



# Stereo vision and LIDAR based Dynamic Occupancy Grid mapping: Application to scenes analysis for Intelligent Vehicles

You Li

## ► To cite this version:

You Li. Stereo vision and LIDAR based Dynamic Occupancy Grid mapping: Application to scenes analysis for Intelligent Vehicles. Computers and Society [cs.CY]. Université de Technologie de Belfort-Montbéliard, 2013. English. NNT : 2013BELF0225 . tel-00982325

**HAL Id: tel-00982325**

**<https://theses.hal.science/tel-00982325>**

Submitted on 23 Apr 2014

**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.



# SPIM

## Thèse de Doctorat




école doctorale sciences pour l'ingénieur et microtechniques

UNIVERSITÉ DE TECHNOLOGIE BELFORT-MONTBÉLIARD

# Stereo Vision and Lidar based Dynamic Occupancy Grid Mapping

Application to Scene Analysis for Intelligent Vehicles

 You LI



# SPIM

## Thèse de Doctorat



école doctorale sciences pour l'ingénieur et microtechniques  
UNIVERSITÉ DE TECHNOLOGIE BELFORT-MONTBÉLIARD

N° 2 | 2 | 5

THÈSE présentée par

You LI

pour obtenir le

Grade de Docteur de

l'Université de Technologie de Belfort-Montbéliard

Spécialité : **Informatique**

# Stereo Vision and Lidar based Dynamic Occupancy Grid Mapping

Application to Scene Analysis for Intelligent Vehicles

Soutenue publiquement le 03 December 2013 devant le Jury composé de :

SERGIU NEDEVSKI	Rapporteur	Professeur à Technical University of Cluj-Napoca (Roumanie)
MICHEL DEVY	Rapporteur	Directeur de Recherche CNRS à LAAS-CNRS de Toulouse
HANZI WANG	Rapporteur	Professeur à Xiamen University (Chine)
VINCENT FREMONT	Examineur	Maître de Conférences HDR à Université de Technologie de Compiègne
JEAN-CHARLES NOYER	Examineur	Professeur à Université du Littoral Côte d'Opale
OLIVIER AYCARD	Examineur	Maître de Conférences HDR à Université de Grenoble 1
CINDY CAPPELLE	Examineur	Maître de Conférences à Université de Technologie de Belfort-Montbéliard
YASSINE RUICHEK	Directeur de thèse	Professeur à Université de Technologie de Belfort-Montbéliard



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	Problem Statement . . . . .	3
1.3	Experimental Platform . . . . .	5
1.4	Structure of the manuscript . . . . .	5
<b>2</b>	<b>Basic Knowledge</b>	<b>7</b>
2.1	Sensor Models . . . . .	7
2.1.1	Coordinate Systems . . . . .	7
2.1.2	Lidar Measurement Model . . . . .	8
2.1.3	Monocular Camera Measurement Model . . . . .	9
2.1.4	Binocular Stereo Vision System Model . . . . .	13
2.2	Stereo Vision System Calibration . . . . .	14
2.2.1	Intrinsic Calibration of a camera . . . . .	14
2.2.2	Extrinsic Calibration of Binocular Vision System . . . . .	14
2.2.3	Image Undistortion and Stereo Rectification . . . . .	15
2.2.4	Corner Points Triangulation . . . . .	17
2.3	Least Squares Estimation . . . . .	19
2.3.1	Linear Least Squares . . . . .	20
2.3.2	Non-linear Least Squares . . . . .	25
2.3.3	Robust Estimation methods . . . . .	28
2.4	Image Local Feature Detectors and Descriptors . . . . .	30
2.4.1	Local Invariant Feature Detectors . . . . .	30
2.4.2	Feature descriptors . . . . .	39
2.4.3	Associating feature points through images . . . . .	43
2.5	Conclusion . . . . .	46
<b>3</b>	<b>Stereo Vision based Ego-motion Estimation</b>	<b>47</b>
3.1	Introduction . . . . .	48
3.2	Circular Feature Points Detection and Association . . . . .	50
3.2.1	Circular Feature Point Association by KLT Tracker . . . . .	51
3.2.2	Circular Feature Point Association by Matching . . . . .	51
3.3	Ego-motion Computation . . . . .	54
3.3.1	3D-2D Constraint based Ego-motion Estimation . . . . .	54
3.4	Experiments of Stereo Visual Odometry . . . . .	57
3.4.1	Comparing Different Feature Association Approaches . . . . .	57
3.4.2	Stereo Visual Odometry in Urban Environment . . . . .	62
3.5	Conclusion and Future Works . . . . .	70

<b>4</b>	<b>Independent Moving Object Detection, Segmentation and Recognition</b>	<b>71</b>
4.1	Independent Moving Object Detection and Segmentation . . . . .	71
4.1.1	Introduction . . . . .	71
4.1.2	UV-Disparity Based Independent Moving Objectxs Detection and Segmentation . . . . .	73
4.1.3	Experimental results of U-disparity image based Independent Moving Object Detection . . . . .	77
4.2	Moving Object Recognition using Spatial Information . . . . .	80
4.2.1	Introduction . . . . .	80
4.2.2	Spatial Feature Extraction and Classification . . . . .	81
4.2.3	Experimental Results . . . . .	84
4.3	Conclusion and Future Works . . . . .	89
<b>5</b>	<b>Extrinsic Calibration between a Stereo Vision System and a Lidar</b>	<b>91</b>
5.1	Introduction . . . . .	91
5.1.1	Related works . . . . .	92
5.1.2	Problem Formulation . . . . .	93
5.1.3	Performance Evaluation . . . . .	94
5.2	3D Plane Reconstruction based Extrinsic Calibration . . . . .	95
5.2.1	Corner Points Triangulation . . . . .	96
5.2.2	3D Plane Estimation . . . . .	96
5.2.3	Automatic Lidar Measurements Extraction . . . . .	100
5.2.4	Estimating Rigid Transformation between the lidar and the Stereoscopic System . . . . .	101
5.2.5	Summary of the Calibration Procedure . . . . .	103
5.3	Experimental Results . . . . .	104
5.3.1	Computer Simulations . . . . .	104
5.3.2	Real Data Test . . . . .	110
5.3.3	Outdoor Experiments . . . . .	111
5.4	Conclusion And Future Work . . . . .	116
<b>6</b>	<b>Occupancy Grid Mapping of Environments</b>	<b>117</b>
6.1	Occupancy Grid Mapping by Stereo Vision . . . . .	120
6.1.1	Introduction . . . . .	120
6.1.2	Foundations . . . . .	121
6.1.3	Ground Plane Analysis . . . . .	122
6.1.4	Building Occupancy Grid Map . . . . .	128
6.1.5	Experimental Results of Stereo Vision based Dynamic Occupancy Grid Mapping . . . . .	131
6.2	Occupancy Grid Mapping by Lidar . . . . .	136
6.2.1	Introduction . . . . .	136
6.2.2	Ray Casting . . . . .	136
6.2.3	Lidar Inverse Model . . . . .	137

---

6.2.4	Experimental Results . . . . .	139
6.3	Occupancy Grid Mapping by Fusing Lidar and Stereo Vision . . . .	142
6.3.1	Introduction . . . . .	142
6.3.2	Fusing by Linear Opinion Pool . . . . .	142
6.3.3	Experimental Results . . . . .	144
6.4	Conclusion and Future Works . . . . .	148
<b>7</b>	<b>Conclusions and Future Works</b>	<b>149</b>
7.1	Conclusions . . . . .	149
7.2	Future Works . . . . .	150
<b>A</b>	<b>Publications</b>	<b>153</b>
	<b>Bibliography</b>	<b>155</b>





# Introduction

---

## Contents

<b>1.1</b>	<b>Background . . . . .</b>	<b>1</b>
<b>1.2</b>	<b>Problem Statement . . . . .</b>	<b>3</b>
<b>1.3</b>	<b>Experimental Platform . . . . .</b>	<b>5</b>
<b>1.4</b>	<b>Structure of the manuscript . . . . .</b>	<b>5</b>

---

## 1.1 Background

During the past decades, due to the reason of road safety, advanced driver assistance systems (ADAS) or intelligent vehicles (IV) have obtained more and more attentions and developments from research society and industry community. Advanced driver assistance systems (ADAS), are systems to help driver in the driving tasks. ADAS usually consists of *adaptive cruise control (ACC)*, *lane departure warning system*, *collision avoidance system*, *automatic parking*, *traffic sign recognition*, etc. ADAS provides relatively basic control assistance by sensing the environment or assessing risks. Nowadays, ADAS have already been applied in some top-brand vehicles, e.g. Mercedes-Benz E-class <sup>1</sup>.

Intelligent vehicle (IV) systems are seen as a next generation beyond current ADAS. IV systems sense the driving environment and provide information or vehicle control to assist the driver in optimal vehicle operation. The range of applications for IV systems are quite broad and are applied to all types of road vehicles – cars, heavy trucks, and transit buses. In [Bishop 2005], IV applications can generally be classified into four categories: *convenience systems*, *safety systems*, *productivity systems* and *traffic-assist systems*, which are listed as follows:

- **Convenience Systems**
  - Parking Assistance
  - Adaptive Cruise Control (ACC)
  - Lane-Keeping Assistance
  - Automated Vehicle Control

---

<sup>1</sup><http://telematicsnews.info/2013/01/16/naias-mercedes-benz-e-class-fitted-with-multiple-adasj4162/>

- **Safety Systems**

- Assisting driver perception
- Crash prevention
- Degraded driving
- Precrash

- **Productivity Systems**

- Truck Applications
- Transit Bus Applications

- **Traffic-Assistance Systems**

- Vehicle Flow Management (VFM)
- Traffic-Responsive Adaptation
- Traffic Jam Dissipation
- Platooning

In order to stimulate the development of intelligent vehicles, American Department of Defence has organized three autonomous driving competitions (DARPA Grand Challenge in 2004 <sup>2</sup> and 2005 <sup>3</sup>, DARPA Urban Challenge in 2007 <sup>4</sup>. Not coincidentally, Chinese government has organized similar intelligent vehicle competitions since 2009 <sup>5</sup>. Recently, Google released its first driverless car in May 2012 <sup>6</sup>. Up to September 2012, three U.S. states (Nevada, Florida and California) have passed laws permitting driverless cars. Examples of intelligent vehicle prototypes from DARPA competitions and Google are shown in Fig. 1.1.

---

<sup>2</sup><http://archive.darpa.mil/grandchallenge04/>

<sup>3</sup><http://archive.darpa.mil/grandchallenge05/>

<sup>4</sup><http://archive.darpa.mil/grandchallenge>

<sup>5</sup><http://baike.baidu.com/view/4572422.htm>

<sup>6</sup>[http://en.wikipedia.org/wiki/Google\\_driverless\\_car](http://en.wikipedia.org/wiki/Google_driverless_car)



(a) MIT intelligent vehicle in DARPA 2007  
(from [grandchallenge.mit.edu](http://grandchallenge.mit.edu))



(b) Stanford intelligent vehicle in DARPA 2007  
(from [cs.stanford.edu/group/roadrunner](http://cs.stanford.edu/group/roadrunner))



(c) Google driverless car (from <http://gas2.org>)

Figure 1.1: Intelligent vehicles joined in DARPA grand challenge 2007 and Google driverless car

## 1.2 Problem Statement

In intelligent vehicles, perception systems are of the most importance. Perception systems could sense and interpret surrounding environment based on various kinds of sensors, such as radar, 2D/3D lidar (laser range finder), monocular/binocular/omnidirectional vision systems, etc. Indeed, as described in [Fletcher 2008, Petrovskaya 2009], all the experimental vehicles joined in 2007 DARPA grand challenge were equipped with advanced sensors (shown in Fig. 1.1). In most cases, these sensors shoulder heavy responsibilities of environmental perception (self-localization, collision avoidance, motion planning, etc.).

Towards the objective of environmental perception, stereo vision system and lidar are two conventional sensors. In general, stereo vision system is a passive sensor system consisting of at least two cameras. It offers abundant texture and potential semantic information of surrounding environments, and it is superior in 3D measuring at a relative low expense. Many perceptual tasks could be solved by stereo vision system, e.g. visual SLAM [Davision 2007], visual detection and tracking [Srinivasa 2002], visual classification and recognition [Gavrila 2004]. On the other side, lidar is an active range sensor which efficiently measures surrounding distances by visible/invisible laser. The effective detection range of lidar is always larger than

stereo vision system. Merely based on lidars, a great number of techniques are also developed for scene perception and understanding, e.g. [Streller 2002, Cole 2006]. In this thesis, we focus on the usages of stereo vision system and lidar.

This thesis is part of the project CPER "Intelligence du Véhicule Terrestre" (Intelligence of ground vehicle), developed within Systems and Transportation Laboratory (SeT) of Institute of Research on Transportation, Energy and Society (IRTES), UTBM, France. The problem addressed in this thesis is how to map dynamic environment by stereo vision and lidar. The main objectives and contributions are as follows:

- The first objective is to estimate the ego-motion of a vehicle when it is driven. Ego-motion estimation is a fundamental part for an intelligent vehicle. Only by knowing the ego-motion of the vehicle itself at first, an intelligent vehicle can further analyze the surrounding dynamic environment. To achieve this objective, we implement a stereo vision based ego-motion estimation method. Our contribution is a comparison between tracker based and matching based circular feature associations as well as performance analysis of image feature detectors. From the comparison, the best image feature detector and feature association approach are selected for estimating ego-motion of the vehicle.
- The second objective is to detect and recognize independent moving objects around the moving vehicle. This work is important for collision avoidance. We propose a stereo vision based independent moving object detection method, as well as a spatial information based object recognition method. The independent moving objects are detected based on the results of ego-motion estimation, with a help of U-disparity map. The recognition is performed by training classifiers from spatial information of the moving objects.
- The third objective is extrinsic calibration between a stereoscopic system and 2D lidar. The extrinsic calibration between the two sensors is to calculate a rigid 3D transformation, which connects the positions of the two sensors. We use an ordinary chessboard to achieve the extrinsic calibration and intrinsic calibration of the stereoscopic system at the same time. Also, we present an optimal extrinsic calibration of the stereoscopic system and the lidar basing on geometric constraints that use 3D planes reconstructed from stereo vision.
- The fourth objective is mapping the sensed environment by occupancy grids. Occupancy grid map is a classic but practical tool of interpreting environment. In the thesis, we propose a stereo vision based occupancy grid mapping method. Our contribution is about the improvement of mapping results by analyzing the ground plane and augmenting the mapping by integrating motion and object recognition information. Then, we attempt to combine stereo vision and lidar to improve the occupancy grid map.

### 1.3 Experimental Platform

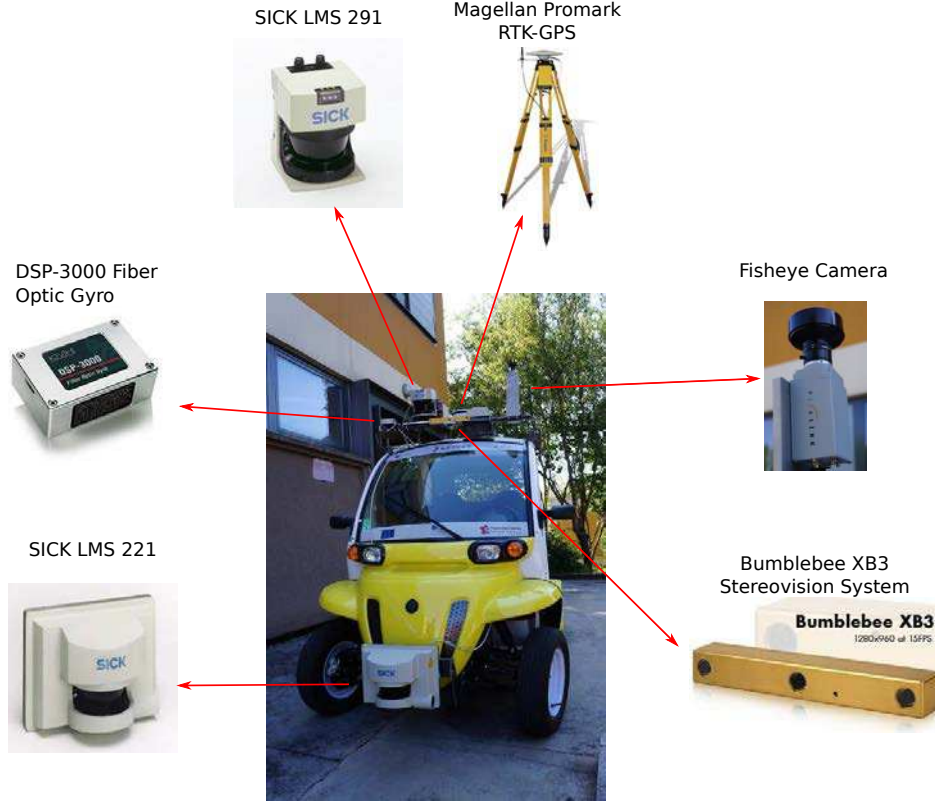


Figure 1.2: Experimental platform: SetCar developed by IRTES-SET laboratory in UTBM

As demonstrated in Fig. 1.2, our platform SetCar is equipped by many sensors (2D lidar, stereo vision system, fisheye camera, RTK-GPS, gyro, etc). A Bumblebee XB3 stereo vision system is installed on the top, and oriented to the front. It contains three collinear cameras with a maximum baseline of 240mm. Images are captured in a format of  $1280 \times 960$  pixels and a speed of 15 fps (frames per second). In our application, only the left and right cameras are used. A SICK LMS 221/291 lidar provides range measurements in a  $180^\circ$  scanning plane, with an angular resolution of  $0.5^\circ$ . It scans in 75Hz and 80 meters as maximum detective range. All of the installed sensors are fixed rigidly.

### 1.4 Structure of the manuscript

In chapter 2, we review basic knowledges used in the thesis. The basic knowledges contain the measurement models of stereo vision system and lidar, linear/nonlinear least square methods and image feature detectors/descriptors. The reviewed basic knowledges make understanding the thesis more easy.

