an approach which makes use of data with probabilistic labels. The error criterion used classically in AdaBoost can be replaced by a cost-weighted error criterion. The experimental results show that this semi-supervised approach gives good results on classical datasets as well as on the real traffic sequences.

4.2 Perspectives

Vision-based advanced driver assistance systems, especially pedestrian protection system, are still young and promising in the field of robotic and intelligent vehicles. We consider the following for the future works.

Real-Time Application

Computation efficiency is a crucial requirement of a system for being applied in a real environment setting. However, our system can not be implemented in real-time due to the following reasons: first, dense optical flow and disparity map computa-

tion is time consuming. Despite great improvements in image processing hardware, parallel processing techniques and several technical breakthroughs in computer vision, approach satisfying both accurate and efficient computation have not been developed yet. In order to reduce the system's computation time, two alternatives may be considerd. The first one is to develop new dense optical flow and disparity map algorithms to reach the real time demand. Alternatively, 3D scene flow may be computed directly for moving object detection, as a coupled approach for estimating the 3D geometry and motion densely at every pixel from two consecutive stereo frames. Related work can be found in [145, 146, 15, 147, 148]. Our approach aims at estimating the motion likelihood for each image pixel first, and then to segment the moving objects based on this motion and structure information. In our approach, dense optical flow computation costs about half of the execution time. Therefore, a high-efficiency dense optical flow approach will highly reduce the execution time of our approach. In addition, the motion likelihood computation costs about 10 seconds each frame. Parallel processing technique can be applied in this step because the computation process is independent of each pixel.

Object Tracking for Trajectories Construction

After we detect all the moving objects surrounding our host vehicle, their motion trajectories should be constructed through tracking algorithms such as Kalman filter, Particle filter, Probability Hypothesis Density (PHD) filter or online learning based methods. Based on their motion trajectories, the motion behaviors can be easily analyzed and predicted. In the framework of tracking-by-detection, one object trajectory will collapse if the object's bounding box has not been generated in the following several frames. However, we find some useful cues of the object's location in the motion likelihood image. Usually, the pixels of the object have higher motion likelihoods than background pixels in the motion likelihood image even its bounding box has not been generated. We can integrate this information into a tracking approach [149] to improve our detection performances.

Boosting With Prior Information

In the future, we plan to design our boosting approach to improve the classifier's performance by using additional samples with weak prior knowledge (e.g., human beliefs or uniform class probabilities). The theory of belief functions [150] is another way of representing uncertain class information. Belief decision trees [151] and credal boosting [152] have been proposed to handle the classification problem under the

belief function framework. Additionally, the class labels of the training instances could be assigned by combining or fusing several expert's opinion [153, 154, 155]. A modified boosting algorithm can be proposed by using these training samples with credal class labels.

Appendix A

Dense Feature Matching and Tracking

A.1 Dense Pixel Matching and Tracking

Pixel matching and tracking are two common and crucial techniques in the field of computer vision. Due to the rapid development in the image processing hardware, dense approaches recieve more and more interest. A general overview of dense pixel matching and tracking approaches is given in the following paragraphs.

A.1.1 Dense Pixel Matching

The crucial point of the matching algorithm is to solve the correspondence problem between the left and right images. Although the stereo image rectification process may reduce the search problem from 2D to 1D, it is also not an easy task due to texture insufficient, luminance changes and views point (e.g. some parts of the scene can only be seen in one camera). Alough amount of works have been proposed. Some surveys of stereo vision algorithms can be found in [156, 157, 158]. In general the stereo algorithms can be categorized into sparse and dense approaches. Dense approaches gain popularity as the computational power grows. Dense approaches draw more and more attention compared to sparse ones. They can be divided into local and global methods based on the kinds of constraints (local or global) they use.

Local Methods build constraints using a local surrounding area for a interest pixel. Although local methods are efficient owning to their local constraints, they are very sensitive to locally ambiguous regions in images such as regions with similar textures or occlusions. Block matching, gradient methods and feature matching are three typical strategies used in local methods. Block matching methods seek to estimate the disparity for a pixel by building a small region (rectangle or circle window) around this pixel in the left image and finding the optimal correspondent pixel in the right image by comparing the similarity using a sliding window. The search region reduces to 1D by using the epipolar constraint. Normalized cross correlation (NCC), the sum of squared differences (SSD) metric and the sum of absolute differences (SAD) are popular statistical methods for determining the similarity between two small image windows. Gradient-based methods seek to determine small local disparities between two images by formulating a differential equation relating motion and image brightness. For this purpose, the assumption is made that the image brightness of a point in the scene is constant between the two views. Block matching and gradient methods are well-known to be sensitive to depth discontinuities, since the region of support near a discontinuity contains points from more

than one depth. These methods are also sensitive to regions with uniform texture in images. Feature-based methods seek to overcome these problems by limiting the regions of support to specific reliable features in the images. The reader may refer to [159, 160, 161, 162] for more infromation.

Global methods exploit global constraints in order to reduce sensitivity to occlusion or texture uniformity. Global methods aim at building the optimal disparity function by minimizing a global cost function. Usually this cost function combines data and smoothness terms. Compared to local methods, global ones produce much more accurate results, but are more time consuming and computational demanding. Dynamic Programming techniques [163, 164], Graph Cuts (GC) [165, 166] and Belief Propagation [167, 168] have been used to obtain the dense disparity map. An approach called Semi-Global Matching [37, 169] has been proposed to solve the computation efficiency. Many real time approaches [170, 171, 46] have been proposed so far with the rapid development of parallel computing and the GPU processing.

A.1.2 Dense Optical Flow Estimation

The aim of optical flow estimation is to compute an approximation to the motion field from time-varying image intensity. Due to its significance in the field of computer vision, various approaches have been proposed. Dense optical flow was firstly proposed by Horn and Schunck [172] in 1981. Since them many approaches have been introduced to solve this problem. A general review of optical flow computation and evaluation is introduced in [173, 174, 175]. The most common assumption used in optical flow is brightness constancy assumption. However, this assumption does not hold in some special cases, such as changes of illumination. Another ambiguity in optical flow computation is caused by the aperture problem [176, 177], which requires some additional motion information or assumptions to be provided. One common assumption is the smoothness of the optical flow field which is known as regularization. Depending on the type of the regularization, optical flow can be categorized into feature-based approaches and the variational approaches. More information of different dense optical flow approaches can be found in [178, 179].

Appendix B

Sparse Key Point Extraction and Matching

B.1 Sparse Feature Extraction, Tracking and Matching

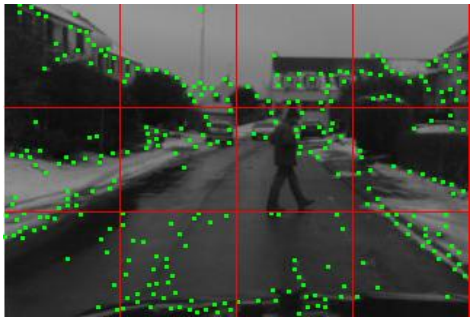
Key points are widely used to represent the image information in a sparse manner. Indeed, key points can represent most of the meaningful information of the scene and their processing is more efficient than handling all the image pixels. Usually, key points appear at specific locations in the image, such as corners, blobs or edges. These kinds of local image feature are often called key-points features or interest points features, and correspond to the appearance of pixel patches around the point location. Key-point features can be used to find a sparse set of corresponding points in different images, often for camera relative pose estimation or for images alignments. Generally, key points detection and matching procure can be realized in two stages. First is feature detection (extraction). The whole image is processed to extract the most significant locations which are likely to match well in other images. Second is feature based matching and tracking. Indeed both of them aim to find the corresponding points from one image to another. In stereo vision system, the correspondence features can be searched along the epipolar line by using the geometry constraint. Thanks to the stereo rectification process, the epipolar lines coincide with the rows of the left and right images. For each feature, we search among all candidates to find the best matching via a template blocking matching strategy. Zero mean Normalized Cross-Correlation (ZNCC) [180] is used to measure the similarity between template and original image blocks here because the normalized methods can reduce the effects of luminance variation. For the tracking, we used the classical Lucas-Kanade [48] method who has been proved to be robust and fast.