In chapter 3, we first summarize a framework of stereo visual odometry. Within the framework, different kinds of feature detection and association are then compared at length. From the comparison results, the best approaches of detecting and associating image features are selected.

In chapter 4, independent moving objects are firstly detected and segmented based on the results of ego-motion estimation described in chapter 3. Next, spatial information of the segmented objects are extracted based on a kernel PCA method. Several classifiers are learned from these spatial information for achieving independent moving object recognition.

In chapter 5, we propose a chessboard based extrinsic calibration between a stereoscopic system and a 2D lidar. To improve the accuracy of calibration results, we model the sensor measurements and take the sensor models into account. We show that by considering the sensor models, the calibration results are improved compared with a previous method.

In chapter 6, dynamic occupancy grid map is built firstly by stereo vision system. It integrates motion and object recognition information. The mapping process relies on 3D reconstruction of the stereo measures and is improved by analyzing geometrical feature of ground plane. At the same time, occupancy grid map is also created by lidar measures. By the extrinsic calibration results in chapter 5, we adopt a linear opinion pool to combine the two kinds of measures to create a occupancy grid map.

In chapter 7, conclusions and some research perspectives for this thesis are presented.

# Basic Knowledge

## Contents

<b>2.1</b>	<b>Sensor Models</b>	<b>7</b>
2.1.1	Coordinate Systems	7
2.1.2	Lidar Measurement Model	8
2.1.3	Monocular Camera Measurement Model	9
2.1.4	Binocular Stereo Vision System Model	13
<b>2.2</b>	<b>Stereo Vision System Calibration</b>	<b>14</b>
2.2.1	Intrinsic Calibration of a camera	14
2.2.2	Extrinsic Calibration of Binocular Vision System	14
2.2.3	Image Undistortion and Stereo Rectification	15
2.2.4	Corner Points Triangulation	17
<b>2.3</b>	<b>Least Squares Estimation</b>	<b>19</b>
2.3.1	Linear Least Squares	20
2.3.2	Non-linear Least Squares	25
2.3.3	Robust Estimation methods	28
<b>2.4</b>	<b>Image Local Feature Detectors and Descriptors</b>	<b>30</b>
2.4.1	Local Invariant Feature Detectors	30
2.4.2	Feature descriptors	39
2.4.3	Associating feature points through images	43
<b>2.5</b>	<b>Conclusion</b>	<b>46</b>

## 2.1 Sensor Models

As demonstrated in Fig. 1.2, our platform SetCar is equipped by many sensors (2D lidar, stereo vision system, fisheye camera, RTK-GPS, gyro, etc). In this section, we introduce the sensor measuring models of lidar and stereo vision system respectively.

### 2.1.1 Coordinate Systems

To analyze such a multi-sensor system, we set several coordinate systems with respect to the different sensors, as demonstrated in Fig. 2.1. Let  $\mathbb{R}_{stereo}^3$ ,  $\mathbb{R}_{left}^3$ ,  $\mathbb{R}_{right}^3$ , and  $\mathbb{R}_{lidar}^3$ , be the coordinate systems attached to the stereo vision system, the



left and right cameras, and the lidar respectively. All the coordinate systems are connected with each other by a rigid Euclidean transformation (a rotation matrix  $R$  and a translation vector  $T$ ). For the reason of simplicity, we assume the left camera coordinate system as the reference of the stereo vision coordinate system, hence,  $\mathbb{R}^3_{stereo} = \mathbb{R}^3_{left}$ . After setting up all the sensor coordinate systems and their connections, we are going to introduce measurement models of these sensors.

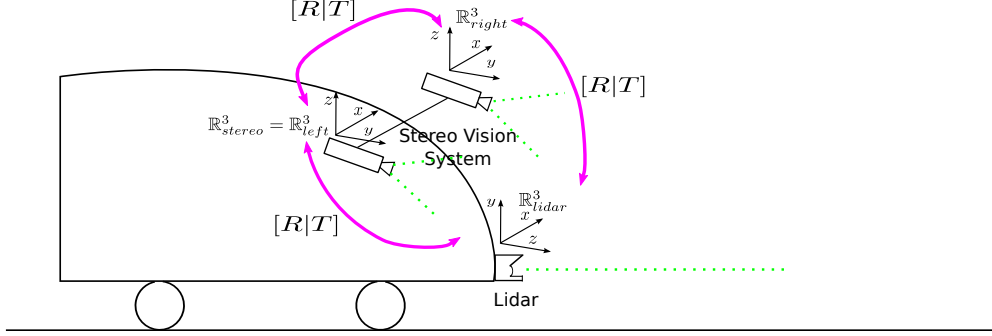


Figure 2.1: Coordinate systems in our platform

### 2.1.2 Lidar Measurement Model

In the lidar coordinate system, we specify the origin as the point which emits laser rays. The directions  $X, Y, Z$  are set as rightward, upward and forward from the sensor respectively, as shown in Fig. 2.1. The lidar we used (SICK LMS 221/291) provides an array of raw distance measurements  $r_i, \theta_i$  in polar coordinate system. In real applications, these raw measurements are converted into Cartesian form  $(X_i \ Y_i \ Z_i)^T$  (note that in our assumption,  $Y_i$  always equals to zero):

$$\begin{bmatrix} X_i \\ Z_i \end{bmatrix} = \begin{bmatrix} r_i \times \cos \theta_i \\ r_i \times \sin \theta_i \end{bmatrix} \quad (2.1)$$

**Error Model:** In most cases, lidar measurements contain errors, as described in the manual of SICK LMS 221/291<sup>1</sup>. According to this manual, we assume the lidar measurements, ranges  $r_i$  and scan angles  $\theta_i$ , have additive Gaussian noises:

$$\begin{aligned} r_i &= R_i + n_r \\ \theta_i &= \Theta_i + n_\theta \end{aligned} \quad (2.2)$$

where  $R_i$  and  $\Theta_i$  are the "true" measurements.  $n_r$  and  $n_\theta$  are independent additive zero-mean Gaussian noises with variances  $\delta_r$  and  $\delta_\theta$ , respectively. Hence, after being converted into Cartesian form, we have:

$$\begin{aligned} \begin{bmatrix} X_i \\ Z_i \end{bmatrix} &= (R_i + n_r) \begin{bmatrix} \cos(\Theta_i + n_\theta) \\ \sin(\Theta_i + n_\theta) \end{bmatrix} = (R_i + n_r) \begin{bmatrix} \cos \Theta_i \cos n_\theta - \sin \Theta_i \sin n_\theta \\ \sin \Theta_i \cos n_\theta + \cos \Theta_i \sin n_\theta \end{bmatrix} \\ &= \bar{\mathbf{P}}^i_{lidar} + \mathbf{n}_p \end{aligned} \quad (2.3)$$

<sup>1</sup><http://sicktoolbox.sourceforge.net/docs/sick-lms-technical-description.pdf>

where  $\tilde{\mathbf{P}}_{lidar}^i$  is the "true" measurement in 3D Cartesian coordinate,  $\mathbf{n}_p$  is the variance vector. When assuming  $n_\theta \ll 1$  (which is the fact in most lidar equipments), hence,  $\sin n_\theta \sim n_\theta$ ,  $\cos n_\theta \sim 1$ . Eq. (2.3) yields:

$$\mathbf{n}_p = (R_i + n_r)n_\theta \begin{bmatrix} -\sin \Theta_i \\ \cos \Theta_i \end{bmatrix} + n_r \begin{bmatrix} \cos \Theta_i \\ \sin \Theta_i \end{bmatrix} \quad (2.4)$$

From the above, the covariance matrix for a LIDAR measurement is:

$$\begin{aligned} Q_{lidar}^i &\triangleq E[\mathbf{n}_p \mathbf{n}_p^T] \\ &= \frac{(R_i)^2 \delta_\theta^2}{2} \begin{bmatrix} 2 \sin^2 \Theta_i & -\sin 2\Theta_i \\ -\sin 2\Theta_i & 2 \cos^2 \Theta_i \end{bmatrix} + \frac{\delta_r^2}{2} \begin{bmatrix} 2 \cos^2 \Theta_i & \sin 2\Theta_i \\ \sin 2\Theta_i & 2 \sin^2 \Theta_i \end{bmatrix} \end{aligned} \quad (2.5)$$

where  $Q_{lidar}^i$  is the covariance matrix of the  $i$ th lidar measurement  $\mathbf{P}_{lidar}^i$ .  $\delta_r$  and  $\delta_\theta$  are the variances of independent additive zero-mean Gaussian noises  $n_r$  and  $n_\theta$ . In practice,  $\Theta_i$  and  $R_i$  are approximated by  $\theta_i$  and  $r_i$ , respectively. Thus, in homogeneous coordinates, the covariance matrix of  $\tilde{\mathbf{P}}_{lidar}^i$  can be expressed as:

$$\Sigma_{\tilde{\mathbf{P}}_{lidar}^i} = \begin{bmatrix} Q_{11}^i & 0 & Q_{12}^i \\ 0 & 0 & 0 \\ Q_{21}^i & 0 & Q_{22}^i \\ & & & 0 \end{bmatrix} \quad (2.6)$$

where  $Q_{kj}^i$  corresponds to the  $(k, j)$ th element in Eq. (2.5). The error transformation is shown in Fig. 2.2

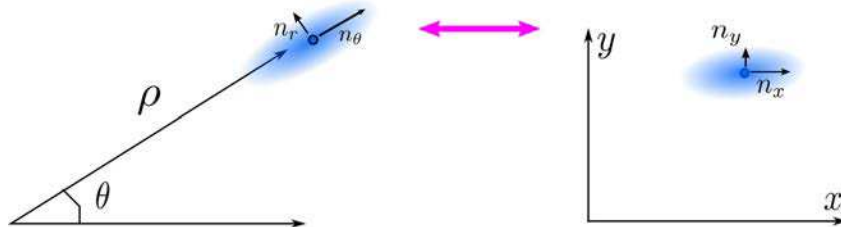


Figure 2.2: Error transformation from polar coordinate system to Cartesian coordinate system

### 2.1.3 Monocular Camera Measurement Model

#### 2.1.3.1 Pinhole Camera Model

The imaging process for an ordinary CCD/CMOS camera is a transformation from photons into electrons. For most cameras with CCD/CMOS like sensors, pinhole model is a basic but useful model to present this mapping. A typical pinhole camera model is shown in Fig. 2.3. Let *camera center*  $O$  be the origin of camera coordinate system  $\mathbb{R}_{camera}^3$ , and consider the image coordinate system  $\mathbb{R}_{image}^2$ , which locates in the *image plane*  $Z = f$ , where  $f$  is *focal length*, the distance between the camera

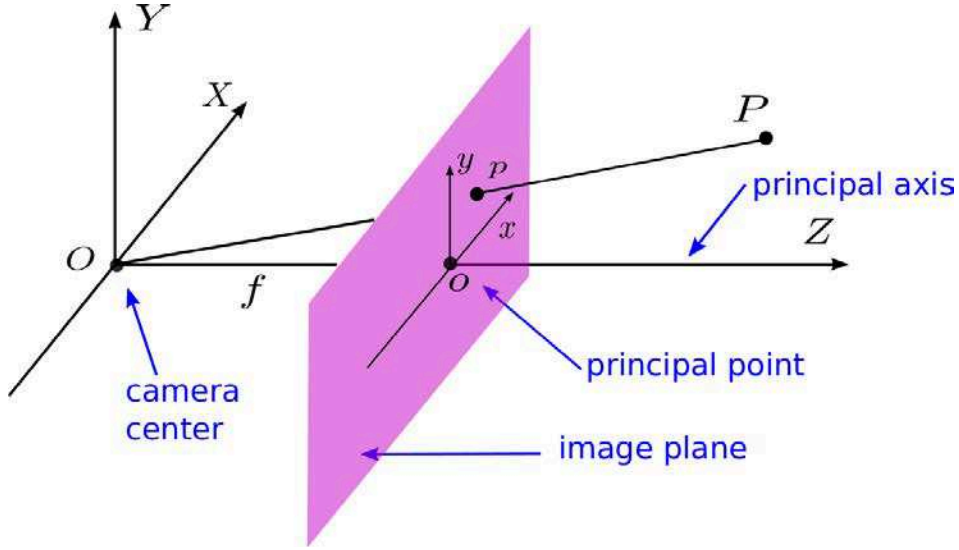


Figure 2.3: Pinhole camera geometry

center and the image plane. The line from the camera center  $O$ , perpendicular to the image plane, is called the *principal axis* and the point it meets the image plane is called the *principal point*. In the pinhole camera model, the projection of a point  $P = (X, Y, Z)^T$  in  $\mathbb{R}_{camera}^3$  into the image plane  $\mathbb{R}_{image}^2$  is  $p = (x, y)^T$ . One can easily find their relationship:

$$(X, Y, Z)^T \mapsto (Xf/Z, Yf/Z)^T = (x, y)^T \quad (2.7)$$

Eq. 2.7 is inconvenient in practice. It can be written in terms of matrix manipulation under homogeneous coordinates:

$$Z \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (2.8)$$

Note that in homogeneous coordinates <sup>2</sup>, a point  $(x, y, w)^T$  is equivalent to  $(x/w, y/w, 1)^T$ , where  $w$  is a scale factor. Hence,

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (2.9)$$

In fact, the coordinates in the image do not correspond to the physical coordinates in the image plane. For a CCD camera, the relationship depends on the size and shape of the pixels and of the position of the CCD chip in the camera. The transformation is illustrated in Fig. 2.4: The image coordinates  $(u, v)$  are obtained by:

<sup>2</sup>[https://en.wikipedia.org/wiki/Homogeneous\\_coordinates](https://en.wikipedia.org/wiki/Homogeneous_coordinates)

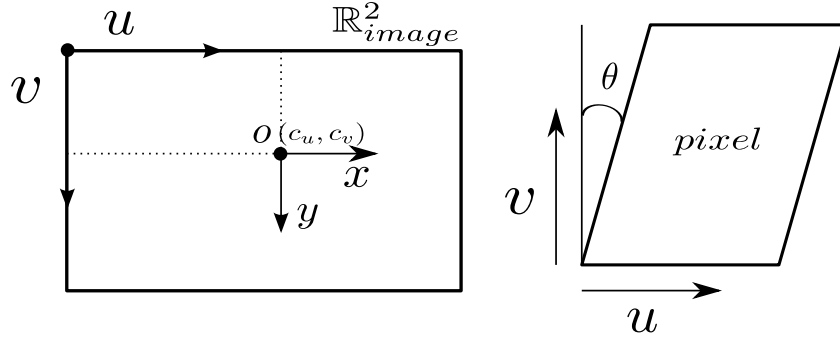


Figure 2.4: From physical coordinates to image coordinates in image plane

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} f k_u & (\tan \theta) f k_v & c_u & 0 \\ 0 & f k_v & c_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (2.10)$$

where  $k_u$  and  $k_v$  are scale factors with [pixels/length] units, which are the number of pixels within a unit length along  $u$  and  $v$  direction, respectively.  $(c_u, c_v)$  is the position of principal point.  $\theta$  is the skew angle between  $u$  and  $v$  axis, as indicated in Fig. 2.4. Let  $f_u = f k_u$ ,  $f_v = f k_v$ , and the skew angle  $\theta = 0$ , Eq. 2.10 is simplified as:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (2.11)$$

Now, let

$$\mathbf{K} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \quad (2.12)$$

then, Eq. 2.11 is denoted as:

$$\tilde{\mathbf{p}} = \mathbf{K}[\mathbf{I}|\mathbf{0}]\tilde{\mathbf{P}} \quad (2.13)$$

where  $\tilde{\mathbf{p}}$  and  $\tilde{\mathbf{P}}$  are the homogeneous coordinates of a point in image coordinate system and camera coordinate system, respectively.  $\mathbf{K}$  is called *intrinsic matrix*, or *camera calibration matrix*.  $\mathbf{I}$  is a  $3 \times 3$  identity matrix.

### 2.1.3.2 Lens Distortion Model

The pinhole camera model described previously in Sec. 2.1.3.1 is an ideal model without considering lens distortions. As a result of several types of imperfections in design and assembly of lenses composing the camera optical system, the expression of Eq. 2.13 does not hold true. In this section, we discuss about two types of lens distortion: *radial distortion* and *tangential distortion*.

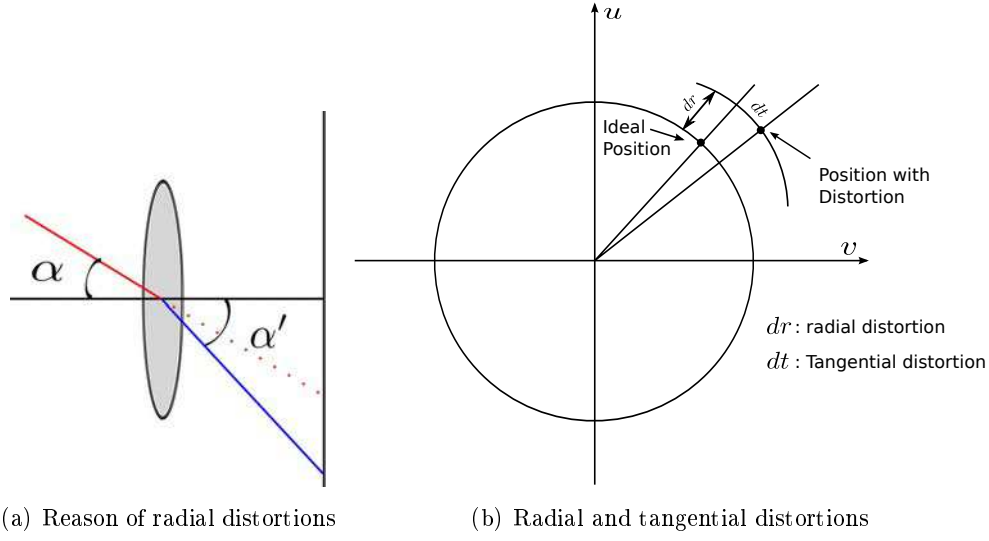


Figure 2.5: lens distortions

- **Radial distortion:** This type of distortion is mainly caused by physical flawed lens geometry, which makes the input and output ray have different angle, as shown in Fig. 2.5 (a). It results in an inward or outward shift of image points from their initial perspective projection, as denoted in  $dr$  in Fig. 2.5 (b).
- **Tangential distortion:** This type of distortion is caused by errors in optical component shapes and optics assembly. It is also called *decentering distortion*. It is illustrated in Fig. 2.5 (b) as  $dt$ .

Let a point  $(x_d, y_d)$  be the distorted coordinates of a point  $(x, y)$  in image plane coordinate system  $\mathbb{R}_{image}^2$ , and  $r^2 = x^2 + y^2$ . Then:

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \cdot \begin{pmatrix} x \\ y \end{pmatrix} + dt \quad (2.14)$$

where  $k_1, k_2, k_3$  are the coefficients of radial distortion.  $dt$  is the tangential distortion vector:

$$dt = \begin{bmatrix} 2p_1 xy + p_2(r^2 + 2x^2) \\ p_1(r^2 + 2y^2) + 2p_2 xy \end{bmatrix} \quad (2.15)$$

where  $p_1, p_2$  are the coefficients of tangential distortion. Therefore, under the lens distortion model, the pixel coordinates  $(u, v)$  are related to distorted coordinates similar to Eq. 2.13:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{K} \begin{pmatrix} x_d \\ y_d \\ 1 \end{pmatrix} \quad (2.16)$$

where  $(x_d, y_d)$  is represented in Eq. 2.14.

### 2.1.4 Binocular Stereo Vision System Model

As an extension of monocular camera model, a general binocular stereo vision model is shown in Fig. 2.6. The left and right cameras are modeled by the classical pinhole

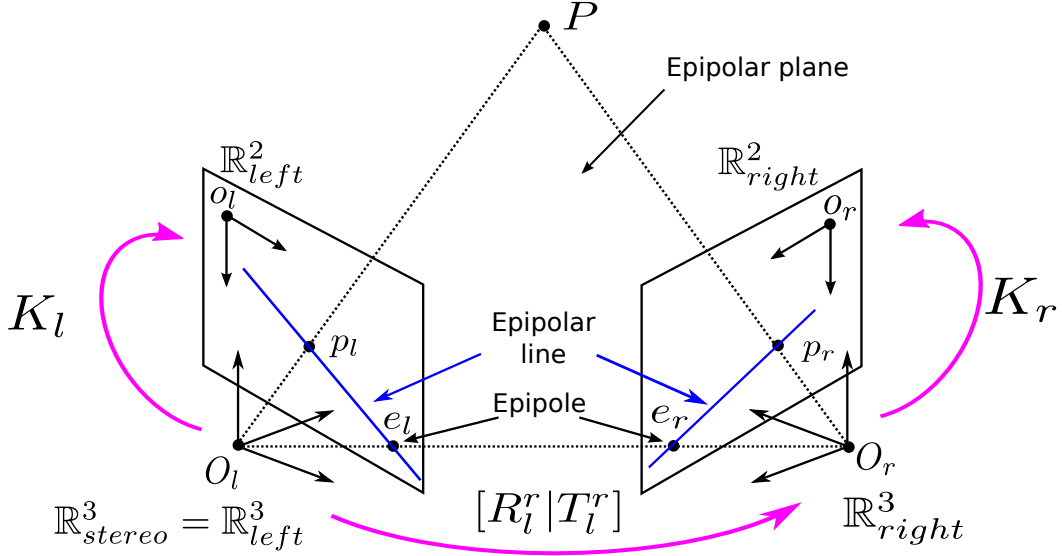


Figure 2.6: Geometry of binocular stereo vision system

model, with intrinsic matrices  $\mathbf{K}_l$  and  $\mathbf{K}_r$  respectively. Therefore, suppose there is a point  $P$  in 3D space, its corresponding coordinates in the left camera coordinate system  $\mathbb{R}^3_{left}$ , the right camera coordinate system  $\mathbb{R}^3_{right}$ , the left image coordinate system  $\mathbb{R}^2_{left}$  and the right image coordinate system  $\mathbb{R}^2_{right}$  are  $P_l, P_r, p_l, p_r$  respectively. Then, they are connected as:

$$\begin{aligned}\tilde{p}_l &= \mathbf{K}_l [\mathbf{I} | \mathbf{0}] \cdot \tilde{P}_l \\ \tilde{p}_r &= \mathbf{K}_r [\mathbf{I} | \mathbf{0}] \cdot \tilde{P}_r \\ P_l &= \mathbf{R}_r^l \cdot P_r + \mathbf{T}_r^l\end{aligned}\tag{2.17}$$

where  $\tilde{\cdot}$  denotes homogeneous coordinates.  $\mathbf{R}_r^l$ , (a  $3 \times 3$  rotation matrix) together with  $\mathbf{T}_r^l$  (a  $3 \times 1$  translation vector) represent rigid 3D Euclidean transformation from  $\mathbb{R}^3_{right}$  to  $\mathbb{R}^3_{left}$ . In similar,  $\mathbf{R}_l^r$  and  $\mathbf{T}_l^r$  describe a rigid 3D Euclidean transformation from  $\mathbb{R}^3_{left}$  to  $\mathbb{R}^3_{right}$ . Usually,  $\mathbf{R}$  and  $\mathbf{T}$  are called *extrinsic parameters*.

Another characteristic of binocular vision system is the epipolar geometry. In epipolar geometry, the point intersections of the base line  $O_l O_r$  with the two image planes are *epipoles*. Any point  $P$  in 3D space with two camera centers  $O_l$  and  $O_r$  define an *epipolar plane*. The line intersections of the epipolar plane with the two image planes are called *epipolar lines*. Indeed, all epipolar lines intersect at epipole. An epipolar plane defines the correspondence between the epipolar lines. All the epipoles, epipolar plane and epipolar lines are drawn in Fig. 2.6. Based on the above definitions, *epipolar constraint* is that: suppose  $p_l$  is the left image position

for a scene point  $P$ , then, the corresponding point  $p_r$  in the right image must lie on the epipolar line. The mathematical interpretation is that, for a corresponding point pair  $(p_l, p_r)$  in homogeneous coordinates  $\tilde{z}$ , we have:

$$\tilde{p}_l^T F \tilde{p}_r = 0 \quad (2.18)$$

where  $F$  is called *fundamental matrix*.  $F\tilde{p}_r$  describes a line (an epipolar line) on which the corresponding point  $\tilde{p}_l$  must lie.

## 2.2 Stereo Vision System Calibration

### 2.2.1 Intrinsic Calibration of a camera

Intrinsic calibration of a camera is to find the intrinsic parameters described in Sec. 2.1.3 (focal length, principal point position, as well as lens distortion). It is essential in 3D computer vision, such as 3D Euclidean structure from motion, object avoidance in robot navigation, etc.. A calibrated camera can be used as a quantitative sensor being capable of measuring distance and 3D structure information.

Intrinsic calibration of a camera has been researched for decades. Among all the proposed methods, 2D planar pattern based methods, e.g. Zhengyou Zhang's method [Zhang 2000], is the most popular. This method requires the camera to observe a planar pattern (e.g. 2D chessboard) shown at a few different orientations. Either the camera or the planar pattern can be freely moved. It is implemented in a well known camera calibration toolbox for Matlab<sup>3</sup>, developed by Jean-Yves Bouguet [Bouguet 1999]. This Matlab calibration toolbox is applied in our practices. The whole process is as follows:

- Prepare a chessboard pattern and attach it to a planar surface.
- Take a few images of the chessboard under different positions with different orientations, as shown in Fig. 2.7 (a).
- Detect the corner points on the chessboard images, as shown in Fig. 2.7 (b).
- Estimate the intrinsic parameters and the coefficients of radial distortion through a nonlinear optimization process. The calibration results and error analysis are shown in Fig. 2.7.

### 2.2.2 Extrinsic Calibration of Binocular Vision System

Extrinsic calibration of a stereo rig is to estimate extrinsic parameters between left and right cameras (a rotation matrix and a translation vector, as described in Sec. 2.1.4). Given corresponding image points from two views, the extrinsic parameters can be calculated from SVD factorizing the fundamental matrix  $F$  [Hartley 2004].

<sup>3</sup>[http://www.vision.caltech.edu/bouguetj/calib\\_doc/](http://www.vision.caltech.edu/bouguetj/calib_doc/)

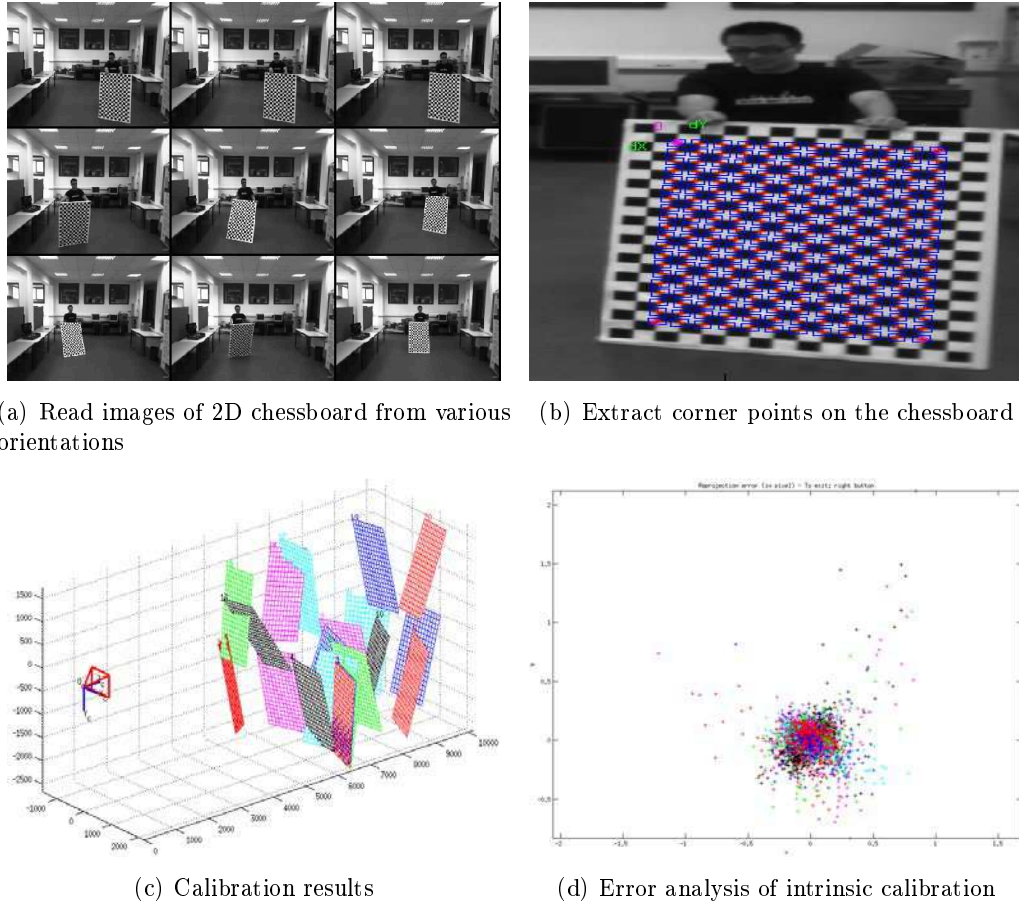


Figure 2.7: Calibrating a camera in Bouguet's Matlab camera calibration toolbox

Whereas, in Jean-Yves Bouguet's Matlab calibration toolbox, intrinsic parameters for the left and right cameras are calculated at first. Then, the method in [Hartley 2004] is adopted for acquiring an initial estimation of the extrinsic parameters. At last, this initial estimation is optimized by minimizing reprojection errors of all the corresponding corner points in the two views. A demonstrative example in Fig. 2.8 shows the extrinsic calibration results of our stereo vision system.

## 2.2.3 Image Undistortion and Stereo Rectification

### 2.2.3.1 Image Undistortion

Image undistortion is to compensate the effects of lens distortions described in Sec. 2.1.3.2. Indeed, it consists of a transformation from physically distorted image to an ideal image under the pinhole camera model (see in Sec. 2.1.3.1). For every pixel in undistorted image, we have to compute its distorted location. That is, for each pixel  $(u, v)$  in the corrected image, undistortion process computes its corresponding



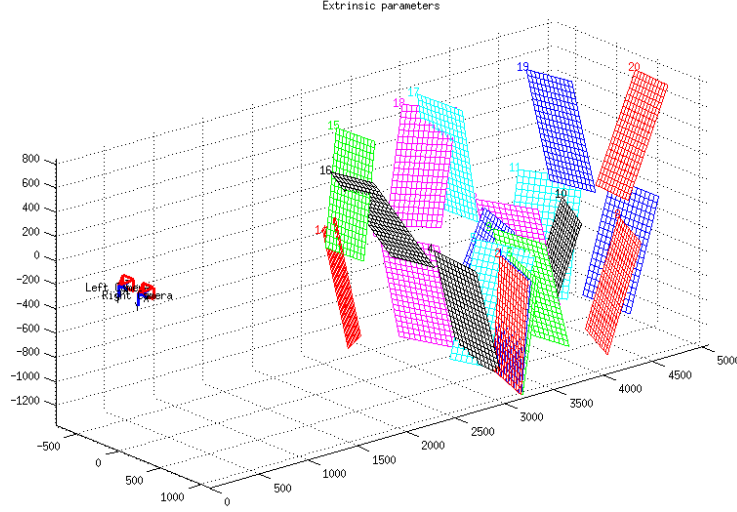


Figure 2.8: Extrinsic calibration results of a stereo vision system

coordinates in the original image. The process is as follows:

$$\begin{aligned}
 x &\leftarrow (u - c_x)/f_x \\
 y &\leftarrow (v - c_y)/f_y \\
 x' &\leftarrow x(1 + k_1r^2 + k_2r^4 + k_3r^6) + 2p_1xy + p_2(r^2 + 2x^2) \\
 y' &\leftarrow y(1 + k_1r^2 + k_2r^4 + k_3r^6) + p_1(r^2 + 2y^2) + 2p_2xy \\
 map_x(u, v) &\leftarrow x'f_x + c_x \\
 map_y(u, v) &\leftarrow y'f_y + c_y
 \end{aligned} \tag{2.19}$$

where  $(c_x, c_y), f_x, f_y$  are the intrinsic parameters calculated in Sec. 2.2.1.  $k_1, k_2, k_3, p_1, p_2$  represent radial and tangential distortion coefficients described in Sec. 2.1.3.2.

### 2.2.3.2 Stereo Rectification

Stereo rectification is to align the image planes of the two views. In practice, if the two cameras are aligned to be coplanar, matching point between two images is simplified to searching in one dimension - a horizontal line parallel to the baseline between the cameras. In fact, stereo rectification is to move the epipoles to infinity and match up epipolar lines. Its effects are demonstrated in Fig. 2.9. An outline of stereo rectification is given in [Hartley 2004]:

1. Find initial point correspondences
2. Compute the fundamental matrix
3. Compute projection transformations  $H_l$  and  $H_r$  that map the epipoles  $e_l$  and  $e_r$  to infinity.

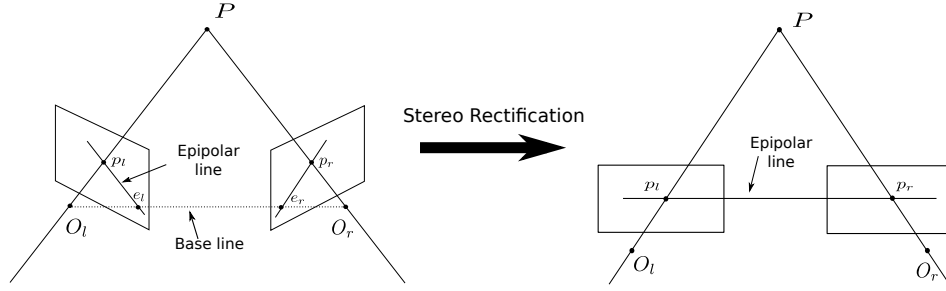


Figure 2.9: Stereo rectification effects

4. Warp the left image according to  $H_l$  and the right image according to  $H_r$

For the inputs of a stereo vision system (the image pairs acquired by the left and right cameras), image undistortion and stereo rectification are usually utilized as preprocessings to simplify 3D computer vision tasks. Be similar to Eq. 2.19, for each pixel  $(u, v)$  in the destination image, undistortion together with stereo rectification compute its corresponding coordinates in the original image:

$$\begin{aligned}
 x &\leftarrow (u - c'_x) / f'_x \\
 y &\leftarrow (v - c'_y) / f'_y \\
 [XYW]^T &\leftarrow R^{-1} \times [xy1]^T \\
 x' &\leftarrow X/W \\
 y' &\leftarrow Y/W \\
 x'' &\leftarrow x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2 (r^2 + 2x'^2) \\
 y'' &\leftarrow y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1 (r^2 + 2y'^2) + 2p_2 x' y' \\
 map_x(u, v) &\leftarrow x'' f_x + c_x \\
 map_y(u, v) &\leftarrow y'' f_y + c_y
 \end{aligned} \tag{2.20}$$

where  $(k_1, k_2, k_3, p_1, p_2)$  are the distortion coefficients in one view (e.g. the left camera).  $R$  is the rectification transformation ( $3 \times 3$  matrix) for one view.  $c_x, c_y, f_x, f_y$  are the original intrinsic parameters for one view.  $c'_x, c'_y, f'_x, f'_y$  are intrinsic parameters of new camera projection matrix after rectification. Note that after rectification, the intrinsic matrices  $K_l$  and  $K_r$  are the same. That is, after undistortion and rectification, each stereo image pair can be thought as acquired by one camera with pure translation movement under ideal pinhole model.