B.1.1 Bucketing-Based Feature Extraction

In order to uniformly detect features, a bucketing technique is employed in our algorithm. The whole image is divided into several non-overlapping blocks (see Fig. (B.1)-(c)). In each subimage (block), we extract features respectively and keep 5 points. We can choose varying thresholds in different blocks according to the image structure. This technique is beneficial in several ways. First of all, this technique guarantees that the used features are well distributed along the z-axis. This results in a precise estimation of the overall ego-motion of the vehicle. In addition, the key points are uniformly distributed over the whole image. This is especially important

(a) Harris feature extraction $\lambda_{ql} = 0.2$

(b)Harris feature extraction $\lambda_{ql} = 0.05$



(c) Harris feature extraction, bucketing results (about 25 points in each block)

Figure B.1: *Harris features extraction*

in dynamic scenes including some moving objects. The bucketing technique guarantees that some image features are on the static background rather than all on the moving objects. The details of this technique can be found in Alg. (B.1).

B.1.2 Feature-Based Stereo Matching

Here, we want to find the matched points between a rectified stereo image pair ($\mathbf{I}_l$ and $\mathbf{I}_r$) corresponding to the left and right views of a scene. First, evenly distributed features points $\mathbf{p}_{t-1,l}$ are extracted in the left image $\mathbf{I}_l$ based on Alg. (B.1). Then, matched feature points are searched in the right image according to the epipolar constraint and brightness similarity. The complete feature-based stereo matching is summarized in Alg. (B.2). It takes into account the brightness information and the geometrical constraints induced by the stereoscopic vision system. In real-time applications, this algorithm can be easily implemented and parallelized.

Algorithm B.1 Balanced features extraction

Input: - Image I
Parameters for features extraction:
- Size of the block bw *and* bh
- Minimum features in each block $minN$
- Initial feature quality threshold λ_{ql}
- Maximum times of change the feature quality threshold $maxIter$

Output: - Balanced distributed features $\mathbf{p}$ in image I

1: ▶ Divide the image into N subimages uniformly according to bw, bh
2: **for** $i = 0$ **do** N ▷ Extract features in each subimage
3: ▶ Extract features $\mathbf{p}_i$ in i_{th} subimage using the initial threshold λ_{ql} and the feature number is n_i.
4: **if** $0.75 * minN \leq n_i \leq 2{*}minN$ **then**
5: ▶ Add $\mathbf{p}_i$ into $\mathbf{p}$;
6: **else if** $n_i < 0.75 * minN$ **then**
7: ▶ $iter = 0; \lambda_{ql}^1 = \lambda_{ql}$; $n_i' = n_i$;
8: **while** $n_i' < 0.75 * minN$ & $iter < maxIter$ **do**
9: ▶ $\lambda_{ql}^1 = 0.5{*}\lambda_{ql}^1$; Extract features $\mathbf{p}_i'$ in i_{th} subimage using threshold λ_{ql}^1 and the feature number is n_i';
10: ▶ $iter = iter + 1$;
11: **end while**
12: ▶ Add $\mathbf{p}_i'$ into $\mathbf{p}$;
13: **else**
14: ▶ $iter = 0$; $\lambda_{ql}^1 = \lambda_{ql}$; $n_i' = n_i$;
15: **while** $n_i' > 2 * minN$ & $iter < maxIter$ **do**
16: ▶ $\lambda_{ql}^1 = 2{*}\lambda_{ql}^1$; Extract features $\mathbf{p}_i'$ in i_{th} subimage using threshold λ_{ql}^1 and the feature number is n_i';
17: ▶ $iter = iter + 1$;
18: **end while**
19: ▶ Add $\mathbf{p}_i'$ into $\mathbf{p}$;
20: **end if**
21: **end for**

Algorithm B.2 Feature-based stereo matching

Input: - Left image $\mathbf{I}_l$ and right image $\mathbf{I}_r$;
Parameters for Blocking Matching:
- Half block matching window size $b \times b$;
- Matching point searching range$[d_{min},\ d_{max}]$;
- ZNCC minimum acceptable value thr_{zncc}

Output: - Matched features $\mathbf{p}_{t-1}$ in left and right images;

1: ▶ Extract features in $\mathbf{I}_l$ based on Alg. (B.1), the features are $\mathbf{p}_{t-1,l}$ and the number is N;
2: ▶ $blockSize = 2 * b + 1$; image width W and height H
3: **for** $i = 0$ **do** N
4: ▶ Get the u *and* v of $\mathbf{p}^i_{t-1,l}$
5: **if** $u > b$ & $u < W - b$ & $v > b$ & $v < H - b$ **then**
6: ▶ The location of the block window: $v_{min} = v - b$, $v_{max} = v + b$,$u_{min} = u - b$, $u_{max} = u + b$;
7: ▶ The template image in $\mathbf{I}_l$: $templae = \mathbf{I}_l(v_{min}\ :\ v_{max},\ u_{min}\ :\ u_{max})$;
8: ▶ The searching range in $\mathbf{I}_r$: $numBlocks = min(d_{max} - d_{min},\ u - b)$;
9: **for** $j = 1$ **do** $numBlocks$ ▷ Search for the best matching point in $\mathbf{I}_r$
10: ▶ $block = \mathbf{I}_r(v_{min} : v_{max}, u_{min} - j\ :\ u_{max} - j)$;
11: ▶ Compute the ZNCC value $zncc_j = ZNCC(template,\ block)$;
12: **end for**
13: ▶ Find the index j_o with the highest ZNCC value $zncc_o$;
14: **if** $zncc_o > thr_{zncc}$ **then** ▷ We have found the matching point $\mathbf{I}_r$
15: ▶ The match point location is $(u - j_o,\ v)$ and save the matched point;
16: **end if**
17: **else**
18: ▶ The matching point can not be found and move the next point;
19: **end if**
20: **end for**
21: ▶ Return the matched feature points $\mathbf{p}_{t-1}$;

Appendix C

Uncertainty Propagation

C.1 Basic Knowledge of Covariance Matrix Estimation

It is very important to understand how to propagate a random perturbation through any algorithm step during parameters estimation process. Covariance matrix is one basic measure to describe the size of the random perturbation resulted from the input random noise. Generally, there are two kinds of methods to estimate covariance matrix in the real computer vision problems. The first one is the Monte Carlo method, which is easy and accurate but time consumption. Furthermore, it can only provide a solution to one specific problem. Another approach is based on the first order error propagation strategy, which uses a first-order Taylor series expansion to linearize the nonlinear problems. Compared to the Monte Carlo method, this method can give a close form solution but less accurate results.

C.1.1 Monte Carlo Method

Monte Carlo method is a ideal choice to test the validity of the first-order approximation methods, which can provide absolute confidence even for non-Gaussian distribution. Monte Carlo method can be simply expressed as: given a function $\mathbf{y} = f(\mathbf{x})$ and some (assumed perfectly known) data $\mathbf{x}$, a large population of corrupted data $(\mathbf{x_1} = \mathbf{x} + \mathbf{r_1}, \mathbf{x_2} = \mathbf{x} + \mathbf{r_2} \cdots, \mathbf{x_n} = \mathbf{x} + \mathbf{r_n})$ is created by repeatedly adding different kinds of random noise $\mathbf{r}_i$ to $\mathbf{x}$. The distribution of $\mathbf{y}$ from the distribution of samples $\mathbf{y}_i = f(\mathbf{x}_i)$ is estimated from the large population of $\mathbf{x}$. The covariance of $\mathbf{y}$ can be obtained as follow:

$$\Sigma_{\mathbf{y}} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{y}_i - \overline{\mathbf{y}})(\mathbf{y}_i - \overline{\mathbf{y}})^T \tag{C.1}$$

in which, $\overline{\mathbf{y}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i$. Note that the function $f(\ .\ )$ should to be known explicitly and the vector $\mathbf{y}$ could be found using a numerical optimization algorithm. If the probability density function (PDF) of the random noise $\mathbf{r}_i$ is chosen to be the same as the original data then it is possible to estimate the true PDF of $\mathbf{y}$ by the distribution of the $\mathbf{y}_i$ which has been found.

C.1.2 Covariance Matrix Using First Order Approximations

The inefficiency of Monte Carlo method is often a limitation in the practice application. Additionally, a closed form of the covariance matrix is more convenient to

analyze and understand the uncertainty in a real system. There are two general solutions of the covariance matrices according to the form of function $f(\,.\,)$.

C.1.2.1 Explicit functions

Given an explicit continuously differentiable function $f(\mathbf{x})$ and covariance matrix $\Sigma_{\mathbf{x}}$of data $\mathbf{x}$, we aims at finding the covariance matrix $\Sigma_{\mathbf{y}}$ of the result $\mathbf{y} = f(\mathbf{x})$. Taylor series expansion of $f(\mathbf{x})$ around expected value $\overline{\mathbf{x}}$ of $\mathbf{x}$ is shown as below:

$$f(\overline{\mathbf{x}} + \triangle\mathbf{x}) = f(\overline{\mathbf{x}}) + \nabla f(\overline{\mathbf{x}})\Delta\mathbf{x} + o(\|\Delta\mathbf{x}\|^2)$$

and the first order approximation to the covariance matrix for the estimated vector $\mathbf{y}^\star = f(\overline{\mathbf{x}})$ is give by:

$$\begin{aligned}
\Sigma_{\mathbf{y}} &= E([f(\overline{\mathbf{x}} + \triangle\mathbf{x}) - \overline{\mathbf{y}}][f(\overline{\mathbf{x}} + \triangle\mathbf{x}) - \overline{\mathbf{y}}]^T) \\
&\approx E([f(\overline{\mathbf{x}} + \triangle\mathbf{x}) - \overline{\mathbf{y}}][f(\overline{\mathbf{x}} + \triangle\mathbf{x}) - \overline{\mathbf{y}}]^T) \\
&\approx E(\nabla\mathbf{f} \triangle \mathbf{x} \triangle \mathbf{x}^T) \\
&= \nabla\mathbf{f}\ \Sigma_{\mathbf{x}}\ \nabla\mathbf{f}^{\mathbf{T}}
\end{aligned} \tag{C.2}$$

From Eq. (C.2), we can see that the covariance of the estimated value of $\mathbf{y}$ is only determined by the variance of $\mathbf{x}$ and the Jacobian Matrix of function $f(\ .\ )$. Two approximations have been introduced: the first one is $\overline{\mathbf{y}} \approx f(\overline{\mathbf{x}})$, using the estimated value of $\mathbf{y}^\star$ to replace for the respected value of $\overline{\mathbf{y}}$. This will bring some uncertainty for the covariance matrix. Another one is the truncation of the Taylor series using a linear approximation which also introduces some uncertainty.

C.1.2.2 Implicit function

The previous section describes how to calculate the covariance matrix when the function $f(\ .\ )$ is known explicitly and the Jacobian matrix is easily to be obtained by taking all the partial derivatives. However, in most cases of the computer vision problems, the estimated parameters are obtained by using an iterative algorithm. The object function defines an minimization between the data and results. We want to estimate the sensitivity of the vector parameters which minimize the object function of the original data to changes in the data. This problem is how to get the Jacobian matrix from the implicit function. Assuming that the implicit function (object function) is defined as:

$$F(\mathbf{x}, \Theta) = 0 \tag{C.3}$$

in which, Θ are the parameters needed to be estimated, $\mathbf{x}$ are the vector of observed data using to estimate θ. Set

$$g(\mathbf{x}, \Theta) = \frac{\partial F}{\partial \theta}(\mathbf{x}, \Theta) \tag{C.4}$$

and the covariance matrix of the estimated parameters $\hat{\Theta}$ could be expressed as below:

$$\Sigma_{\hat{\Theta}} = \frac{\partial g}{\partial \Theta}(\hat{\mathbf{x}}, \hat{\Theta})^{-1} \frac{\partial g}{\partial \mathbf{x}}^{T}(\hat{\mathbf{x}}, \hat{\Theta})) \; \Sigma_{\mathbf{x}} \; \frac{\partial g}{\partial \mathbf{x}}(\hat{\mathbf{x}}, \hat{\Theta}) \frac{\partial g}{\partial \theta}(\hat{\mathbf{x}}, \hat{\Theta})^{-1} \tag{C.5}$$

$\hat{\mathbf{x}}$ represent the observed data vector produced by the unobserved vector $\mathbf{x}$, $\hat{\mathbf{x}} = \mathbf{x} + \triangle\mathbf{x}$. $\hat{\Theta} = \Theta + \Delta\Theta$, represents the estimated parameters vector which equals to true unknown parameters Θ adding the random perturbation $\triangle\Theta$ propagated from the $\triangle\mathbf{x}$.

C.2 Uncertainty Propagation in Ego-Motion Estimation

Here, we rewrite the cost function of ego-motion estimation as below:

$$F(\Theta, \mathbf{x}) = \sum_{i=1}^{N} \|\mathbf{x}_t^i - \hat{\mathbf{x}}_t^i\|^2_{\Sigma_{\mathbf{x}_t^i}} = \sum_{i=1}^{N} \|\mathbf{x}_t^i - \mathbf{f}(\Theta, \mathbf{x}_{t-1}^i)\|^2_{\Sigma_{\mathbf{x}_t^i}} \forall i = 1 \cdots N, \tag{C.6}$$

where $\mathbf{x}_t^i = (u_{t,l}^i, v_{t,l}^i, u_{t,r}^i, v_{t,r}^i)^T$ and $\mathbf{x}_{t-1}^i = (u_{t-1,l}^i, v_{t-1,l}^i, u_{t-1,r}^i, v_{t-1,r}^i)^T$ are the matched image points at previous and current frames; $\hat{\mathbf{x}}_t^i = (\hat{u}_{t,l}^i, \hat{v}_{t,l}^i, \hat{u}_{t,r}^i, \hat{v}_{t,r}^i)^T$ is the predicted image points at current frame. The function $\mathbf{f}(\,.\,)$ represents the projection of image points from previous frame to current frame, which is expressed as:

$$\hat{\mathbf{x}}_t^i = \mathbf{f}(\Theta, \mathbf{x}_{t-1}^i) = \begin{pmatrix} \frac{r_{00}fb(u_{t-1,l}^i - u_0) + r_{01}fb(v_{t-1,l}^i - v_0) + r_{02}f^2b + d_i f t_x}{r_{20}b(u_{t-1,l}^i - u_0) + r_{21}b(v_{t-1,l}^i - v_0) + r_{22}fb + d_i t_z} + u_0 \\ \frac{r_{10}fb(u_{t-1,l}^i - u_0) + r_{11}fb(v_{t-1,l}^i - v_0) + r_{12}f^2b + d_i f t_y}{r_{20}b(u_{t-1,l}^i - u_0) + r_{21}b(v_{t-1,l}^i - v_0) + r_{22}fb + d_i t_z} + v_0 \\ \frac{r_{00}fb(u_{t-1,l}^i - u_0 - d_i) + r_{01}fb(v_{t-1,l}^i - v_0) + r_{02}f^2b + d_i f t_x}{r_{20}b(u_{t-1,l}^i - u_0 - d_i) + r_{21}b(v_{t-1,l}^i - v_0) + r_{22}fb + d_i t_z} + u_0 \\ \frac{r_{10}fb(u_{t-1,l}^i - u_0 - d_i) + r_{11}fb(v_{t-1,l}^i - v_0) + r_{12}f^2b + d_i f t_y}{r_{20}b(u_{t-1,l}^i - u_0 - d_i) + r_{21}b(v_{t-1,l}^i - v_0) + r_{22}fb + d_i t_z} + v_0 \end{pmatrix}, \tag{C.7}$$

where

- $r_{00} = cos(r_y)cos(r_z)$
- $r_{01} = -cos(r_y)sin(r_z)$
- $r_{02} = sin(r_y)$
- $r_{10} = sin(r_x)sin(r_y)cos(r_z) + cos(r_x)sin(r_z)$
- $r_{11} = -sin(r_x)sin(r_y)sin(r_z) + cos(r_x)cos(r_z)$
- $r_{12} = -sin(r_x)cos(r_y)$
- $r_{20} = -cos(r_x)sin(r_y)cos(r_z) + sin(r_x)sin(r_y)$
- $r_{21} = cos(r_x)sin(r_y)sin(r_z) + sin(r_x)cos(r_z)$
- $r_{22} = cos(r_x)cos(r_y)$

Obviously, covariance matrix of Θ should be calculated using Eq. (C.5) as

$$\Sigma_\Theta = \left(\frac{\partial g}{\partial \Theta}\right)^{-1} \left(\frac{\partial g}{\partial \mathbf{x}}\right)^{T} \Sigma_{\mathbf{x}} \left(\frac{\partial g}{\partial \mathbf{x}}\right) \left(\frac{\partial g}{\partial \Theta}\right)^{-T}, \tag{C.8}$$

where $g(\mathbf{x}, \Theta) = \frac{\partial F(\mathbf{X},\Theta)}{\partial \Theta}$ is the gradient vector of $F(\Theta, \mathbf{x})$ of Θ and $\Sigma_{\mathbf{x}}$.