The results after undistortion and rectification are shown in Fig. 2.10, where the left and right images are well aligned (the greelines are helped to see whether the corresponding points in the two views are colinear).

#### 2.2.4 Corner Points Triangulation

3D triangulation is a process of reconstructing a point in 3D coordinates from its corresponding projections in two or more images, giving the configuration parameters of the cameras. Here, we only discuss 3D point triangulation from two views.



Figure 2.10: Image undistortion and stereo rectification results

In ideal situation, where the two stereo images are perfectly rectified and undistorted, is drawn in Fig. 2.11. Recalled to Sec. 2.2.3.2, the left and right cameras

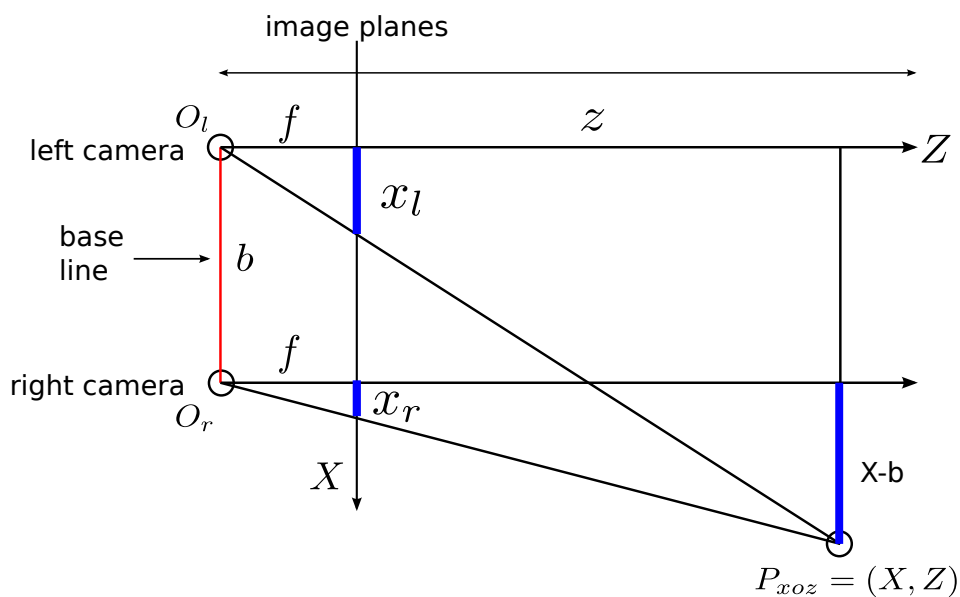


Figure 2.11: 3D triangulation between two parallel, identical cameras

are assumed to be identical and the images planes are colinear. Let  $f$  be the focal length,  $b$  be the distance between the two camera centers,  $XZ$  is a plane where the optical axes lie,  $XZ$  plane is perpendicular to the image plane of both cameras,  $X$  axis equals the baseline. A 3D point  $P$ 's projection in  $XZ$  plane is  $P_{xoz} = (X, Z)$ . As marked in the figure, the two lines that connect camera centers and  $P$  intersect image planes at  $x_l$  and  $x_r$ , which are indeed the image coordinates in  $X$

axis of  $P$ 's two projections. It is easy to deduct the followings from similar triangles:

$$\begin{aligned}\frac{Z}{f} &= \frac{X}{x_l} \\ \frac{Z}{f} &= \frac{X - b}{x_r}\end{aligned}\tag{2.21}$$

As the  $Y$ -axis is perpendicular to the  $XZ$  plane, we can have the following equation in similar:

$$\frac{Z}{f} = \frac{Y}{y_l} = \frac{Y}{y_r}\tag{2.22}$$

Usually, we name  $\Delta = x_l - x_r$  as *disparity*. Then, 3D coordinates  $(X, Y, Z)$  of a point  $P$  in the left camera coordinate system can be derived as:

$$\begin{aligned}X &= \frac{(u_l - c_u) \cdot b}{\Delta} \\ Y &= \frac{(v_l - c_v) \cdot b}{\Delta} \\ Z &= \frac{f \cdot b}{\Delta}\end{aligned}\tag{2.23}$$

where  $u_l$  and  $v_l$  are the pixel coordinates of  $P$  in the left image,  $(c_u, c_v)$  is the position of left camera's principal point.

## 2.3 Least Squares Estimation

The method of least squares [Rao 2008] is about estimating parameters of a model by minimizing the squared discrepancies between observed data and their expected values provided by a model. Suppose a data set consisting of  $n$  data pairs  $(\mathbf{x}_i, y_i)$ ,  $i = 1, \dots, n$ , where  $\mathbf{x}_i$  is an independent observed variable vector and  $y_i$  is an observed response whose value is decided by observation. The model is represented as

$$y = f(\mathbf{x}, \boldsymbol{\beta})\tag{2.24}$$

where  $\boldsymbol{\beta}$  is a vector of parameters defined by the model. The goal is to find  $\boldsymbol{\beta}$  for the model which "best" fit the observation. The least squares method aims to find the optimal values of  $\boldsymbol{\beta}$  by minimizing squared residuals:

$$\begin{aligned}S &= \sum_{i=1}^n r_i^2 \\ r_i &= y_i - f(\mathbf{x}_i, \boldsymbol{\beta})\end{aligned}\tag{2.25}$$

where  $S$  is a summation of residuals  $r_i$ .

According to whether or not the residuals  $r_i$  are linear or not, least squares estimation can be divided into two categories: *linear least squares* and *non-linear least squares*. If measurement errors exist in observed independent variables  $x_i$ , *errors-in-variables* [Rao 2008] model may be considered instead of standard least square estimation. In this section, linear least squares/non-linear squares estimators and their variations are discussed.

### 2.3.1 Linear Least Squares

In linear least square problems, the model  $f(\mathbf{x}_i, \boldsymbol{\beta})$  can be expressed linearly in terms of parameters  $\boldsymbol{\beta}$ . Mathematically, linear least squares solves an over-determined system of linear equation. Usually, linear least squares problem have a closed-form solution. For example, consider an over-determined system:

$$\sum_{j=1}^n X_{ij}\beta_j = y_i, (i = 1, 2, \dots, m) \quad (2.26)$$

of  $m$  linear equations with  $n$  coefficients,  $\beta_1, \beta_2, \dots, \beta_n$  ( $m > n$ ). This can be written in matrix form as:

$$\mathbf{X}\boldsymbol{\beta} = \mathbf{y} \quad (2.27)$$

where

$$\mathbf{X} = \begin{bmatrix} X_{11} & X_{12} & \dots & X_{1n} \\ X_{21} & X_{22} & \dots & X_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{m1} & X_{m2} & \dots & X_{mn} \end{bmatrix}, \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \quad (2.28)$$

As mentioned before, least squares estimation seeks the coefficients  $\boldsymbol{\beta}$  minimizing the summation of residuals:

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= \arg \min_{\boldsymbol{\beta}} S(\boldsymbol{\beta}) \\ S(\boldsymbol{\beta}) &= \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 \end{aligned} \quad (2.29)$$

The solution for Eq. 2.29 can be proven to be [Rao 2008]:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \mathbf{X}^+ \mathbf{y} \quad (2.30)$$

where  $\mathbf{X}^+$  is the Moore-Penrose pseudo-inverse [Bretschner 1995] of  $\mathbf{X}$ .

#### 2.3.1.1 Ordinary Linear Least Squares

When applying least squares in real data sets, errors/noises are inevitable. Suppose there exist errors between the actually observed responses  $y_i$  and the "predicted" outcomes  $\mathbf{x}_i^T \boldsymbol{\beta}$ , the model in Eq. 2.27 can be written as:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \quad (2.31)$$

where  $\mathbf{y}$  and  $\boldsymbol{\varepsilon}$  are  $m \times 1$  vectors representing actual response values and errors. Assume the errors are zero mean independent and identically distributed (i.i.d) Gaussian variables that:  $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$ , where  $\mathbf{I}$  is an  $m \times m$  identity matrix, and  $\sigma^2$  determines the variance of each observation. The ordinary least squares estimation of  $\boldsymbol{\beta}$  can be given by explicit formula as in Eq. 2.30:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (2.32)$$

Hence, the predicted value from the observation is:

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \quad (2.33)$$

Beside estimating the value of  $\boldsymbol{\beta}$ , the covariance matrix of least squares estimator is also important. It can be shown that, given  $\mathbf{X}$ , the covariance matrix  $\mathbf{Q}_{\hat{\boldsymbol{\beta}}}$  of the estimator  $\hat{\boldsymbol{\beta}}$ , equals to [Hayashi 2011]:

$$\mathbf{Q}_{\hat{\boldsymbol{\beta}}} = (\mathbf{X}^T\mathbf{X})^{-1}\sigma^2 \quad (2.34)$$

$\sigma^2$  can be estimated from the data as:

$$\hat{\sigma}^2 = \frac{1}{m-n}\|\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\|^2 \quad (2.35)$$

Therefore,  $\mathbf{Q}_{\hat{\boldsymbol{\beta}}}$  can be estimated by:

$$\mathbf{Q}_{\hat{\boldsymbol{\beta}}} = (\mathbf{X}^T\mathbf{X})^{-1}\hat{\sigma}^2 \quad (2.36)$$

### 2.3.1.2 Weighted Linear Least Squares

The ordinary linear least squares assumes that each data point provides equally precise information about the estimation. In other words, the error term  $\boldsymbol{\varepsilon}$  is constant over all observations. Clearly, this assumption does not hold in most modeling applications. Weighted linear least squares offers a better approach by attempting to give each data point its proper amount of influence over the parameter estimation. In this case, one can minimize the weighted sum of squares:

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= \arg \min_{\boldsymbol{\beta}} \sum_{i=1}^m w_i |y_i - \mathbf{x}_i\boldsymbol{\beta}|^2 = \arg \min_{\boldsymbol{\beta}} \|\mathbf{W}^{1/2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\|^2 \\ &= \arg \min_{\boldsymbol{\beta}} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T \mathbf{W} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \end{aligned} \quad (2.37)$$

where  $w_i > 0$  is the weight of the  $i$ th observation, and  $\mathbf{W}$  is a  $m \times m$  diagonal weight matrix. Ideally, the weight matrix should be equal to the reciprocal of the variance of the measurement.

$$\mathbf{W} = \text{cov}(\mathbf{y})^{-1} = \begin{bmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_m \end{bmatrix} = \begin{bmatrix} \frac{1}{\sigma_1^2} & & & \\ & \frac{1}{\sigma_2^2} & & \\ & & \ddots & \\ & & & \frac{1}{\sigma_m^2} \end{bmatrix} \quad (2.38)$$

where  $\sigma_i^2$  is the variance of error in  $i$ th observation data point. Indeed,  $\mathbf{W}$  is the covariance matrix of the error  $\boldsymbol{\varepsilon}$ . When  $\mathbf{W} = \mathbf{1}/\sigma^2 \cdot \mathbf{I}$ , the weighted linear least squares becomes ordinary linear least squares. A closed-form solution is [Strutz 2010]:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{y} \quad (2.39)$$

The covariance matrix  $\mathbf{Q}_{\hat{\beta}}$  of estimation  $\hat{\beta}$ , can be acquired by applying error propagation law:

$$\mathbf{Q}_{\hat{\beta}} = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{X} (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \quad (2.40)$$

However, in many real practices, we do not know  $\sigma_i^2$ . We could set the weights manually or by an iterative approach: iteratively reweighted least squares (IRLS) [Chartrand 2008]. IRLS uses an iterative approach to solve a weighted least squares problem of the form:

$$\hat{\beta}^{(t+1)} = \arg \min_{\beta} \sum_{i=1}^m w_i |y_i - \mathbf{x}_i \beta_i|^2 \quad (2.41)$$

In each step  $t + 1$ , the IRLS estimates  $\hat{\beta}^{(t+1)}$  by the aforementioned weighted linear least squares:

$$\hat{\beta}^{(t+1)} = (\mathbf{X}^T \mathbf{W}^{(t)} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W}^{(t)} \mathbf{y} \quad (2.42)$$

where  $\mathbf{W}^{(t)}$  is a diagonal weight matrix in  $t$ th step with each element:

$$w_i^{(t)} = |y_i - \mathbf{x}_i \beta^{(t)}|^{-1} \quad (2.43)$$

### 2.3.1.3 Total Linear Least Squares

The above ordinary and weighted linear least squares only assume errors in the observed response  $\mathbf{y}$ , as interpreted in Eq. 2.31. However, it is a more natural way to model data if both observation  $\mathbf{X}$  and response  $\mathbf{y}$  are contaminated by errors. Total least squares (TLS) [Huffel 1991], is a natural generalization of ordinary least squares where errors in all observational variables are taken into account.

TLS problem is defined as follows: we are given an over-determined set of  $m$  equations  $x_i \beta = y_i$  ( $i = 1 \dots m$ ) in  $n$  unknowns  $\beta_j$  ( $j = 1 \dots n$ ), compiled to a matrix equation  $\mathbf{X} \beta \approx \mathbf{y}$ . Both the vector  $\mathbf{y}$  as well as the matrix  $\mathbf{X}$  are subject to errors. The total least squares problem consists in finding an  $m \times n$  matrix  $\hat{\mathbf{X}}$  and an  $m \times 1$  vector  $\hat{\mathbf{y}}$  for which the equation

$$\hat{\mathbf{X}} \beta = \hat{\mathbf{y}} \quad (2.44)$$

has an exact solution, under the condition that the deviation between  $(\mathbf{X}|\mathbf{y})$  and  $(\hat{\mathbf{X}}|\hat{\mathbf{y}})$  is minimal in terms of the *Frobenius norm*:

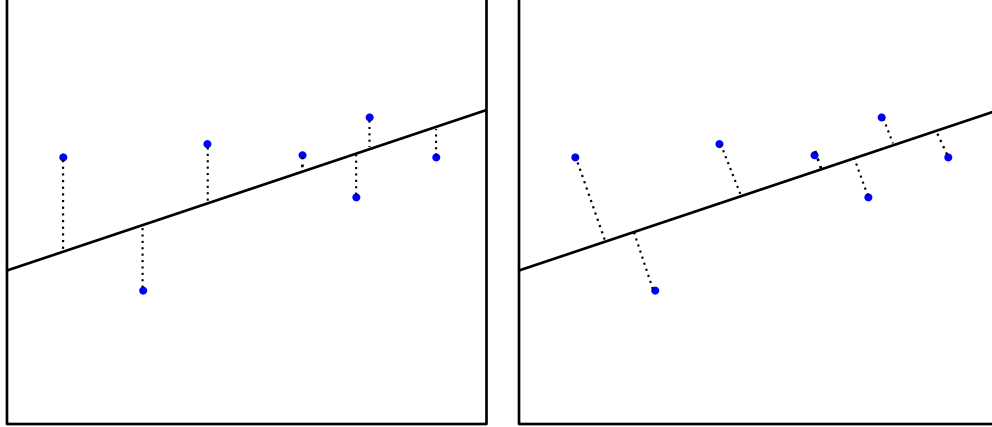
$$\min \quad \|(\mathbf{X}|\mathbf{y}) - (\hat{\mathbf{X}}|\hat{\mathbf{y}})\|_F \quad (2.45)$$

For a  $m \times n$  matrix  $\mathbf{A} = (a_{i,j})$   $i = 1 \dots m, j = 1 \dots n$ , the *Frobenius norm* is defined as:

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{i,j}|^2} = \sqrt{\text{trace}(\mathbf{A}^* \mathbf{A})} = \sqrt{\sum_{i=1}^{\min(m,n)} \sigma_i^2} \quad (2.46)$$

where  $\mathbf{A}^*$  denotes the conjugate transpose,  $\sigma_i$  are the singular values of  $\mathbf{A}$ . Furthermore, we denote the errors lying in  $\mathbf{X}$  and  $\mathbf{y}$  as  $\mathbf{E}$  ( $m \times n$  matrix) and  $\mathbf{r}$  ( $m \times 1$  vector), respectively. The TLS model becomes:

$$(\mathbf{X} + \mathbf{E}) \beta = (\mathbf{y} + \mathbf{r}) \quad (2.47)$$



(a) Ordinary least squares, errors are distributed only in  $y$ -coordinates, distances are measured along the  $y$ -axis  
 (b) Total least squares, errors are distributed in both  $x$ - and  $y$ -coordinates, orthogonal distances are used

Figure 2.12: The difference between Ordinary Least Squares and Total Least Squares

then, TLS problem can be formally written as:

$$\begin{aligned} & \underset{\mathbf{E}, \mathbf{r}}{\text{minimize}} && \|[\mathbf{E}|\mathbf{r}]\|_F \\ & \text{subject to} && (\mathbf{X} + \mathbf{E})\boldsymbol{\beta} = \mathbf{y} + \mathbf{r} \end{aligned} \quad (2.48)$$

where  $[\mathbf{E}|\mathbf{r}]$  is an augmented matrix and  $\|\cdot\|_F$  is the *Frobenius norm*.

Let  $\mathbf{C} = [\mathbf{X}|\mathbf{y}]$  be the  $m \times (n+1)$  augmented matrix of  $\mathbf{X}$  and  $\mathbf{y}$  side by side, and  $\mathbf{C} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$  is its SVD decomposition. Assume that the singular values of  $\mathbf{C}$  are such that:  $\sigma_1 \geq \dots \geq \sigma_k > \sigma_{k+1} = \dots \sigma_{n+1}$ . Then, it has been shown in [Huffel 1991] that  $\boldsymbol{\beta}_{TLS}$  minimizing Eq. 2.48 when all the errors are i.i.d Gaussian variables, is:

$$\boldsymbol{\beta}_{TLS} = -\frac{\mathbf{y}}{\alpha} \quad (2.49)$$

where the vector  $(\mathbf{y}, \alpha)$  of norm 1, with  $\alpha \neq 0$ , is in the subspace  $\mathbf{S}_k$  spanned by the right singular vectors  $\mathbf{v}^{k+1}, \dots, \mathbf{v}^{n+1}$  of  $\mathbf{V}$ . To put it simple,  $\boldsymbol{\beta}_{TLS}$  is given by taking the negative of the results of normalizing the first  $n$  elements of the last column  $\mathbf{V}$  by the last element of  $\mathbf{V}$ .