According to Eq. (C.6), the lost function is a sum lost of N pair points. So in order to simplify the formula expression and not lose generality, we are able to only consider one pair point in the following covariance matrix computation steps. Assume that

$$\begin{aligned} F'(\Theta, \mathbf{x}) &= \|\mathbf{x}_t - \mathbf{f}(\Theta, \mathbf{x}_{t-1})\|^2_{\Sigma_{\mathbf{x}_t}} \\ &= (\mathbf{x}_t - \mathbf{f}(\Theta, \mathbf{x}_{t-1}))^T \Sigma^{-1}_{\mathbf{x}_t} (\mathbf{x}_t - \mathbf{f}(\Theta, \mathbf{x}_{t-1})) \end{aligned}, \tag{C.9}$$

so $g'(\mathbf{x}, \Theta) = \frac{\partial F'(\mathbf{X},\mathbf{\Theta})}{\partial \Theta}$ can be expressed as

$$\begin{aligned} g'(\mathbf{x}, \Theta) &= \frac{\partial((\mathbf{x}_t - \mathbf{f}(\Theta, \mathbf{x}_{t-1}))^T)}{\partial \Theta} \Sigma^{-1}_{\mathbf{x}_t} (\mathbf{x}_t - \mathbf{f}(\Theta, \mathbf{x}_{t-1})) + \\ &\quad \frac{\partial(\Sigma^{-1}_{\mathbf{x}_t}(\mathbf{x}_t - \mathbf{f}(\Theta, \mathbf{x}_{t-1})))^T}{\partial \Theta} (\mathbf{x}_t - \mathbf{f}(\Theta, \mathbf{x}_{t-1})) \\ &= \left(\frac{2\partial(\mathbf{x}_t - \mathbf{f}(\Theta, \mathbf{x}_{t-1}))}{\partial \Theta}\right)^T \Sigma^{-1}_{\mathbf{x}_t} (\mathbf{x}_t - \mathbf{f}(\Theta, \mathbf{x}_{t-1})) \\ &= -2\left(\frac{2\partial(\mathbf{f}(\Theta, \mathbf{x}_{t-1}))}{\partial \Theta}\right)^T \Sigma^{-1}_{\mathbf{x}_t} (\mathbf{x}_t - \mathbf{f}(\Theta, \mathbf{x}_{t-1})) \end{aligned} \tag{C.10}$$

After obtaining $g'(\mathbf{x}, \Theta)$, the partial derivatives, $\frac{\partial g'}{\partial \Theta}$ and $\frac{\partial g'}{\partial \mathbf{x}}$ of $g'(\Theta, \mathbf{x})$ can be calculated as:

$$\begin{aligned}\frac{\partial g'}{\partial \Theta} &= \frac{\partial(-2\left(\frac{2\partial(\mathbf{x}_t-\mathbf{f}(\Theta,\mathbf{x}_{t-1}))}{\partial\Theta}\right)^T\Sigma_{\mathbf{x}_t}^{-1}(\mathbf{x}_t-\mathbf{f}(\Theta,\mathbf{x}_{t-1})))}{\partial\Theta} \\ &= -2\frac{\partial^2(\mathbf{f}(\Theta,\mathbf{x}_{t-1}))}{\partial\Theta^2}\Sigma_{\mathbf{x}_t}^{-1}(\mathbf{x}_t-\mathbf{f}(\Theta,\mathbf{x}_{t-1}))- \\ &\quad 2\frac{\partial(\Sigma_{\mathbf{x}_t}^{-1}(\mathbf{x}_t-\mathbf{f}(\Theta,\mathbf{x}_{t-1})))^T}{\partial\Theta}.\frac{\partial(\mathbf{f}(\Theta,\mathbf{x}_{t-1}))}{\partial\Theta} \\ &= 2(\frac{\partial\mathbf{f}}{\partial\Theta})^T\Sigma_{\mathbf{x}_t}^{-1}\frac{\partial\mathbf{f}}{\partial\Theta}-2\frac{\partial^2\mathbf{f}}{\partial\Theta^2}\Sigma_{\mathbf{x}_t}^{-1}(\mathbf{x}_t-\mathbf{f}(\Theta,\mathbf{x}_{t-1}))\end{aligned} \tag{C.11}$$

and

$$\begin{aligned}\frac{\partial g'}{\partial \mathbf{x}} &= \begin{pmatrix}\frac{\partial g'}{\partial \mathbf{x}_{t-1}}\\ \frac{\partial g'}{\partial \mathbf{x}_t}\end{pmatrix} = \begin{pmatrix}\frac{\partial(-2\left[\frac{2\partial(\mathbf{f}(\Theta,\mathbf{x}_{t-1}))}{\partial\Theta}\right]^T\Sigma_{\mathbf{x}_t}^{-1}(\mathbf{x}_t-\mathbf{f}(\Theta,\mathbf{x}_{t-1})))}{\partial\mathbf{x}_{t-1}}\\ \frac{\partial(-2\left[\frac{2\partial(\mathbf{f}(\Theta,\mathbf{x}_{t-1}))}{\partial\Theta}\right]^T\Sigma_{\mathbf{x}_t}^{-1}(\mathbf{x}_t-\mathbf{f}(\Theta,\mathbf{x}_{t-1})))}{\partial\mathbf{x}_t}\end{pmatrix}, \\ &= \begin{pmatrix}2(\frac{\partial\mathbf{f}}{\partial\mathbf{x}_{t-1}})^T\Sigma_{\mathbf{x}_t}^{-1}\frac{\partial\mathbf{f}}{\partial\Theta}-2\frac{\partial^2\mathbf{f}}{\partial\Theta\partial\mathbf{x}_{t-1}}\Sigma_{\mathbf{X_t}}^{-1}(\mathbf{x}_t-\mathbf{f}(\Theta,\mathbf{x}_{t-1}))\\ -2\Sigma_{\mathbf{x}_t}^{-1}\frac{\partial\mathbf{f}(\Theta,\mathbf{x}_{t-1})}{\partial\Theta}\end{pmatrix}\end{aligned} \tag{C.12}$$

where $\frac{\partial\mathbf{f}}{\partial\Theta}$ and $\frac{\partial\mathbf{f}}{\partial\mathbf{x}_{t-1}}$ are the partial derivatives of $\mathbf{f}(.)$ which can be calculated according to Eq. (C.7). Here, $\frac{\partial\mathbf{f}}{\partial\Theta}$ is a 4×6 matrix and $\frac{\partial\mathbf{f}}{\partial\mathbf{x}}$ is a 4×4 matrix, they are calculated as:

$$\frac{\partial\mathbf{f}}{\partial\Theta}|_{6\times 4} = \begin{bmatrix}\frac{\partial f_1}{\partial r_x} & \frac{\partial f_1}{\partial r_y} & \frac{\partial f_1}{\partial r_z} & \frac{\partial f_1}{\partial t_x} & \frac{\partial f_1}{\partial t_y} & \frac{\partial f_1}{\partial t_z}\\ \frac{\partial f_2}{\partial r_x} & \frac{\partial f_2}{\partial r_y} & \frac{\partial f_2}{\partial r_z} & \frac{\partial f_2}{\partial t_x} & \frac{\partial f_2}{\partial t_y} & \frac{\partial f_2}{\partial t_z}\\ \frac{\partial f_3}{\partial r_x} & \frac{\partial f_3}{\partial r_y} & \frac{\partial f_3}{\partial r_z} & \frac{\partial f_3}{\partial t_x} & \frac{\partial f_3}{\partial t_y} & \frac{\partial f_3}{\partial t_z}\\ \frac{\partial f_4}{\partial r_x} & \frac{\partial f_4}{\partial r_y} & \frac{\partial f_4}{\partial r_z} & \frac{\partial f_4}{\partial t_x} & \frac{\partial f_4}{\partial t_y} & \frac{\partial f_4}{\partial t_z}\end{bmatrix} \tag{C.13}$$

and

$$\frac{\partial\mathbf{f}}{\partial\mathbf{x}_{t-1}}|_{4\times 4} = \begin{bmatrix}\frac{\partial f_1}{\partial u_{t-1,l}} & \frac{\partial f_1}{\partial v_{t-1,l}} & \frac{\partial f_1}{\partial u_{t-1,r}} & \frac{\partial f_1}{\partial v_{t-1,r}}\\ \frac{\partial f_2}{\partial u_{t-1,l}} & \frac{\partial f_2}{\partial v_{t-1,l}} & \frac{\partial f_2}{\partial u_{t-1,r}} & \frac{\partial f_2}{\partial v_{t-1,r}}\\ \frac{\partial f_3}{\partial u_{t-1,l}} & \frac{\partial f_3}{\partial v_{t-1,l}} & \frac{\partial f_3}{\partial u_{t-1,r}} & \frac{\partial f_3}{\partial v_{t-1,r}}\\ \frac{\partial f_4}{\partial u_{t-1,l}} & \frac{\partial f_4}{\partial v_{t-1,l}} & \frac{\partial f_4}{\partial u_{t-1,r}} & \frac{\partial f_4}{\partial v_{t-1,r}}\end{bmatrix}. \tag{C.14}$$

where $f_i, i = 1,\cdots,4$ represents element of row i in $\mathbf{f}(.)$. In order to save space, we only give one example of calculating the element of $\frac{\partial\mathbf{f}}{\partial\Theta}$ and $\frac{\partial\mathbf{f}}{\partial\mathbf{x}}$ here. Other elements can be computed in similar way.

$$\frac{\partial f_1}{\partial r_x} = \frac{\partial\left(\frac{r_{00}fb(u^i_{t-1,l}-u_0)+r_{01}fb(v^i_{t-1,l}-v_0)+r_{02}f^2b+d_ift_x}{r_{20}b(u^i_{t-1,l}-u_0)+r_{21}b(v^i_{t-1,l}-v_0)+r_{22}fb+d_it_z}+u_0\right)}{\partial r_x} \tag{C.15}$$

and

$$\frac{\partial f_1}{\partial u_{t-1,l}} = \frac{\partial\left(\frac{r_{00}fb(u^i_{t-1,l}-u_0)+r_{01}fb(v^i_{t-1,l}-v_0)+r_{02}f^2b+d_ift_x}{r_{20}b(u^i_{t-1,l}-u_0)+r_{21}b(v^i_{t-1,l}-v_0)+r_{22}fb+d_it_z}+u_0\right)}{\partial u_{t-1}}. \tag{C.16}$$

In order to express simply, we assume that $A = r_{00}fb(u^i_{t-1,l} - u_0) + r_{01}fb(v^i_{t-1,l} - v_0) + r_{02}f^2b + d_i f t_x$ and $B = r_{20}b(u^i_{t-1,l} - u_0) + r_{21}b(v^i_{t-1,l} - v_0) + r_{22}fb + d_i t_z$. So Eq.(C.15) and Eq. (C.16) can be simplified as

$$\frac{\partial f_1}{\partial r_x} = \frac{\frac{\partial A}{\partial r_x}B + A\frac{\partial B}{\partial r_x}}{(r_{20}b(u^i_{t-1,l} - u_0) + r_{21}b(v^i_{t-1,l} - v_0) + r_{22}fb + d_i t_z)^2} \tag{C.17}$$

and

$$\frac{\partial f_1}{\partial u_{t-1,l}} = \frac{\frac{\partial A}{\partial u_{t-1,l}}B + A\frac{\partial B}{\partial u_{t-1,l}}}{(r_{20}b(u^i_{t-1,l} - u_0) + r_{21}b(v^i_{t-1,l} - v_0) + r_{22}fb + d_i t_z)^2}. \tag{C.18}$$

We can calculate $\frac{\partial A}{\partial r_x}$, $\frac{\partial B}{\partial r_x}$, $\frac{\partial A}{\partial u_{t-1,l}}$and $\frac{\partial B}{\partial u_{t-1,l}}$ respectively as below:

$$\begin{aligned} \frac{\partial A}{\partial r_x} &= \frac{\partial r_{00}}{\partial r_x}fb(u^i_{t-1,l} - u_0) + \frac{\partial r_{01}}{\partial r_x}fb(v^i_{t-1,l} - v_0) + \frac{\partial r_{02}}{\partial r_x}f^2b + \frac{\partial (d_i f t_x)}{\partial r_x} \\ &= 0 \end{aligned};$$

$$\begin{aligned} \frac{\partial B}{\partial r_x} &= \frac{\partial r_{20}}{\partial r_x}b(u^i_{t-1,l} - u_0) + \frac{\partial r_{21}}{\partial r_x}b(v^i_{t-1,l} - v_0) + \frac{\partial r_{22}}{\partial r_x}fb + \frac{\partial (d_i t_z)}{\partial r_x} \\ &= (sin(r_x)sin(r_y)cos(r_z) + cos(r_x)sin(r_y)).b(u^i_{t-1,l} - u_0) + \\ &\quad (-sin(r_x)sin(r_y)sin(r_z) + cos(r_x)cos(r_z)).b(v^i_{t-1,l} - v_0) - sin(r_x)cos(r_y).fb \end{aligned}; \tag{C.19}$$

$$\begin{aligned} \frac{\partial A}{\partial u_{t-1,l}} &= \frac{\partial (r_{00}fb(u^i_{t-1,l} - u_0) + r_{01}fb(v^i_{t-1,l} - v_0) + r_{02}f^2b + d_i f t_x)}{\partial u_{t-1,l}} \\ &= r_{00}fb \\ &= cos(r_y)cos(r_z)fb \end{aligned} \tag{C.20}$$

and

$$\begin{aligned} \frac{\partial B}{\partial u_{t-1,l}} &= \frac{\partial (r_{20}b(u^i_{t-1,l} - u_0) + r_{21}b(v^i_{t-1,l} - v_0) + r_{22}fb + d_i t_z)}{\partial u_{t-1,l}} \\ &= r_{20}b \\ &= b.(-cos(r_x)sin(r_y)cos(r_z) + sin(r_x)sin(r_y)) \end{aligned}. \tag{C.21}$$

Finally, substitute Eq. (C.9) and (C.10) into Eq. (C.8), the covariance matrix of Θ can be obtained.

C.3 Uncertainty Propagation for RIMF

We use $(p_u, p_v)^T$ to represent the RIMF at location (u, v), which is defined as

$$RIMF(u,v) = \begin{pmatrix} \frac{r_{00}fb(u_{t-1}-u_0)+r_{01}fb(v_{t-1}-v_0)+r_{02}f^2b+dft_x}{r_{20}b(u_{t-1}-u_0)+r_{21}b(v_{t-1}-v_0)+r_{22}fb+dt_z} - u_{t-1} + u_0 - \triangle u' \\ \frac{r_{10}fb(u_{t-1}-u_0)+r_{11}fb(v_{t-1}-v_0)+r_{12}f^2b+dft_y}{r_{20}b(u_{t-1}-u_0)+r_{21}b(v_{t-1}-v_0)+r_{22}fb+dt_z} - v_{t-1} + v_0 - \triangle v' \end{pmatrix}. \tag{C.22}$$

In order to calculate the covariance matrix Σ_{RIMF} of $RIMF$, Eq. (C.5) should be considered because the $RIMF$ is represented as a explicit function in Eq. (C.22). As described in Eq. (C.5), the Jacobian matrix $\mathbf{J}$ of $RIMF$ with respect to each input variable should be computed firstly.

$$\mathbf{J} = \begin{pmatrix} \frac{\partial p_u}{\partial r_x} & \frac{\partial p_u}{\partial r_y} & \frac{\partial p_u}{\partial r_z} & \frac{\partial p_u}{\partial t_x} & \frac{\partial p_u}{\partial t_y} & \frac{\partial p_u}{\partial t_z} & \frac{\partial p_u}{\partial u} & \frac{\partial p_u}{\partial v} & \frac{\partial p_u}{\partial d} \\ \frac{\partial p_v}{\partial r_x} & \frac{\partial p_v}{\partial r_y} & \frac{\partial p_v}{\partial r_z} & \frac{\partial p_v}{\partial t_x} & \frac{\partial p_v}{\partial t_y} & \frac{\partial p_v}{\partial t_z} & \frac{\partial p_v}{\partial u} & \frac{\partial p_v}{\partial v} & \frac{\partial p_v}{\partial d} \end{pmatrix} \tag{C.23}$$

As in Sec. (C.2), here we just calculate several elements of $\mathbf{J}$ as examples and the rest can be computed easily in the similar way. The first element $\frac{\partial p_u}{\partial r_x}$ is computed as below:

$$\begin{aligned} \frac{\partial p_u}{\partial r_x} &= \frac{\partial(\frac{r_{00}fb(u_{t-1}-u_0)+r_{01}fb(v_{t-1}-v_0)+r_{02}f^2b+dft_x}{r_{20}b(u_{t-1}-u_0)+r_{21}b(v_{t-1}-v_0)+r_{22}fb+dt_z} - u_{t-1} + u_0 - \triangle u')}{\partial r_x} \\ &= \frac{\partial(\frac{r_{00}fb(u_{t-1}-u_0)+r_{01}fb(v_{t-1}-v_0)+r_{02}f^2b+dft_x}{r_{20}b(u_{t-1}-u_0)+r_{21}b(v_{t-1}-v_0)+r_{22}fb+dt_z} + u_0)}{\partial r_x} \end{aligned}. \tag{C.24}$$

If we compare Eq. (C.24) with Eq. (C.15), we can easily find that they share the same results. So $\frac{\partial p_u}{\partial r_x}$ can be calculated using the similar way described in Sec. (C.2). Similarly, other elements of $\mathbf{J}$ can also be calculated using the same way that mentioned above.