The ordinary least squares and total least squares methods evaluate the estimation accuracy in different ways: the ordinary least squares minimizes the sum of distances from data to the fitting line, while the total least squares minimizes the sum of orthogonal distances instead. The difference is shown in Fig. 2.12

However, in [Huffel 1991, Markovsky 2007], the covariance of TLS estimator is not given. One important reason is that the likelihood function of TLS problem contains the true (noiseless) values of measurements as *nuisance parameters* [Gleser 1981]. As a result, the dimension of likelihood function increases rapidly with the number



of observations. In [Nestares 2000], the likelihood function of TLS is simplified with several assumptions and the *Cramer-Rao lower bound (CRLB)*<sup>4</sup> of TLS is given:

$$\mathbf{Q}_{(\boldsymbol{\beta})_{TLS}} \geq CRLB(\boldsymbol{\beta}_{TLS}) = \frac{1}{\gamma} \sigma_n^2 \|(\boldsymbol{\beta})_{TLS}\|^2 (\mathbf{X}^T \mathbf{X})^{-1} \quad (2.50)$$

where  $\sigma_n$  is the variance of the assuming i.i.d distributed measurement noises.  $\gamma = \frac{\sigma_0^2}{\sigma_n^2 + \sigma_0^2}$  is related with the signal to noise ratio (SNR). If the SNR is high ( $\sigma_0^2 \gg \sigma_n^2$ ), then  $\gamma \approx 1$ .

---

<sup>4</sup>[http://en.wikipedia.org/wiki/Cramer-Rao\\_bound](http://en.wikipedia.org/wiki/Cramer-Rao_bound)

### 2.3.2 Non-linear Least Squares

Be different to the linear least squares introduced in Sec. 2.3.1, the observation model  $f(\mathbf{x}, \boldsymbol{\beta})$  (in Eq. 2.24) in non-linear least squares can not be represented in form of linear combination of parameters  $\boldsymbol{\beta} = \beta_i (i = 1 \dots n)$ . Solutions of non-linear least squares usually approximate non-linear model by a linear one, and refine the parameter vector  $\boldsymbol{\beta}$  iteratively. Recall to Eq. 2.24, given  $m$  data points  $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_m, \mathbf{y}_m)$  and a non-linear model  $\mathbf{y} = f(\mathbf{x}, \boldsymbol{\beta})$ , with  $m \geq n$ . Non-linear least squares is to find the parameter vector  $\boldsymbol{\beta}$ , which minimizes the sum of squares:

$$S = \sum_{i=1}^m r_i^2 \quad (2.51)$$

where the residuals  $r_i$  are given by:

$$r_i = y_i - f(\mathbf{x}_i, \boldsymbol{\beta}), i = 1, 2, \dots, m \quad (2.52)$$

According to the knowledge of calculus, the sum of squares  $S$  reaches a minimum when the gradient is zero. Since the model contains  $n$  parameters, there are  $n$  gradient equations:

$$\frac{\partial S}{\partial \beta_j} = 2 \sum_i r_i \frac{\partial r_i}{\partial \beta_j} = 0 \quad (j = 1, \dots, n) \quad (2.53)$$

For a non-linear model, the partial derivatives  $\frac{\partial r_i}{\partial \beta_j}$  are in terms of desired parameters as well as independent variables. Hence, it is usually impossible to get a closed solution. Instead, iterative methods are widely chosen to approximate optimal results in successive manners:

$$\boldsymbol{\beta}_j \approx \boldsymbol{\beta}_j^{k+1} = \boldsymbol{\beta}_j^k + \Delta \boldsymbol{\beta}_j \quad (2.54)$$

where  $k$  denotes the  $k$ th round of iteration.  $\Delta \boldsymbol{\beta}$  is known as the shift vector. In each step of iteration, the model  $f(\mathbf{x}, \boldsymbol{\beta})$  can be linearly approximated by a first-order Taylor series expansion at  $\boldsymbol{\beta}^k$ :

$$f(\mathbf{x}_i, \boldsymbol{\beta}) \approx f(\mathbf{x}_i, \boldsymbol{\beta}^k) + \sum_j \frac{\partial f(\mathbf{x}_i, \boldsymbol{\beta}^k)}{\partial \beta_j} (\beta_j - \beta_j^k) \approx f(\mathbf{x}_i, \boldsymbol{\beta}^k) + \sum_j \mathbf{J}_{ij} \Delta \beta_j \quad (2.55)$$

where  $\mathbf{J}_{ij}$  is the  $(i, j)$ th element of *Jacobian matrix*<sup>5</sup>  $\mathbf{J}$ .

$$\mathbf{J} = \begin{bmatrix} \partial f(\mathbf{x}_1, \boldsymbol{\beta}^k) / \partial \beta_1 & \cdots & \partial f(\mathbf{x}_1, \boldsymbol{\beta}^k) / \partial \beta_n \\ \vdots & \ddots & \vdots \\ \partial f(\mathbf{x}_m, \boldsymbol{\beta}^k) / \partial \beta_1 & \cdots & \partial f(\mathbf{x}_m, \boldsymbol{\beta}^k) / \partial \beta_n \end{bmatrix} \quad (2.56)$$

Therefore,  $J_{ij} = -\frac{\partial r_i}{\partial \beta_j}$ . and the residuals in Eq. 2.52 are represented as:

$$\begin{aligned} r_i &= y_i - (f(\mathbf{x}_i, \boldsymbol{\beta}^k) + \sum_j \mathbf{J}_{ij} \Delta \beta_j) \\ &= \Delta y_i - \sum_{j=1}^n \mathbf{J}_{ij} \Delta \beta_j \end{aligned} \quad (2.57)$$

<sup>5</sup>[https://en.wikipedia.org/wiki/Jacobian\\_matrix](https://en.wikipedia.org/wiki/Jacobian_matrix)

Substituting Eq. 2.57 into Eq. 2.53, and rearranging the equations in terms of matrix, we have:

$$(\mathbf{J}^T \mathbf{J}) \Delta \boldsymbol{\beta} = \mathbf{J}^T \Delta \mathbf{y} \quad (2.58)$$

This equation forms the basis for many other iterative approaches of non-linear least squares problem.

### 2.3.2.1 Gauss-Newton Algorithm

Gauss-Newton algorithm (GNA) [Bjorck 1996] is very popular to solve non-linear least squares problems. Its biggest advantage is to avoid computing second derivatives.

According to Eq. 2.58, the shift vector for each iteration step can be solved by linear least squares as in Eq. 2.32

$$\Delta \boldsymbol{\beta} = (\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T \Delta \mathbf{y} \quad (2.59)$$

Then, in each iteration step, estimated parameters are updated as:

$$\boldsymbol{\beta}^{k+1} = \boldsymbol{\beta}^k + (\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T \mathbf{r}(\boldsymbol{\beta}^k) \quad (2.60)$$

where the residual  $\mathbf{r}(\boldsymbol{\beta}^k)$  is expressed as:

$$\mathbf{r}(\boldsymbol{\beta}^k) = \mathbf{y} - f(\mathbf{x}, \boldsymbol{\beta}^k) \quad (2.61)$$

In summary, given an initial guess of  $\boldsymbol{\beta}_0$  at the beginning, Gauss-Newton algorithm iteratively updates the required parameters as in Eq. 2.60 until  $\mathbf{r}(\boldsymbol{\beta}^k)$  closes to zero or  $\Delta \boldsymbol{\beta}^k$  is tiny.

### 2.3.2.2 Levenberg-Marquardt Algorithm

Levenberg-Marquardt algorithm (LMA) [Levenberg 1944, Marquardt 1963] provides a more robust solution of non-linear least squares than Gauss-Newton algorithm (GNA). In many cases, it converges to local minimum, even if the initial guess starts very far from the final minimum.

Like Gauss-Newton algorithm, Levenberg-Marquardt algorithm is also an iterative method. Recall to Eq. 2.58, which is the basis of Gauss-Newton algorithm. Levenberg [Levenberg 1944] firstly replace this equation by a "damped version":

$$(\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I}) \Delta \boldsymbol{\beta} = \mathbf{J}^T \Delta \mathbf{y} \quad (2.62)$$

where  $\mathbf{I}$  is the identity matrix.  $\lambda$  is a non-negative damping factor, which is adjusted at each iteration. If the  $S$  in Eq. 2.51 reduces rapidly, a small value of  $\lambda$  can be used, making the algorithm closer to Gauss-Newton algorithm. Whereas, if an iteration causes insufficient reduction of the residual,  $\lambda$  can be increased, giving a step closer to the gradient descent direction (note that the gradient of  $S$  with respect to  $\boldsymbol{\beta}$  equals  $-2(\mathbf{J}^T \Delta \mathbf{y})$ ). Levenberg's algorithm has the disadvantage that if the value

of the damping factor  $\lambda$  is large, inverting  $\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I}$  is not used at all. Marquardt proposed in [Marquardt 1963] to substitute Eq. 2.62 as:

$$(\mathbf{J}^T \mathbf{J} + \lambda \text{diag}(\mathbf{J}^T \mathbf{J})) \Delta \boldsymbol{\beta} = \mathbf{J}^T \Delta \mathbf{y} \quad (2.63)$$

where  $\text{diag}(\mathbf{J}^T \mathbf{J})$  is a diagonal matrix consisting of the diagonal elements of  $\mathbf{J}^T \mathbf{J}$ . This equation is the basis of Levenberg-Marquardt algorithm. Similar to Gauss-Newton method, after giving an initial guess  $\boldsymbol{\beta}_0$ , LMA calculates the shift vector as:

$$\Delta \boldsymbol{\beta} = (\mathbf{J}^T \mathbf{J} + \lambda \text{diag}(\mathbf{J}^T \mathbf{J}))^{-1} \mathbf{J}^T \Delta \mathbf{y} \quad (2.64)$$

Then, in an iteration step  $k + 1$ , parameters are updated as:

$$\boldsymbol{\beta}^{k+1} = \boldsymbol{\beta}^k + (\mathbf{J}^T \mathbf{J} + \lambda \text{diag}(\mathbf{J}^T \mathbf{J}))^{-1} \mathbf{J}^T \mathbf{r}(\boldsymbol{\beta}^k) \quad (2.65)$$

The iterative procedure stops either  $\mathbf{r}(\boldsymbol{\beta}^k)$  is close to zero or  $\Delta \boldsymbol{\beta}^k$  is tiny.

### 2.3.3 Robust Estimation methods

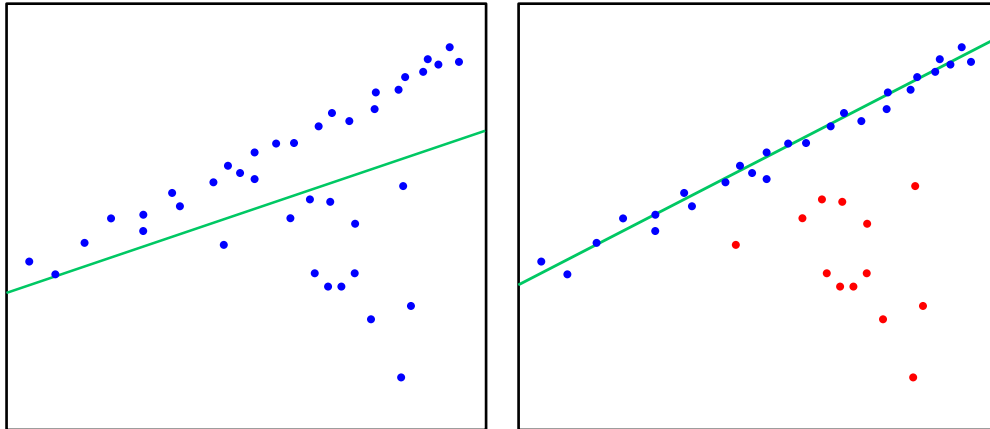
Robust estimation methods, or robust statistical methods, is designed to deal with outliers in data. Another motivation is to provide methods with good performance when there is a mixture of models. From statistical theory, M-estimator or least-median squares [Godambe 1991] are proposed. While here, we only describe RANSAC algorithm, a widely used robust estimation method developed from computer vision community.

#### 2.3.3.1 RANSAC algorithm

RANSAC (RANdom SAmple Consensus) [Fischler 1981] is a general parameter estimation approach designed to cope with a large proportion of outliers in input data. It is also a non-deterministic algorithm, which means that it gives a reasonable result by a certain probability.

In aforementioned least squares methods, all the observational data are assumed to be as *inliers* – data fit the model only subject to noise at some extent. However, *outliers*, data do not fit the model at all, are pervasive in real applications. The outliers maybe come from extreme values of noise, erroneous measurements, incorrect hypotheses about the interpretation of data, etc.

A simple example is line fitting in two dimensions from a set of observations contains both inliers and outliers. As drawn in Fig. 2.13, observational data of a 2D line are severely contaminated by outliers (marked in red in Fig. 2.13 (b)). If we directly estimate the 2D parameters by ordinary least squares, the result would be far from the true model, as shown in Fig. 2.13 (a). While for RANSAC, the outliers have no influences on the final results, as drawn in Fig. 2.13 (b).



(a) 2D line fitting by ordinary least squares (b) 2D line fitting by RANSAC with considering the outliers

Figure 2.13: The difference between ordinary least squares and RANSAC in 2D line fitting, when facing outliers

The procedure of RANSAC algorithm is show in Algorithm 1

---

**Algorithm 1** RANSAC algorithm
 

---

- 1: Randomly select the minimum number of points required to determine the model parameters, e.g. two points for 2D line fitting. These randomly chosen data are called *hypothetical inliers*.
  - 2: Solve the parameters of the model.
  - 3: All other data are then tested under the estimated model. Determine how many data points fit with the model with a predefined tolerance  $t$ . If it fits well, it is considered as a hypothetical inlier.
  - 4: If the fraction of the inliers exceeds a predefined threshold  $\tau$ , re-estimate the model parameters using all the hypothetical inliers.
  - 5: Otherwise, repeat 1 through 4 within a maximum iteration times  $K$ .
  - 6: **Return** the estimated parameters if the model is sufficiently good, or the number of iteration reaches to the maximum iteration times.
- 

A trivial problem for RANSAC is how to judge the estimated model. The criterion of the absolute errors of the inliers might make the algorithm pick the model fitting a minimal set of inliers (as the model will result in zero error estimation). Usually, the "best model" is typically chosen as the one with the largest consensus set. Therefore, the model with too few satisfied points is rejected in each step.

The number of iterations  $N$  is chosen high enough to ensure that the probability  $p_{ransac}$  (usually set to 0.99) that at least one of the sets of random samples contains only inliers. Thus,  $p_{ransac}$  gives the probability that the algorithm produces a useful result. Let  $p_{inlier}$  represent the probability that any selected data point is an inlier, that is,

$$p_{inlier} = \frac{\text{number of inliers in data}}{\text{number of points in data}} \quad (2.66)$$

and  $p_{outlier} = 1 - p_{inlier}$  the probability of observing an outlier.  $K$  iterations of minimum  $L$  number of points are required, where

$$1 - p_{ransac} = (1 - p_{inlier}^L)^K \quad (2.67)$$

and taking the logarithm of both sides, leads to

$$K = \frac{\log(1 - p_{ransac})}{\log(1 - (1 - p_{outlier})^L)} \quad (2.68)$$

## 2.4 Image Local Feature Detectors and Descriptors

Local feature detectors and descriptors are of the most important research areas in computer vision society. Numerous image features have been proposed during the past decades. Detecting specific visual features and associating them across different images are applied in a diverse computer vision tasks, including stereo vision [Tuytelaars 2000, Matas 2002], vision based SLAM [Se 2002], image stitching [Brown 2003, Li 2010] and object recognition [Schmid 1997, Sivic 2005]. According to the types of image feature's usages, there are three main categories:

- Detecting image features for semantic representation. For example, edges detected in aerial images often correspond to roads.
- Stable localizing points through different images. What the features actually represent is not really important, as long as their location is accurate and stable enough against changing environments. This maybe the most common usage applied broadly in camera calibration, 3D reconstruction, pose estimation, image registration and mosaicing. A classic example is using Kanade-Lucas-Tomasi(KLT) tracker [Shi 1994] to track image features.
- Image features and their descriptors can be used to interpreted images, such as object recognition, scene classification. In fact, what the feature descriptors really represent are unimportant, the goal is to analyze their statistical performances in an image database.

In this section, we only describe conventional feature detectors and descriptors and their usages in the second category. In essence, localizing a same feature between different images is to find its counterparts. A typical procedure is given as follows:

1. Local invariant feature detector is performed to identify a set of image locations (point, edge or region), which are stable against the variation of environment.
2. A vector carrying on certain visual information around each detected image feature are computed as feature descriptor.
3. Associating detected features between different images acquired under various situations. This process could be completed by feature tracking or matching feature descriptors.

### 2.4.1 Local Invariant Feature Detectors

[Tuytelaars 2008] gives a comprehensive survey about local invariant feature detectors. Here, we brief this survey and add some latest progresses on image feature detectors.

A local invariant feature is an image pattern which differs from its neighborhood. It can be in form of a point, an edge or a small image patch. The "invariant" requires relative constant detecting results under certain transformations, such as rotation,

scale, illumination changes etc. In general, the following properties are used to evaluate the quality of feature detectors:

- *Repeatability*: Given two images of the same object or scene acquired under different conditions, the percentage of features that occur in both images are defined as repeatability. The repeatability maybe the most important property, which represent the invariance and robustness of a feature detector to various transformations.
- *Quantity*: The number of detected features should be sufficiently large, such that a reasonable number of features are detected even on small objects. However, the optimal number of features depends on the application. Ideally, the number of detected features should be adaptable over a large range by a simple and intuitive threshold. The density of features should reflect the information content of the image to provide a compact image representation.
- *Accuracy*: The detected features should be accurately localized, both in image location, as with respect to scale and possibly shape. This property is particularly important in stereo matching, image registration and pose estimation, etc.
- *Efficiency*: Preferably, the detection of features in a new image should allow for time-critical applications.

Since feature detectors are the very fundament of computer vision, numerous approaches have been proposed. In this section, we only introduce several most representative methods.

#### 2.4.1.1 Corner Detectors

A corner can be defined as a point for which there are at least two dominant and different edge directions in its neighborhood. An example is drawn in Fig. 2.14. There are three points (pixels) – point *a* on the corner, point *b* on the edge and point *c* inside the object. For the point *c*, the surrounding pixels (within solid square) remain invariant in all direction. As for the point *b*, the neighboring pixels change in one direction (perpendicular to the edge). Whereas, the pixels around the point *a* differ from each other in all directions.

**Moravec Corner Detector:** It was Moravec [Moravec 1979] who firstly proposed a corner detector finding points that have local maximum in a directional variance measure. The proposed method analyzes a shifting window around a pixel in all directions. As mentioned before, if the pixel is on the corner, none of nearby patches will look similar. The corner strength is defined as the smallest *sum of squared differences* (SSD) between two patches. If this value is a maximum in local, then, a corner point is found. However, this detector is *anisotropic* as the intensity variation is only calculated at a discrete set of shifts. Therefore, the operator is not rotationally invariant, which will cause the detector to have poor repeatability rate.



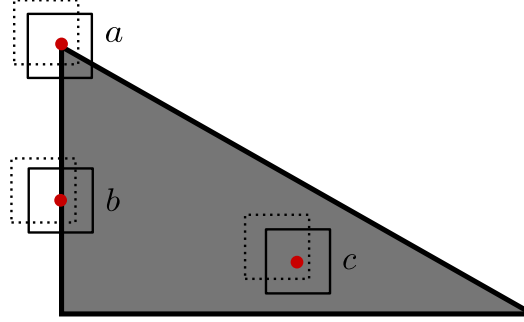


Figure 2.14: The difference of a point on corner, edge and inside

**Harris Corner Detector:** Harris and Stephens [Harris 1988], improved Moravec's method by considering the differential of the corner score with respect to direction directly, instead of using shifted patches. This corner score is called the *auto-correlation matrix*, which describes the gradient distribution in a neighborhood of a point. For every pixel  $(x, y)$  in an image  $I$ , this matrix is expressed as:

$$M(x, y) = \begin{bmatrix} \sum_{u,v} w_{u,v} \cdot [I_x(x_r, y_r)]^2 & \sum_{u,v} w_{u,v} \cdot I_x(x_r, y_r) I_y(x_r, y_r) \\ \sum_{u,v} w_{u,v} \cdot I_x(x_r, y_r) I_y(x_r, y_r) & \sum_{u,v} w_{u,v} \cdot [I_y(x_r, y_r)]^2 \end{bmatrix} \quad (2.69)$$

where  $I_x$  and  $I_y$  represent the derivatives of the image in  $x$  and  $y$  directions respectively,  $(x_r, y_r) = (x + u, y + v)$ , and  $w(u, v)$  is a window representing a weighting function, e.g. a binary rectangular window. [Harris 1988] proposed to use a Gaussian window  $w(u, v) = \exp(-(u^2 + v^2)/2\sigma^2)$  since it is *isotropic*. By analyzing the eigenvalues  $\lambda_1$  and  $\lambda_2$  of  $M(x, y)$ , the point can be classified as :

- If  $\lambda_1 \approx 0$  and  $\lambda_2 \approx 0$ , then, this pixel  $(x, y)$  has no interest.
- If  $\lambda_1 \approx 0$  and  $\lambda_2$  has large positive value, then, an edge is found.
- If  $\lambda_1$  and  $\lambda_2$  both have large positive values, then, a corner is found.

To reduce computational expense, [Harris 1988] proposed the following corner score, which depends on the eigenvalues, but avoid direct calculation:

$$\begin{aligned} c(x, y) &= \lambda_1 \lambda_2 - k \cdot (\lambda_1 + \lambda_2)^2 \\ &= \det(M(x, y)) - k \cdot \text{trace}^2(M(x, y)) \end{aligned} \quad (2.70)$$

where  $\det(*)$  and  $\text{trace}(*)$  denote the determinant and trace of a matrix respectively.  $k$  is determined empirically (usually in the range of 0.04-0.15). Therefore, Harris corner detector does not actually compute the eigenvalues of Harris matrix in Eq. 2.69. Instead, it calculates the Harris corner score.

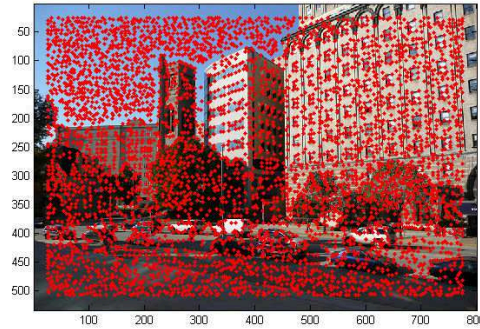
**Shi-Tomasi's "Good Feature to Track":** Shi and Tomasi [Shi 1994] theoretically analyzed which feature is good enough to track. Based on the assumption of an affine

image transformation, they found that it is more convenient and accurate to use the smallest eigenvalue of the autocorrelation matrix as the corner score.

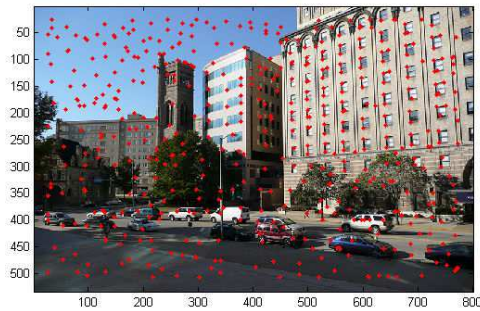
$$c(x, y) = \min(\lambda_1, \lambda_2) > \theta \cdot \max_{x, y} c(x, y) \quad (2.71)$$

where  $\theta$  is a predefined percentage factor to control the minimum score. Compared to the Harris score (Eq. 2.70), this requires an additional square root computation on each pixel.

**Non-Maximum Suppression:** Performing corner detectors or other detectors (e.g. object detectors) in a manner of scanning window usually results in multiple responses within high interest regions. This effect makes detecting results too concentrated in a region. A standard method to deal with this is to remove detector responses in the neighborhood of detections with locally maximal score, which is called *Non-Maximum Suppression* (NMS). For each corner candidate within a region, NMS sets the corner score for this point to zero if its corner score is not larger than the corner score of all candidates within this region. After non-maximum suppression, the corners are the locally maximum valued pixels. An example is shown in Fig. 2.15



(a) Harris corner detecting results before Non-Maximum Suppression



(b) Harris corner detecting results after Non-Maximum Suppression

Figure 2.15: The effects of Non-Maximum Suppression in corner detection [Dominguez 2007]

**Features from Accelerated Segment Test (FAST):** Rosten and Drummond [Rosten 2005, Rosten 2006] developed a high-speed corner detector: Features from Accelerated Segment Test, coined as FAST. This method performs on a discretized circle (Bresenham circle) around a candidate point  $p$  as shown in Fig. 2.16. The black point in the center is the candidate point  $p$ . The 16 grey points are the discretized approximation of the outlined circle around it.

The FAST detector classifies  $p$  as a corner if there is a large number of pixels on a circle of fixed radius around  $p$  such that these pixels are all significantly brighter, or darker, than  $p$  by a threshold  $t$ . The algorithm is further accelerated by a trained decision tree to operate as few pixels as possible. With this decision tree, only 2.26 pixels in average are tested [Rosten 2006]. To perform a Non-Maximum Suppression,

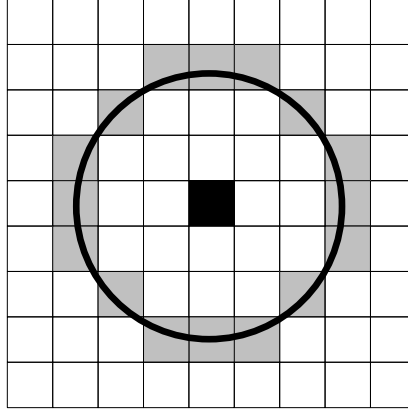


Figure 2.16: Bresenham circle used in FAST detector [Rosten 2006]

the following corner score is computed for each candidate point:

$$c(p) = \max\left\{\sum_{q \in S_+} |I_q - I_p| - t, \sum_{q \in S_-} |I_q - I_p| - t\right\} \quad (2.72)$$

where  $S_+$  is the subset of pixels on the circle that are brighter than  $p$  (by the threshold  $t$ ) and  $S_-$  is the subset of pixels that are darker than  $p$  (by the threshold  $t$ ).

#### 2.4.1.2 Interest Point Detectors

Interest point detector is an extension of corner point. A point can be characterized as an interest point if:

- The local visual information (e.g. texture, gradient) around the point is rich.
- Its position is stable under certain image transformations, such as scale/illumination/affine changes.

A corner point detector (described in last section) is a special kind of interest point detectors that it is mainly designed for robust image feature tracking. Here, we

discuss more general cases of detecting interest points (or regions) for more general tasks. Please note that in the history of computer vision, the applications of terminologies "blob detector" and "interest point detector" are overlapping. Output of blob detectors are usually well-defined point positions, which may correspond to local extremas in an operation applied in local regions.

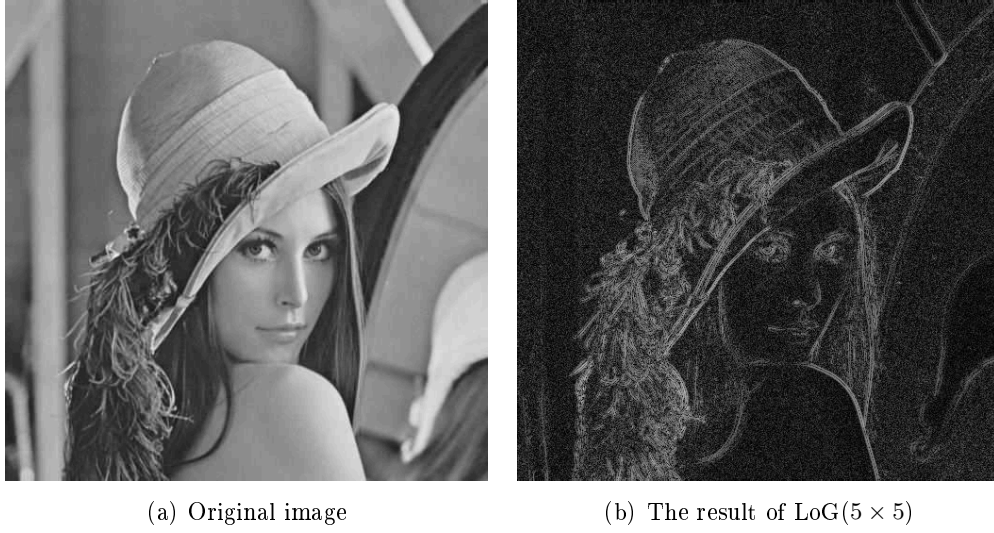


Figure 2.17: The effects of Laplacian of Gaussians

**Hessian Matrix and Laplacian of Gaussians (LoG):** The Harris matrix in Eq. 2.69 is about the first order derivatives of an image. Considering the following  $2 \times 2$  *Hessian* matrix for an image  $I(x, y)$ :

$$H(x, y, \sigma) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix} \quad (2.73)$$

with  $L_{xx}$ ,  $L_{xy}$  and  $L_{yy}$  refer to the second order derivatives of a Gaussian smoothed image at  $(x, y)$ . Gaussian smoothing is performed due to second order derivative's extreme sensibility to noise:

$$L(x, y; \sigma) = G(x, y, \sigma) * I(x, y) \quad (2.74)$$

where  $G(x, y, \sigma)$  is a Gaussian kernel at scale  $\sigma$ :

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/(2\sigma^2)} \quad (2.75)$$

Hessian matrix encodes rich information about local image structure.

Laplacian of Gaussians (LoG) is probably the first commonly used blob feature detector. It is defined by the trace of Hessian matrix:  $\nabla^2 L = L_{xx} + L_{yy}$ . LoG preserves the edges and corners on an image, as seen in Fig. 2.17. These edges and corners are good candidates for a good feature detector.

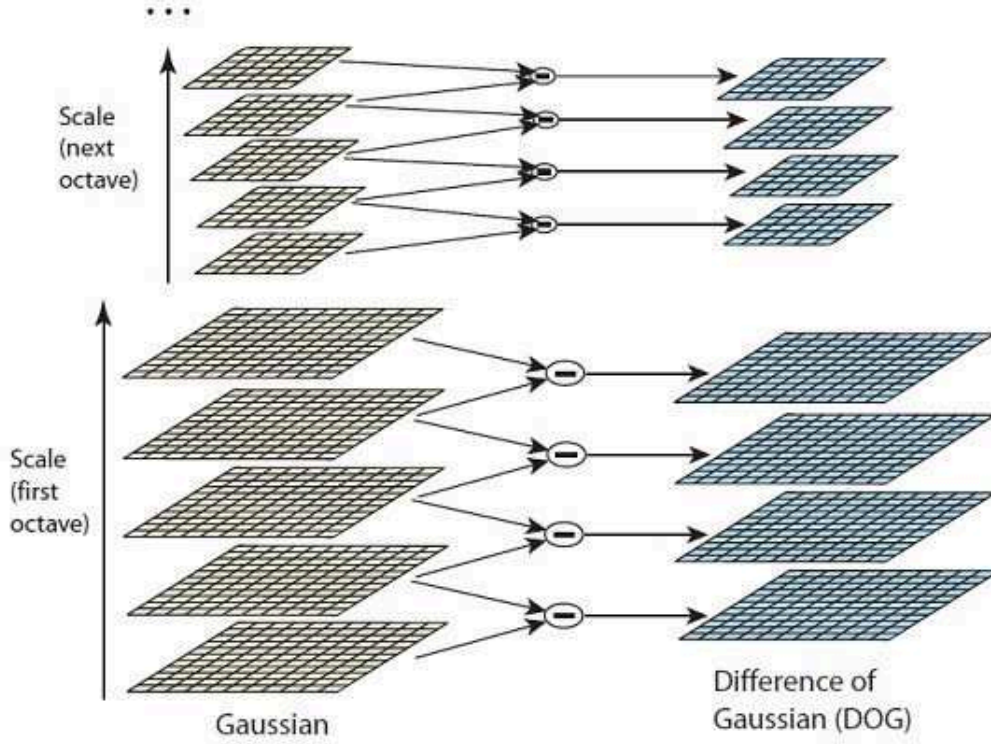


Figure 2.18: Creating octaves of scale spaces to find local extremas [Lowe 2004]

However, the LoG operator strongly depends on the size of blob in the image and the size of Gaussian kernel used for smoothing. In order to automatically detect blobs of different scale in the image, a multi-scale approach is proposed. A straightforward approach is to use the *scale-normalized Laplacian operator* (normalized LoG):

$$\nabla_{norm}^2 L(x, y; \sigma^2) = \sigma^2 (L_{xx} + L_{yy}). \quad (2.76)$$

This operator is to detect scale-space extremas, which are local maxima/minima of  $\nabla_{norm}^2 L(x, y; \sigma^2)$  with respect to both space and scale changes [Lindeberg 1994, Bretzner 1998]. In practice, computing second order derivatives is computationally intensive. Several approximations described in the following (*Difference of Gaussians* and *Fast Hessian*) are proposed to speed up the computation time.

**Difference of Gaussians (DoG):** The Difference of Gaussians detector (DoG), as a part of SIFT [Lowe 2004], is proposed to approximate normalized LoG. Specifically, a DoG operator is given by

$$\begin{aligned} D(x, y, \sigma) &= L(x, y, \lambda\sigma) - L(x, y, \lambda\sigma) \\ &= (G(x, y, k\sigma) - G(x, y, \lambda\sigma)) * I(x, y) \end{aligned} \quad (2.77)$$

where  $G(x, y, \sigma)$  is the Gaussian kernel defined in Eq. 2.75.  $\lambda$  is a constant multiplicative factor. The convolution results  $L(x, y, k\sigma)$  with different  $k$  create a *scale space*. SIFT uses octaves of scale spaces to finding extremas of DoG. The approach to quickly construct  $D(x, y, \sigma)$  in several scale spaces is shown in Fig. 2.18.

For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left of Fig. 2.18. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right of Fig. 2.18. After processing each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated. Key points are then extracted at local minima/maxima of the DoG images through scales.

The DoG approximation greatly reduces the computation cost – it can be computed by a simple image subtraction. As an approximation of normalized LoG operator, DoG is scale invariant.