References

[1] Yue Wang, Eam Khwang Teoh, and Dinggang Shen. Lane detection and tracking using b-snake. *Image and Vision computing*, 22(4):269–280, 2004.

[2] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517. IEEE, 2012.

[3] Yinghua He, Hong Wang, and Bo Zhang. Color-based road detection in urban traffic scenes. *Intelligent Transportation Systems, IEEE Transactions on*, 5(4):309–318, 2004.

[4] Hui Kong, J-Y Audibert, and Jean Ponce. General road detection from a single image. *Image Processing, IEEE Transactions on*, 19(8):2211–2220, 2010.

[5] Bihao Wang and Vincent Frémont. Fast road detection from color images. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 1209–1214. IEEE, 2013.

[6] Jun Miura, Tsuyoshi Kanda, and Yoshiaki Shirai. An active vision system for real-time traffic sign recognition. In *Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE*, pages 52–57. IEEE, 2000.

[7] Arturo De la Escalera, J Ma Armingol, and Mario Mata. Traffic sign recognition and analysis for intelligent vehicles. *Image and vision computing*, 21(3):247–258, 2003.

[8] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[9] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based mod-

els. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[10] Zehang Sun, George Bebis, and Ronald Miller. On-road vehicle detection: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(5):694–711, 2006.

[11] Feng Liu and Michael Gleicher. Learning color and locality cues for moving object detection and segmentation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 320–327. IEEE, 2009.

[12] Heechul Jung, Jeongwoo Ju, and Junmo Kim. Rigid motion segmentation using randomized voting. In *Computer Vision and Pattern Recognition, 2014. CVPR 2014. Proceedings of the 2014 IEEE Computer Society Conference on*. IEEE, 2014.

[13] Chang Yuan, Gerard Medioni, Jinman Kang, and Isaac Cohen. Detecting motion regions in the presence of a strong parallax from a moving camera by multiview geometric constraints. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(9):1627–1641, 2007.

[14] Soumyabrata Dey, Vladimir Reilly, Imran Saleemi, and Mubarak Shah. Detection of independently moving objects in non-planar scenes via multi-frame monocular epipolar constraint. In *Computer Vision–ECCV 2012*, pages 860–873. Springer, 2012.

[15] Andreas Wedel, Thomas Brox, Tobi Vaudrey, Clemens Rabe, Uwe Franke, and Daniel Cremers. Stereoscopic scene flow computation for 3d motion understanding. *International Journal of Computer Vision*, 95(1):29–51, 2011.

[16] Victor Romero-Cano and Juan I Nieto. Stereo-based motion detection and tracking from a moving platform. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 499–504. IEEE, 2013.

[17] Stefan Walk, Konrad Schindler, and Bernt Schiele. Disparity statistics for pedestrian detection: Combining appearance, motion and stereo. In *Computer Vision–ECCV 2010*, pages 182–195. Springer, 2010.

[18] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):743–761, 2012.

[19] Javier Marin, David Vázquez, Antonio M López, Jaume Amores, and Bastian Leibe. Random forests of local experts for pedestrian detection. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2592–2599. IEEE, 2013.

[20] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

[21] Noah Snavely, Steven M Seitz, and Richard Szeliski. Skeletal graphs for efficient structure from motion. In *CVPR*, volume 1, page 2, 2008.

[22] Takeo Miyasaka, Yoshihiro Ohama, and Yoshiki Ninomiya. Ego-motion estimation and moving object tracking using multi-layer lidar. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 151–156. IEEE, 2009.

[23] Julien Moras, Véronique Cherfaoui, and Philippe Bonnifait. Moving objects detection by conflict analysis in evidential grids. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 1122–1127. IEEE, 2011.

[24] Lijun Wei, Cindy Cappelle, Yassine Ruichek, and Frédérick Zann. Intelligent vehicle localization in urban environments using ekf-based visual odometry and gps fusion. In *In 18th IFAC World*, pages 13776–13781, 2011.

[25] Lijun Wei, Cindy Cappelle, and Yassine Ruichek. Camera/laser/gps fusion method for vehicle positioning under extended nis-based sensor validation. *Instrumentation and Measurement, IEEE Transactions on*, 62(11):3110–3122, 2013.

[26] Soumyabrata Dey, Vladimir Reilly, Imran Saleemi, and Mubarak Shah. Detection of independently moving objects in non-planar scenes via multi-frame monocular epipolar constraint. In *ECCV*, pages 860–873. Springer, 2012.

[27] You Li and Yassine Ruichek. Moving objects detection and recognition using sparse spatial information in urban environments. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 1060–1065. IEEE, 2012.

[28] Shireen Y Elhabian, Khaled M El-Sayed, and Sumaya H Ahmed. Moving object detection in spatial domain using background removal techniques-state-of-art. *Recent patents on computer science*, 1(1):32–54, 2008.

[29] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28–31. IEEE, 2004.

[30] Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[31] Abhijit Kundu, K. Madhava Krishna, and Jayanthi Sivaswamy. Moving object detection by multi-view geometric techniques from a single camera mounted robot. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4306–4312, 2009.

[32] Abhijit Kundu, K Madhava Krishna, and CV Jawahar. Realtime multibody visual slam with a smoothly moving monocular camera. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2080–2087. IEEE, 2011.

[33] Michal Irani and P Anandan. A unified approach to moving object detection in 2d and 3d scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(6):577–589, 1998.

[34] Qian Yu and Gérard Medioni. A gpu-based implementation of motion detection from a moving platform. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–6. IEEE, 2008.

[35] Aurelie Bugeau and Patrick Perez. Detection and segmentation of moving objects in complex scenes. *Computer Vision and Image Understanding*, 113(4):459–476, 2009.

[36] Liang Wang and Ruigang Yang. Global stereo matching leveraged by sparse ground control points. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3033–3040. IEEE, 2011.

[37] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 807–814, 2005.

[38] Safaa Moqqaddem, Yassine Ruichek, R Touahni, and Abderrahmane Sbihi. A spectral clustering and kalman filtering based objects detection and tracking

using stereo vision with linear cameras. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 902–907. IEEE, 2011.

[39] Safaa Moqqaddem, Y Ruichek, R Touahni, and A Sbihi. Objects detection and tracking using points cloud reconstructed from linear stereo vision. *CURRENT ADVANCEMENTS IN STEREO VISION*, page 161, 2012.

[40] Bernd Kitt, Benjamin Ranft, and Henning Lategahn. Detection and tracking of independently moving objects in urban environments. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1396–1401. IEEE, 2010.

[41] Philip Lenz, Julius Ziegler, Andreas Geiger, and Martin Roser. Sparse scene flow segmentation for moving object detection in urban environments. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 926–932, 2011.

[42] Ashit Talukder and Larry Matthies. Real-time detection of moving objects from moving vehicles using dense stereo and optical flow. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3718–3725. IEEE, 2004.

[43] Andreas Wedel. *Stereo scene flow for 3D motion analysis*. Springerverlag London Limited, 2011.

[44] Ce Liu. *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, Massachusetts Institute of Technology, 2009.

[45] Michael Tao, Jiamin Bai, Pushmeet Kohli, and Sylvain Paris. Simpleflow: A non-iterative, sublinear optical flow algorithm. In *Computer Graphics Forum*, volume 31, pages 345–353. Wiley Online Library, 2012.

[46] Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient large-scale stereo matching. In *Computer Vision–ACCV 2010*, pages 25–38. Springer, 2011.

[47] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.

[48] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[49] Robert M. Haralick. Propagating covariance in computer vision. In *Performance Characterization in Computer Vision*, pages 95–114. Springer, 2000.

[50] John C. Clarke. Modelling uncertainty: A primer. *University of Oxford. Dept. Engineering science, Tech. Rep*, 2161:98, 1998.