**Fast Hessian Detector** Fast Hessian detector, proposed by [Bay 2008] as a basis for SURF (Speeded Up Robust Features), is based on the determinant of Hessian matrix with scale factor  $\sigma$ :

$$\det H(x, y, \sigma) = L_{xx}(x, y, \sigma) \cdot L_{yy}(x, y, \sigma) - L_{xy}^2(x, y, \sigma) \quad (2.78)$$

To further accelerate the computation of Gaussian second order derivatives, [Bay 2008] approximates them with simple box filters (Fig. 2.19). In the figure,

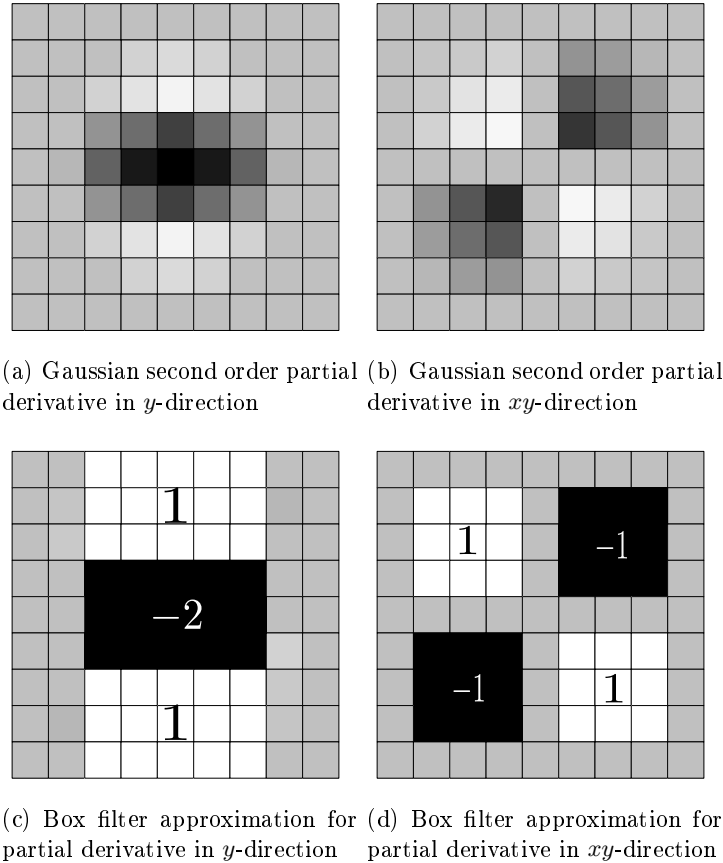


Figure 2.19: Box filters as fast approximation to second order derivatives of Gaussians. [Bay 2008]

black and white denote weights in the pixels, grey regions have weight equal zero. Outputs of box filters can be computed very fast by integral images [Viola 2001] in a constant time. Let  $D_{xx}$ ,  $D_{xy}$  and  $D_{yy}$  be the results of the box filters drawn in Fig. 2.19, the determinant of Hessian matrix is approximated by:

$$\det[H(x, y, \sigma)] \approx D_{xx}(\sigma) \cdot D_{yy}(\sigma) - (0.9D_{xy}(\sigma))^2 \quad (2.79)$$

Eq. 2.79 is taken as a measure of key point, and performed in the image. The responses are stored in a response map, where local extremas are found and refined. In [Bay 2008], it reports that SURF is more than five times faster than DoG.

**Center-Surround Extrema (CenSurE):** CenSurE [Agrawal 2008] is proposed to execute the task of matching two images in real time for camera motion estimation, especially in difficult environment where there is large image motion between frames. While SIFT [Lowe 2004] approximates LoG with difference of Gaussians, CenSurE adopts a further approximation, utilizing bi-level center-surround filters, i.e., with filter values  $-1$  and  $1$ . Several bi-level examples are shown in Fig. 2.20. The circular filter in the left-most of Fig. 2.20 is the most similar to Laplacian, but is hard to compute. The other listed filters from left to right are in an order of decreasing performance and computational cost.

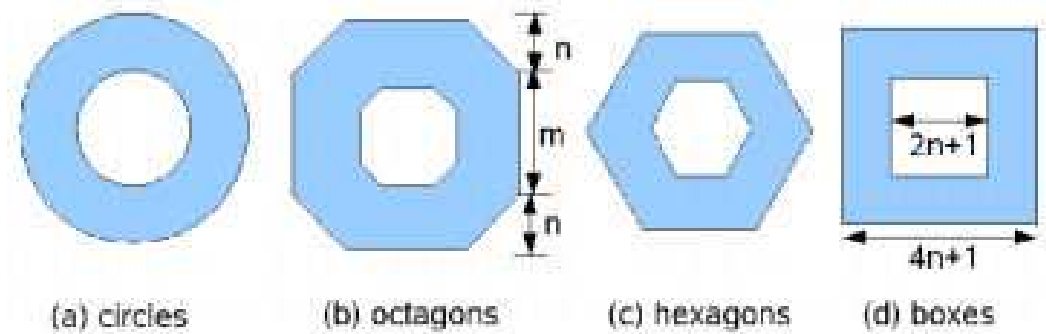


Figure 2.20: CenSurE's bi-level filters [Agrawal 2008]

The computation of bi-level filters is even further speeded up by *slanted integral image*, as a modification of original integral image [Viola 2001]. Following the output of a bi-level filter, a  $3 \times 3 \times 3$  neighborhood non-maximum suppression and an edge/line response suppression are performed to refine the results. The authors claimed that CenSurE outperforms the other detectors (e.g. SIFT, SURF, FAST, Harris corner) in stability and accuracy.

### 2.4.2 Feature descriptors

After extracting interest points, a descriptor is created to identify and match them between different images. The most straightforward description for a feature point is the intensity appearance within a patch around the point itself. Similarity metrics, e.g. the *sum of absolute differences* (SAD), the *sum of squared differences* (SSDs), or the *normalized cross correlation* (NCC) can be used:

$$\begin{aligned}
 M_{SAD} &= \sum_{(i,j) \in p} |I_1(i,j) - I_2(x+i, y+j)| \\
 M_{SSD} &= \sum_{(i,j) \in p} (I_1(i,j) - I_2(x+i, y+j))^2 \\
 M_{NCC} &= \frac{\sum_{(i,j) \in p} I_1(i,j) I_2(x+i, y+j)}{\sqrt{\sum_{(i,j) \in p} I_1^2(i,j) \sum_{(i,j) \in p} I_2^2(x+i, y+j)}}
 \end{aligned} \tag{2.80}$$

In many cases, local appearance around a feature point is not enough, especially when encountering large changes in orientation, scale or viewpoint. In fact, none of the SSD, SAD and NCC are invariant to any of these changes. Therefore, their use is limited to images taken at nearby positions.

Exploring feature descriptors stable in various changes has been conducted for decades in the computer vision field. In the following, we will introduce several popular feature descriptors.

**SIFT descriptor:** One of the most popular descriptor for point features is the SIFT [Lowe 2004]. Fig. 2.21 illustrates how the descriptor is computed. At first,

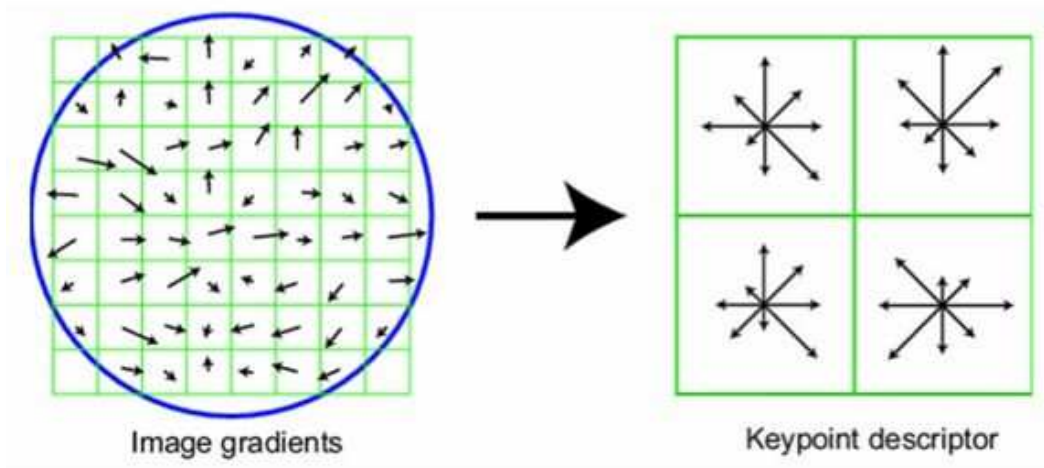


Figure 2.21: SIFT descriptor [Lowe 2004]

SIFT descriptor computes the gradient magnitude and orientation at each image sample point in a surrounding region of a keypoint, as shown in the left of Fig. 2.21. Then, a Gaussian weighting function is used to assign a weight to the magnitude



of each sample point, as illustrated by a circular window on the left image of Fig. 2.21. At last, the gradient magnitudes and orientations of all the sample points are accumulated into orientation histograms over  $4 \times 4$  subregions, as shown in the right of Fig. 2.21. The left image shows eight directions for each orientation histogram, and the length of arrow denotes the magnitude of gradient. Finally, SIFT descriptor forms a vector containing the normalized values (normalized to a unit length) in all orientations. Fig. 2.21 shows  $2 \times 2$  grids of histograms. Whereas in [Lowe 2004]'s implementation, the authors use a  $4 \times 4$  array of histograms with  $4 \times 4 \times 8 = 128$  elements in total.

SIFT descriptor is proved to be stable against changes in illumination, rotation, and scale, and even up to 60 changes in viewpoint. Although proposed in 2004, it still among the top-performanced feature points [Gauglitz 2011]. In general, the SIFT descriptor can be computed for corner or blob detectors. However, its performance will decrease on corners because that corners are not as distinctive as for blob detectors.

**SURF descriptor:** Although SIFT achieve great success, its 128-dimensional feature descriptor costs intensive computation in feature matching or indexing. SURF descriptor [Bay 2008] simplifies SIFT descriptor while keeping almost the same matching performance.

[Bay 2008] firstly computes an orientation for each keypoint. Haar wavelet filters (see in the left figure of Fig. 2.22 (a)) in  $x$  and  $y$  directions are convoluted with a circular region around the keypoint. Similar to SIFT, the filter outputs are weighted by a Gaussian and then represented as 2-dimensional vectors, which are summed up with a rotating angular window. The longest resulting vector determines the orientation of the keypoint. Then, a square region centered at the keypoint, and

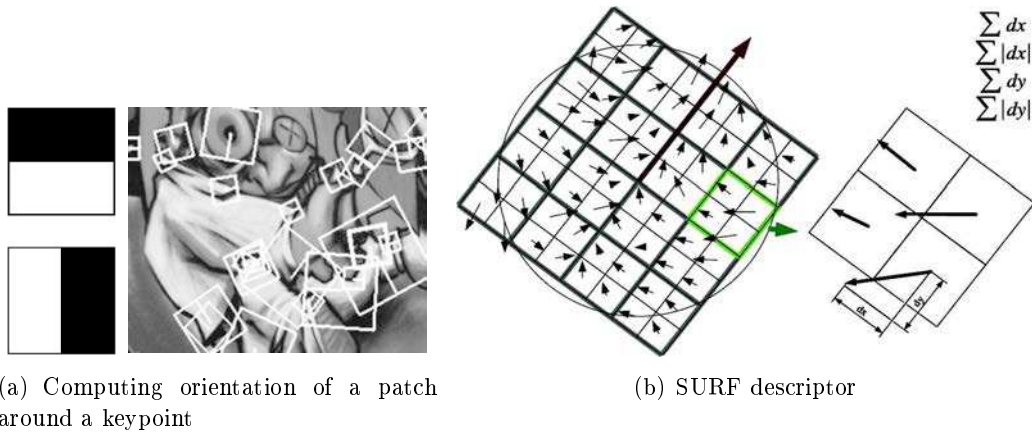


Figure 2.22: SURF descriptor [Bay 2008]

oriented along the computed orientation is extracted (as seen in the right image of Fig. 2.22 (a)). Similar to SIFT, the square region around the keypoint is split up

into  $4 \times 4$  subregions. For each subregion, the following feature vector is computed:

$$\mathbf{v} = [\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|] \quad (2.81)$$

where  $d_x$  and  $d_y$  are responses of Haar wavelet filter in  $x$  and  $y$  directions. Hence, this results in a descriptor vector for all  $4 \times 4$  subregions of a length equals to 64. This process is illustrated in Fig. 2.22 (b).

**ORB descriptor:** In recent years, several binary descriptors have been designed to achieve high computation speed to meet requirements of mobile visual computing. Here, we will introduce two typical binary descriptors: Oriented FAST and Rotated BRIEF (ORB) [Rubblee 2011] and Binary Robust Invariant Scalable Key-points (BRISK) [Leutenegger 2011].

ORB feature [Rubblee 2011] can be viewed as an improved combination of FAST and BRIEF [Calonder 2010] such that it is based on an oriented FAST corner detector plus a rotated BRIEF descriptor. The detector component relies on *intensity centroid* [Rosin 1999] to compute orientation for every detected FAST feature point. As for the feature descriptor, the BRIEF descriptor is a bit string description of an image patch constructed from a set of binary intensity tests. Consider a smoothed image path,  $\mathbf{p}$ . A binary test  $\tau$  is defined by:

$$\tau(\mathbf{x}, \mathbf{y}; \mathbf{p}) = \begin{cases} 1 & : \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & : \mathbf{p}(\mathbf{x}) > \mathbf{p}(\mathbf{y}) \end{cases}$$

where  $\mathbf{p}(\mathbf{x})$  is the intensity at point  $\mathbf{x}$ . The feature is defined as a vector of  $n$  binary tests:

$$f_n(\mathbf{p}) = \sum_{1 \leq i \leq n} 2^{i-1} \tau(\mathbf{p}; \mathbf{x}_i, \mathbf{y}_i) \quad (2.82)$$

In order to let BRIEF be invariant to in-plane rotation, [Rubblee 2011] proposed to steer BRIEF by the orientation of keypoints. For any feature set of  $n$  binary tests at location  $(x_i, y_i)$ , define the  $2 \times n$  matrix:

$$S = \begin{pmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \end{pmatrix} \quad (2.83)$$

Using the patch orientation  $\theta$  estimated by intensity centroid, and the corresponding rotation matrix  $\mathbf{R}_\theta$ , we get a "steered" version  $\mathbf{S}_\theta = \mathbf{R}_\theta \mathbf{S}$ . The steered BRIEF operator becomes:

$$g_n(\mathbf{p}, \theta) := f_n(\mathbf{p}) | (\mathbf{x}_i, \mathbf{y}_i) \in S_\theta \quad (2.84)$$

To recover from the loss of variance in steered BRIEF and to reduce correlation among the binary tests, [Rubblee 2011] develops a learning method for choosing a good subset of binary tests. The result is called rBRIEF, which significantly improves the variance and correlation of steered BRIEF. The rBRIEF descriptor is taken as ORB's descriptor. The authors demonstrate that ORB is at two orders of magnitude faster than SIFT, while performing as well in many situations.

**BRISK descriptor:** Another binary descriptor is BRISK, which is proposed in [Leutenegger 2011]. The detector of BRISK is an extension of FAST detector by searching maxima not only in the image plane, but also in scale-space using the FAST score in Eq. 2.72 as a measure for salience.

Similar to ORB, BRISK descriptor is also composed as a binary string by concatenating the results of simple brightness comparison tests. The direction of each keypoint is also computed to achieve rotation invariance. Furthermore, BRISK descriptor selects the intensity comparisons maximizing descriptiveness between different points. The key concept of BRISK descriptor is using a pattern (as illustrated

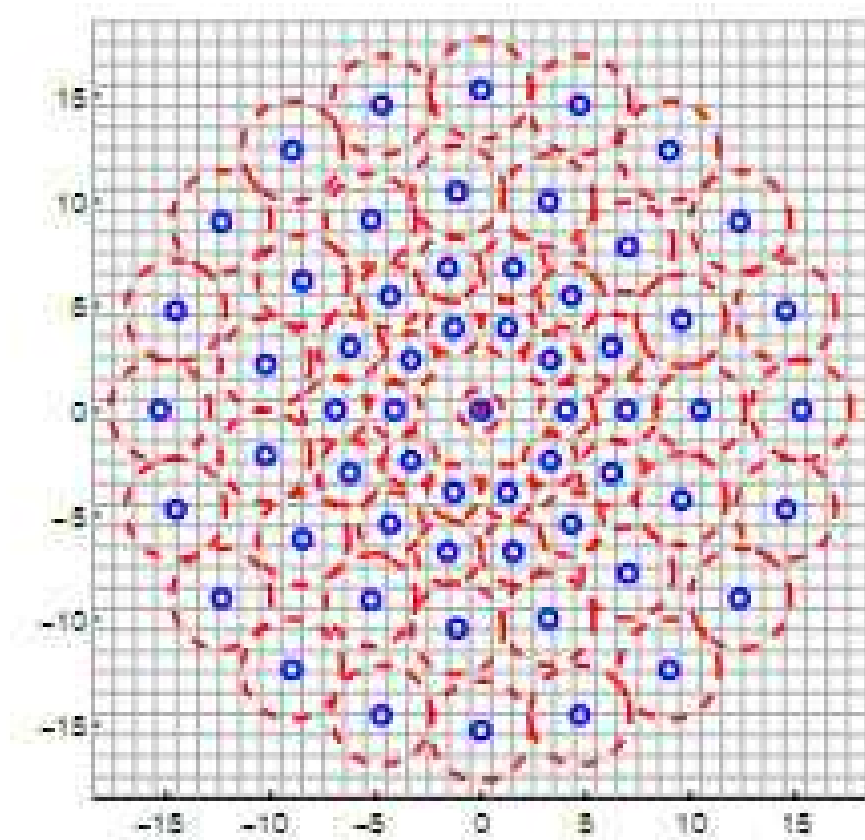


Figure 2.23: BRISK sample pattern [Leutenegger 2011]

in Fig. 2.23) to sample the neighborhood of the keypoint. The pattern defines  $N$  sampling locations equally spaced on concentric circles with the keypoint. Let us consider one of the  $N \cdot (N-1)/2$  sampling-point pairs  $(\mathbf{p}_i, \mathbf{p}_j)$ . The smoothed intensity values at these points which are  $I(\mathbf{p}_i, \sigma_i)$  and  $I(\mathbf{p}_j, \sigma_j)$  (The Gaussian smoother in BRISK is different to the position of  $\mathbf{p}$ ). The local gradient  $\mathbf{g}(\mathbf{p}_i, \mathbf{p}_j)$  is:

$$\mathbf{g}(\mathbf{p}_i, \mathbf{p}_j) = (\mathbf{p}_j - \mathbf{p}_i) \cdot \frac{I(\mathbf{p}_j, \sigma_j) - I(\mathbf{p}_i, \sigma_i)}{\|\mathbf{p}_j - \mathbf{p}_i\|^2} \quad (2.85)$$

Considering the set  $\mathcal{A}$  of all sampling-point pairs, the authors define a subset  $\mathcal{S}$  of short-distance pairing and another subset  $\mathcal{L}$  of long-distance pairings:

$$\begin{aligned}\mathcal{S} &= \{(\mathbf{p}_i, \mathbf{p}_j) \in \mathcal{A} \mid \|\mathbf{p}_j - \mathbf{p}_i\| < \sigma_{max}\} \subseteq \mathcal{A} \\ \mathcal{L} &= \{(\mathbf{p}_i, \mathbf{p}_j) \in \mathcal{A} \mid \|\mathbf{p}_j - \mathbf{p}_i\| > \sigma_{min}\} \subseteq \mathcal{A}\end{aligned}\quad (2.86)$$

The pattern direction is estimated through the point pairs in  $\mathcal{L}$ :

$$\mathbf{g} = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{L} \cdot \sum_{\mathbf{p}_i, \mathbf{p}_j \in \mathcal{L}} g(\mathbf{p}_i, \mathbf{p}_j) \quad (2.87)$$

where  $L$  is the number of point pairs. In order to make a rotation and scale normalized descriptor, BRISK rotates the sampling pattern by  $\alpha = \arctan 2(g_x \cdot g_y)$  around a keypoint. The bit-string descriptor is created by comparing all the short distance intensity comparisons:

$$b = \begin{cases} 1 & I(\mathbf{p}_j^\alpha, \sigma_j) > I(\mathbf{p}_i^\alpha, \sigma_i) \quad (\mathbf{p}_i^\alpha, \mathbf{p}_j^\alpha) \in \mathcal{S} \\ 0 & otherwise \end{cases}$$

The experiments in [Leutenegger 2011] reveal BRISK's high quality performance as in state-of-art algorithms, with a lower computational cost.