[51] Xiaoyan Hu and Philippos Mordohai. A quantitative evaluation of confidence measures for stereo vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2121–2133, 2012.

[52] P.C. Mahalanobis. On the generalised distance in statistics. In *Proceedings of the National Institute of Science of India*, pages 49–55, 1936.

[53] Yuri Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in ND images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112, 2001.

[54] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 309–314, 2004.

[55] Antonio HernÃ¡ndez-Vela, Nadezhda Zlateva, Alexander Marinov, Miguel Reyes, Petia Radeva, Dimo Dimov, and Sergio Escalera. Graph cuts optimization for multi-limb human segmentation in depth maps. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 726–732, 2012.

[56] Xiaoyan Dai. Automatic segmentation fusing color and depth. In *Pattern Recognition (ICPR), 21st International Conference on*, pages 763–766, 2012.

[57] DM Greig, BT Porteous, and Allan H Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 271–279, 1989.

[58] Raphael Labayrade, Didier Aubert, and J-P Tarel. Real time obstacle detection in stereovision on non flat road geometry through" v-disparity" representation. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 646–651. IEEE, 2002.

[59] Zhencheng Hu and Keiichi Uchimura. UV-disparity: an efficient algorithm for stereovision based scene analysis. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 48–54, 2005.

[60] Derek Hoiem, Alexei A Efros, and Martial Hebert. Putting objects in perspective. *International Journal of Computer Vision*, 80(1):3–15, 2008.

[61] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[62] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[63] Ce Liu et al. *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, Massachusetts Institute of Technology, 2009.

[64] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

[65] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[66] Rahul Kumar Namdev, Abhijit Kundu, K Madhava Krishna, and CV Jawahar. Motion segmentation of multiple objects from a freely moving monocular camera. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4092–4099. IEEE, 2012.

[67] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893, 2005.

[68] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. In *Computer Vision–ECCV 2006*, pages 428–441. Springer, 2006.

[69] P. Sabzmeydani and G. Mori. Detecting pedestrians by learning shapelet features. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8, 2007.

[70] S. Paisitkriangkrai, C. Shen, and J. Zhang. Fast pedestrian detection using a cascade of boosted covariance features. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18:1140–1151, 2008.

[71] R. Benenson, M. Omran, J. Hosang, , and B. Schiele. Ten years of pedestrian detection, what have we learned? In *ECCV, CVRSUAD workshop*, 2014.

[72] C. Papageorgiou, T. Evgeniou, and T. Poggio. A trainable pedestrian detection system. In *Proceeding of Intelligent Vehicles*, pages 241–246, October 1998.

[73] D. Gerónimo, A. Sappa, A. López, and D. Ponsa. Adaptive image sampling and windows classification for on-board pedestrian detection. In *Proceedings of the International Conference on Computer Vision Systems, Bielefeld, Germany*, 2007.

[74] David Geronimo, Angel D. Sappa, Daniel Ponsa, and Antonio M. Lopez. 2D-3D-based on-board pedestrian detection system. *Computer Vision and Image Understanding*, 114(5):583–595, 2010.

[75] Arnaud Martin, Hicham Laanaya, and Andreas Arnold-Bos. Evaluation for uncertain image classification and segmentation. *Pattern Recognition*, 39(11):1987–1995, 2006.

[76] Arnaud Martin and Christophe Osswald. Human expert fusion for image classification. *Information Security: An International Journal, Special issue on fusing uncertain, imprecise and paradoxist information (DSmT)*, pages 122–143, 2006.

[77] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[78] Matthew Brown and David G Lowe. Automatic panoramic image stitching using invariant features. *International journal of computer vision*, 74(1):59–73, 2007.

[79] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[80] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):509–522, 2002.

[81] Xiaoyu Wang, Tony X Han, and Shuicheng Yan. An hog-lbp human detector with partial occlusion handling. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 32–39. IEEE, 2009.

[82] Y. Ding and J. Xiao. Contextual boost for pedestrian detection. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2895–2902, 2012.

[83] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2137–2144, 2006.

[84] D. Ramanan. Using segmentation to verify object hypotheses. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8, 2007.

[85] Y. Freund. An adaptive version of the boost by majority algorithm. *Machine learning*, 43:293–318, 2001.

[86] P. Viola and M. J. Jones. Robust real-time face detection. *International journal of computer vision*, 57:137–154, 2004.

[87] Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio. Example-based object detection in images by components. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(4):349–361, 2001.

[88] Krystian Mikolajczyk, Cordelia Schmid, and Andrew Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *Computer Vision-ECCV 2004*, pages 69–82. Springer, 2004.

[89] Bo Wu and Ram Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266, 2007.

[90] Lubomir Bourdev and Jitendra Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1365–1372. IEEE, 2009.

[91] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[92] Laurens JP van der Maaten, Eric O Postma, and H Jaap van den Herik. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, 10(1-41):66–71, 2009.

[93] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.

[94] Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–506. IEEE, 2004.

[95] Takuya Kobayashi, Akinori Hidaka, and Takio Kurita. Selection of histograms of oriented gradients features for pedestrian detection. In *Neural Information Processing*, pages 598–607. Springer, 2008.

[96] Chengbin Zeng and Huadong Ma. Robust head-shoulder detection by pca-based multilevel hog-lbp detector for people counting. In *Pattern Recognition (ICPR), 2010 International Conference on*, pages 2069–2072. IEEE, 2010.

[97] Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[98] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37:297–336, 1999.

[99] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28:337–407, 2000.

[100] Stan Z Li and ZhenQiu Zhang. Floatboost learning and statistical face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1112–1123, 2004.

[101] A. Ferreira. Survey on boosting algorithms for supervised and semi-supervised learning. Technical report, Technical Report, Instituto Superior de Engenharia de Lisboa, 2007.

[102] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001.*, volume 1, pages 5–11. IEEE, 2001.

[103] Qiang Zhu, M-C Yeh, Kwang-Ting Cheng, and Shai Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1491–1498. IEEE, 2006.

[104] Paul Viola, Michael J Jones, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161, 2005.

[105] Qixiang Ye, Jianbin Jiao, and Baochang Zhang. Fast pedestrian detection with multi-scale orientation features and two-stage classifiers. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 881–884. IEEE, 2010.

[106] Vladimir Vapnik. *The nature of statistical learning theory*. springer, 2000.

[107] M. Enzweiler, A. Eigenstetter, B. Schiele, and D. M. Gavrila. Multi-cue pedestrian classification with partial occlusion handling. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 990–997, 2010.

[108] Markus Enzweiler and Dariu M Gavrila. Integrated pedestrian classification and orientation estimation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 982–989. IEEE, 2010.

[109] Markus Enzweiler and Dariu M Gavrila. A multilevel mixture-of-experts framework for pedestrian classification. *Image Processing, IEEE Transactions on*, 20(10):2967–2979, 2011.

[110] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[111] M. Szarvas, A. Yoshizawa, M. Yamamoto, and J. Ogata. Pedestrian detection with convolutional neural networks. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 224–229, 2005.

[112] S. Munder and D. M. Gavrila. An experimental study on pedestrian classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28:1863–1868, 2006.

[113] I. P. Alonso, D. F. Llorca, M. A Sotelo, L. M. Bergasa, P. R. de Toro, J. Nuevo, M. Ocana, and M. A. G. Garrido. Combination of feature extraction methods for svm pedestrian detection. *Intelligent Transportation Systems, IEEE Transactions on*, 8:292–307, 2007.

[114] Subhransu Maji, Alexander C Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[115] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3626–3633. IEEE, 2013.

[116] Wanli Ouyang and Xiaogang Wang. A discriminative deep model for pedestrian detection with occlusion handling. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3258–3265. IEEE, 2012.

[117] Wanli Ouyang, Xingyu Zeng, and Xiaogang Wang. Modeling mutual visibility relationship in pedestrian detection. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3222–3229. IEEE, 2013.

[118] Wanli Ouyang and Xiaogang Wang. Joint deep learning for pedestrian detection. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2056–2063. IEEE, 2013.

[119] Ping Luo, Yonglong Tian, Xiaogang Wang, and Xiaoou Tang. Switchable deep network for pedestrian detection. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 899–906. IEEE, 2014.

[120] Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, et al. *Semi-supervised learning*, volume 2. MIT press Cambridge, 2006.

[121] Xiaojin Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2:3, 2006.

[122] Xiaojin Zhu, Andrew B Goldberg, Ronald Brachman, and Thomas Dietterich. Introduction to semi-supervised learning. 2009.

[123] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209, 1999.

[124] Shai Ben-David, Tyler Lu, and Dávid Pál. Does unlabeled data provably help? worst-case analysis of the sample complexity of semi-supervised learning. In *COLT*, pages 33–44, 2008.

[125] Larry Wasserman and John D Lafferty. Statistical analysis of semi-supervised regression. In *Advances in Neural Information Processing Systems*, pages 801–808, 2007.

[126] Robert E. Schapire, Marie Rochery, Mazin Rahim, and Narendra Gupta. Boosting with prior knowledge for call classification. *Speech and Audio Processing, IEEE Transactions on*, 13:174–181, 2005.

[127] Amir Saffari, Helmut Grabner, and Horst Bischof. Serboost: Semi-supervised boosting with expectation regularization. In *Computer Vision-ECCV 2008*, pages 588–601. Springer, 2008.

[128] Pavan Kumar Mallapragada, Rong Jin, Anil K. Jain, and Yi Liu. Semiboost: Boosting for semi-supervised learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(11):2000–2014, 2009.

[129] Ke Chen and Shihai Wang. Semi-supervised learning via regularized boosting working on multiple semi-supervised assumptions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(1):129–143, 2011.

[130] Ke Chen and Shihai Wang. Regularized boost for semi-supervised learning. In *Advances in neural information processing systems*, pages 281–288, 2008.

[131] Llew Mason, Jonathan Baxter, Peter L. Bartlett, and Marcus Frean. Functional gradient techniques for combining hypotheses. *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 221–246, 1999.

[132] Christian Leistner, Helmut Grabner, and Horst Bischof. Semi-supervised boosting using visual similarity learning. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, 2008.

[133] Lei Zheng, Shaojun Wang, Yan Liu, and Chi-Hoon Lee. Information theoretic regularization for semi-supervised boosting. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1017–1026. ACM, 2009.

[134] Takafumi Kanamori and Takashi Takenouchi. Improving logitboost with prior knowledge. *Information Fusion*, 14(2):208–219, 2013.

[135] Weihong Wang, Yang Wang, Fang Chen, and Arcot Sowmya. A weakly supervised approach for object detection based on soft-label boosting. In *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*, pages 331–338. IEEE, 2013.

[136] Jeff A Bilmes et al. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 4(510):126, 1998.

[137] Zhi-Hua Zhou and Ming Li. Tri-training: Exploiting unlabeled data using three classifiers. *Knowledge and Data Engineering, IEEE Transactions on*, 17(11):1529–1541, 2005.

[138] J Ross Quinlan. Learning efficient classification procedures and their application to chess end games. In *Machine learning*, pages 463–482. Springer, 1983.

[139] Paula Gabbert. Decision trees from uncertain learning sets. In *Systems, Man, and Cybernetics, 1994. Humans, Information and Technology., 1994 IEEE International Conference on*, volume 1, pages 913–918. IEEE, 1994.

[140] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[141] Douglas Reynolds. Gaussian mixture models. *Encyclopedia of Biometrics*, pages 659–663, 2009.

[142] Richard A Redner and Homer F Walker. Mixture densities, maximum likelihood and the em algorithm. *SIAM review*, 26(2):195–239, 1984.

[143] David Gerónimo, Angel D Sappa, Daniel Ponsa, and Antonio M López. 2d–3d-based on-board pedestrian detection system. *Computer Vision and Image Understanding*, 114(5):583–595, 2010.

[144] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[145] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 722–729. IEEE, 1999.

[146] Ye Zhang and Chandra Kambhamettu. Integrated 3d scene flow and structure recovery from multiview image sequences. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 674–681. IEEE, 2000.

[147] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3d scene flow estimation with a rigid motion prior. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1291–1298. IEEE, 2011.

[148] Christoph Vogel, Stefan Roth, and Konrad Schindler. View-consistent 3d scene flow estimation over multiple frames. In *Computer Vision–ECCV 2014*, pages 263–278. Springer, 2014.

[149] Michael D Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier, and Luc Van Gool. Robust tracking-by-detection using a detector confidence particle filter. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1515–1522. IEEE, 2009.

[150] P. Smets. The transferable belief model and random sets. *International Journal of Intelligent Systems*, 7:37–46, 1992.

[151] Z. Elouedi, K. Mellouli, and P. Smets. Belief decision trees: theoretical foundations. *International Journal of Approximate Reasoning*, 28:91–124, 2001.

[152] Benjamin Quost and Thierry Denœux. Learning from data with uncertain labels by boosting credal classifiers. In *Proceedings of the 1st ACM SIGKDD Workshop on Knowledge Discovery From Uncertain Data*, pages 38–47. ACM, 2009.

[153] Arnaud Martin and Christophe Osswald. Experts fusion and multilayer perceptron based on belief learning for sonar image classification. In *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, pages 1–6. IEEE, 2008.

[154] Arnaud Martin. Reliability and combination rule in the theory of belief functions. In *Information Fusion, 2009. FUSION'09. 12th International Conference on*, pages 529–536. IEEE, 2009.

[155] Arnaud Martin, Christophe Osswald, Jean Dezert, and Florentin Smarandache. General combination rules for qualitative and quantitative beliefs. *Journal of Advances in Information Fusion*, 3(2):67–82, 2008.

[156] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

[157] Myron Z Brown, Darius Burschka, and Gregory D Hager. Advances in computational stereo. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(8):993–1008, 2003.

[158] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 519–528. IEEE, 2006.

[159] Zeng-Fu Wang and Zhi-Gang Zheng. A region based stereo matching algorithm using cooperative optimization. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[160] Engin Tola, Vincent Lepetit, and Pascal Fua. A fast local descriptor for dense matching. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[161] Ke Zhang, Jiangbo Lu, and Gauthier Lafruit. Cross-based local stereo matching using orthogonal integral images. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(7):1073–1079, 2009.

[162] Asmaa Hosni, Michael Bleyer, Margrit Gelautz, and Christoph Rhemann. Local stereo matching using geodesic support weights. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 2093–2096. IEEE, 2009.

[163] Geert Van Meerbergen, Maarten Vergauwen, Marc Pollefeys, and Luc Van Gool. A hierarchical symmetric stereo algorithm using dynamic programming. *International Journal of Computer Vision*, 47(1-3):275–285, 2002.

[164] Michael Bleyer and Margrit Gelautz. Simple but effective tree structures for dynamic programming-based stereo matching. In *VISAPP (2)*, pages 415–422, 2008.

[165] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 508–515. IEEE, 2001.

[166] Li Hong and George Chen. Segment-based stereo matching using graph cuts. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–74. IEEE, 2004.

[167] Jian Sun, Nan-Ning Zheng, and Heung-Yeung Shum. Stereo matching using belief propagation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(7):787–800, 2003.

[168] Qingxiong Yang, Liang Wang, Ruigang Yang, Henrik Stewénius, and David Nistér. Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(3):492–504, 2009.

[169] Stefan K Gehrig and Clemens Rabe. Real-time semi-global matching on the cpu. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 85–92. IEEE, 2010.

[170] Qingxiong Yang, Liang Wang, Ruigang Yang, Shengnan Wang, Miao Liao, and David Nister. Real-time global stereo matching using hierarchical belief propagation. In *BMVC*, volume 6, pages 989–998, 2006.

[171] Martin Humenberger, Christian Zinner, Michael Weber, Wilfried Kubinger, and Markus Vincze. A fast stereo matching algorithm suitable for embedded real-time systems. *Computer Vision and Image Understanding*, 114(11):1180–1202, 2010.

[172] Berthold KP Horn and Brian G. Schunck. Determining optical flow. *Artificial intelligence*, 17:185–203, 1981.

[173] John L Barron, David J Fleet, and Steven S Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12(1):43–77, 1994.

[174] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM Computing Surveys (CSUR)*, 27(3):433–466, 1995.

[175] Joachim Weickert, Andrés Bruhn, Thomas Brox, and Nils Papenberg. A survey on variational optic flow methods for small displacements. In *Mathematical models for registration and applications to medical imaging*, pages 103–136. Springer, 2006.

[176] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 2:3, 2001.

[177] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Pattern Recognition*, pages 214–223. Springer, 2007.

[178] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432–2439. IEEE, 2010.

[179] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[180] Luigi Di Stefano, Stefano Mattoccia, and Federico Tombari. Zncc-based template matching using bounded partial correlation. *Pattern recognition letters*, 26(14):2129–2134, 2005.