### 2.4.3 Associating feature points through images

As described in the beginning of Sec. 2.4, one of the most important need when using image feature points is to stable localizing the same point through different image. Many computer vision applications, e.g. stereo matching, pose estimation, augmented reality and object recognition, are relied on precise feature point associating. In this section, we introduce two approaches to associate feature points: the first is feature tracking, the second is feature matching.

#### 2.4.3.1 Kanade-Lucas-Tomasi Tracking (KLT tracker)

KLT tracker is a classic image feature point tracking algorithm in computer vision area. It is based on the early work of [Lucas 1981], then is developed fully by [Tomasi 1991], and is explained in details in [Shi 1994]. Later, [Bouguet 2000] gives a pyramidal implementation to cope with large displacement.

Let two images  $I(\mathbf{x}) = I(x, y)$ ,  $J(\mathbf{x}) = J(x, y)$  refer to the first image and second one, respectively. For an image point  $\mathbf{u} = [u_x, u_y]$  on the first image  $I$ , KLT tracker is to find its corresponding location  $\mathbf{v} = \mathbf{u} + \mathbf{d} = [u_x + d_x, u_y + d_y]^T$  on the second image  $J$ . Let  $\mathbf{W}(\mathbf{x}; \mathbf{p})$  be the warp from image  $I$  to  $J$ , where  $\mathbf{p}$  is a vector of parameters. The best alignment minimizes image dissimilarity:

$$\sum [I(\mathbf{W}(\mathbf{x}; \mathbf{p})) - J(\mathbf{x})]^2 \quad (2.88)$$

Eq. 2.88 expresses nonlinear optimization. Therefore, an iterative method is used to solve  $\mathbf{p}$ . Assume that a  $\mathbf{p}$  is known and the best increment  $\Delta\mathbf{p}$  is sought that the problem is modified as minimizing the following quantities:

$$\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x}; \mathbf{p} + \Delta\mathbf{p})) - J(\mathbf{x})] \quad (2.89)$$

The above problem is iteratively solved with respect to  $\Delta\mathbf{p}$ :  $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$ . Eq. 2.89 can be linearized by first order Taylor expansion:

$$\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x}; \mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - J(\mathbf{x})]^2 \quad (2.90)$$

$\nabla I = [\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}]$  is the gradient image computed at  $\mathbf{W}(\mathbf{x}; \mathbf{p})$ . The term  $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$  is the *Jacobian* of the warp. Computing the derivative of Eq. 2.90 with respect to  $\Delta\mathbf{p}$ , we obtain:

$$2 \sum_{\mathbf{x}} [\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^T [I(\mathbf{W}(\mathbf{x}; \mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - J(\mathbf{x})] \quad (2.91)$$

Setting Eq. 2.91 to zero yields:

$$\nabla \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} [\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^T [J(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}; \mathbf{p}))] \quad (2.92)$$

where  $\mathbf{H}$  is the Hessian matrix:

$$\mathbf{H} = \sum_{\mathbf{x}} [\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^T [\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}] \quad (2.93)$$

By putting the above equations in an iterative approach, we get a whole description of KLT tracker, as summerized by Algorithm 2

---

**Algorithm 2** The Lucas-Kanade algorithm

---

- 1: **repeat**
  - 2:   Warp  $I$  with  $\mathbf{W}(\mathbf{x}; \mathbf{p})$
  - 3:   Warp the gradient  $\nabla I$  with  $\mathbf{W}(\mathbf{x}; \mathbf{p})$
  - 4:   Evaluate the Jacobian  $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$  and compute the steepest descent image  $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
  - 5:   Compute the Hessian  $\mathbf{H}$  according to Eq. 2.93
  - 6:   Compute  $\Delta\mathbf{p}$  according to Eq. 2.92
  - 7:   Update the parameters  $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$
  - 8: **until**  $\|\Delta\mathbf{p}\| < \varepsilon$
- 

### 2.4.3.2 Feature matching

Similar to feature tracking, the process of feature matching is also for finding corresponding features in different images. The difference is that, feature matching is to find correspondences between existed feature points, feature tracking is to compute

a feature point's new location in a new image. Feature matching is achieved by comparing feature descriptors between different feature points. Usually, a similarity metric, such as SSD, NCC, or Euclidean distance between two feature vectors, is computed for searching correspondences.

In essence, the feature matching problem is a *Nearest Neighbor (NN) search* problem, or *Post Office Problem* as:

**Definition 1.** Given a set  $P \in \mathbb{R}^d$  of  $n$  points and a query point  $q \in \mathbb{R}^d$ , the nearest neighbor search of  $q$  is:

$$NN(q) = \arg \min_{p \in P} |p - q| \quad (2.94)$$

where  $q$  is a feature point in  $d$ -dimensional feature space. The most simple solution of NN search is to compare all the candidate points with the query point  $q$ , find a point with the highest similarity score or the closest distance.

In feature matching, simple NN search may result with features in the second image matching with more than one feature in the first image. A *mutual consistency check* is usually used to eliminate this effect by reversely comparing the features in the second image with features in the first image. Only pairs of corresponding features matched in both directions are accepted as correct matching.

However, disadvantage of exhaustive NN search is obvious: its computation grows in a quadratical speed when the number of candidates increases. To speed up the searching process, several indexing structures, such as multidimensional search tree (e.g. k-d tree [Friedman 1977]) or hash table [Lamdan 1988], are proposed to rapidly search for features near a given feature. An example of 3-dimensional k-d tree is shown in Fig. 2.24. It can achieve efficiently NN searching by partitioning feature spaces into indexed leaves.

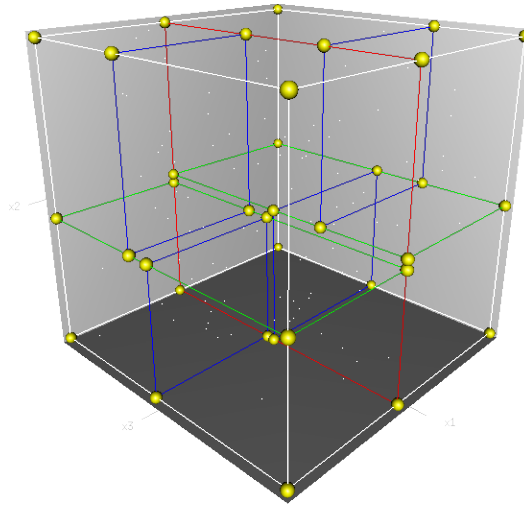


Figure 2.24: A 3-dimensional k-d tree

6

<sup>6</sup>[https://en.wikipedia.org/wiki/K-d\\_tree](https://en.wikipedia.org/wiki/K-d_tree)

Another feature matching strategy is to search for potential correspondences in an expected region in the second image. The size of regions can be predefined or predicted by the motion model of feature points. More works can be referred in [Chili 2008, Handa 2010].

## 2.5 Conclusion

In this chapter, we firstly describe the stereo vision measurement model. Since our stereo vision system consists of two quasi-parallel monocular cameras, the stereo vision measurement model is based on monocular camera pinhole model. The distortion model of the lens is also considered. The stereo vision model is very important and forms the basis of many further applications in the following chapters.

In the second, we review second linear/non-linear least squares methods. These methods are commonly used to solve estimation problems. For linear least square methods, ordinary and total least squares methods are introduced. Ordinary least squares are widely used in many computer vision problems, while the total least squares is an improved version of least squares when considering all the errors. For non-linear least squares, Gaussian-Newton method and Levenberg-Marquardt algorithm are described. We will use these two methods to estimate parameters in the following chapters.

At last, we present an overview of feature detectors/descriptors. In literature, a lot of feature detectors/descriptors are proposed because of wide applications. The number of proposed feature detectors/descriptors is so big that many researchers often feel confused. We try to summarize the development of feature detectors/descriptors by different categories. In the following chapter, we will analyze their performances in visual odometry.

# Stereo Vision based Ego-motion Estimation

---

## Contents

<b>3.1 Introduction</b>	<b>48</b>
<b>3.2 Circular Feature Points Detection and Association</b>	<b>50</b>
3.2.1 Circular Feature Point Association by KLT Tracker	51
3.2.2 Circular Feature Point Association by Matching	51
<b>3.3 Ego-motion Computation</b>	<b>54</b>
3.3.1 3D-2D Constraint based Ego-motion Estimation	54
<b>3.4 Experiments of Stereo Visual Odometry</b>	<b>57</b>
3.4.1 Comparing Different Feature Association Approaches	57
3.4.2 Stereo Visual Odometry in Urban Environment	62
<b>3.5 Conclusion and Future Works</b>	<b>70</b>

---

In this chapter, we provide a stereo vision based method to estimate the ego-motion of a moving intelligent vehicle in urban environments. Estimation of self-motion is important for an intelligent vehicle since:

- It provides a complementary localizing approach for GPS. In urban environments, GPS signals could disappear or be disturbed due to buildings, tunnels, etc. Without any prior knowledge of the environment nor a predefined motion model of the vehicle, vision based ego-motion estimation can estimate the path of a camera-equipped vehicle by calculating the ego-motion between consecutive images in a video flow. Hence, vision based ego-motion estimation could be used for vehicle localization when GPS signals are not available.
- It is the fundament of further analyzing of a dynamic environment. When the intelligent vehicle is driving in a dynamic environment, understanding self-motion is the first step before performing independent moving object detection, tracking and recognition.

In this chapter, we present a stereo vision based method to estimate the ego-motion of a moving intelligent vehicle. The contributions are comparing tracking based approaches with matching based approaches, as well as performances different feature detectors. The objective is to select the best feature point detector and the best circular point association for urban environment perception.



### 3.1 Introduction

Vision based ego-motion estimation, or, *visual odometry* (VO), computes movement of a vehicle by the outputs of on-board single or multiple cameras. Usually, it is used in the field of robotics or intelligent vehicles. Visual odometry operates by incrementally estimating the pose of the vehicle through the changes of images caused by the motion. Compared to global positioning system (GPS), visual odometry could work in more general environments, such as indoor, underwater and even in the planet Mars [Matthies 1987]. Advantages of visual odometry with respect to the other navigation sensors, e.g. inertial measurement units (IMU), laser range finder, is that it is not affected by wheel slip and it provides more accurate trajectory estimates in a low expense.

In literature, visual odometry (VO) is a particular case of *structure from motion* (SFM) [Dellaert 2000]. For an image sequence captured by a moving camera, SFM not only recovers the camera's trajectory, but also reconstructs the 3D structures of all the vision measurements. While in many applications, full 3D reconstruction is not necessary. Visual odometry focuses only on accurately estimating 3D motion of the camera frame by frame.

Approaches to visual odometry range from dense optical flow [McCarthy 2004], matching sparse salient image regions in consecutive images, to semiparametric regression from sparse optical flow [Guizilini 2012] proposed recently. In most navigation/localization applications, real-time performance is a requirement. Hence, sparse correspondence based methods are usually preferable to the other kinds of methods. It was Moravec [Moravec 1980] who firstly proposed a vision-based method to estimate a planetary rover's ego-motion. The proposed method is based on tracking sparse salient points detected in stereo image pairs. Similar early researches ([Matthies 1987, Matthies 1989, Olson 2000]) in visual odometry were driven by NASA Mars exploration. [Moravec 1980]'s method firstly detects corner points (Moravec corner point in Sec. 2.4.1.1) in the images of a sliding camera. Then, 3D positions of the corner points are computed by triangulation. Finally, rigid motion information is estimated by aligning the triangulated 3D points between two consecutive frames. As a summary, [Moravec 1980] proposed a pipeline to solve visual odometry problem based on sparse feature points – its main functioning parts remain unchanged today. The framework of a typical visual odometry is drawn in Fig. 3.1. At first, certain kind of salient feature points are detected and then associated through consecutive input images. Based on the associated pairs of feature points, 2D-2D/3D-2D/3D-3D constraints can be utilized to estimate moving information. Sometimes, a local optimization method (e.g. bundle adjustment [Bill 1999]) can be optionally chosen to improve the results.

Under the framework in Fig. 3.1, many researches have been conducted to improve visual odometry's accuracy and robustness. Since a binocular stereoscopic system is mounted in our platform, we only review the stereo vision based visual odometry methods. [Matthies 1987, Matthies 1989] incorporate error covariance matrix of triangulated feature points into motion estimation step. For this reason, they ob-

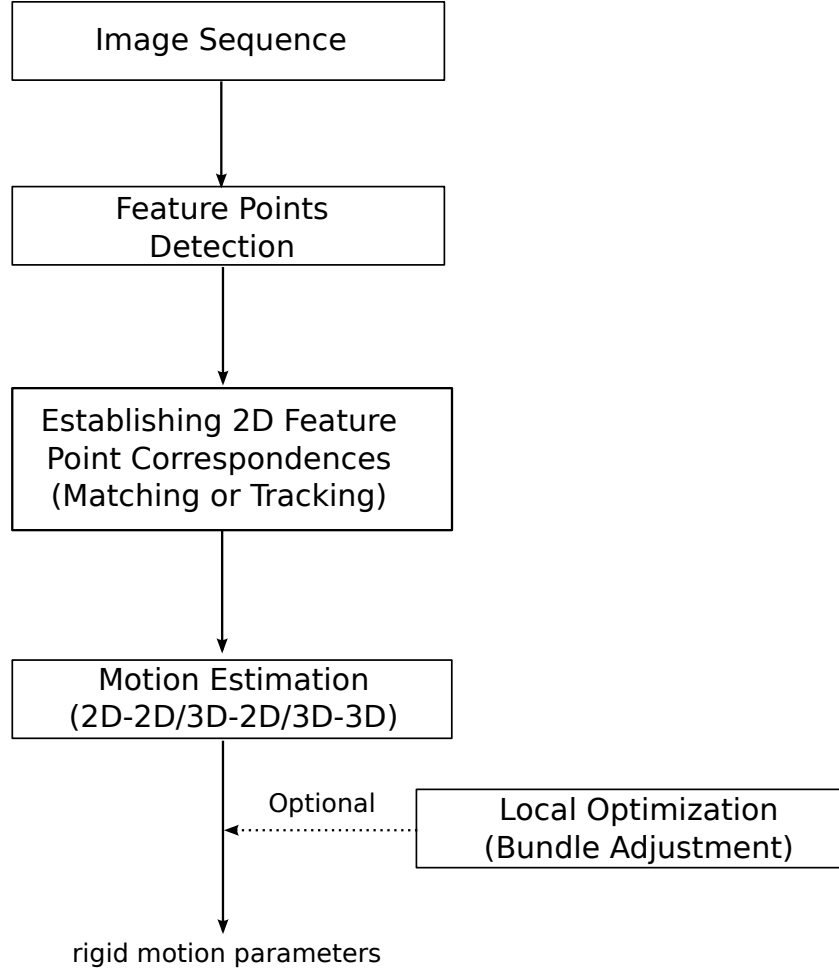


Figure 3.1: Pipeline of sparse feature points based visual odometry

tain more accurate trajectory results, when compared with those in [Moravec 1980]. [Lacroix 1999] detects candidate key points from dense stereo by analyzing the correlation function around its peaks. This kind of key point demonstrates advantages in accurate feature localization. Later works, as [Cheng 2006, Cheng 2007], utilize a similar curvature of the correlation around Harris corner point to define an error covariance matrix. Furthermore, they adopt RANSAC in least squares estimation for rejecting outliers. For each stereo pair, they reconstruct the 3D positions of feature points by triangulation. The motion estimation is treated as a 3D-3D point alignment problem. To build correspondences of feature points through frames, [Nister 2004] uses feature matching to replace previous feature tracking. It could ameliorate feature drift problem when tracking features through multiple frames. The motion information is computed from 3D-2D point constraints. A similar procedure is proposed in [Geiger 2011]. It greatly reduces the feature matching time by extracting binary features. Another different approach proposed in [Comport 2007] estimates the motion from 2D-2D image point matches without their 3D positions.

This solution derives from a "quadrifocal tensor". The authors claim that more accurate motion estimation could be reached by the 2D-2D based motion estimation.

In the following, we will follow the framework of classic stereo visual odometry to estimate vehicle's ego-motion. The contribution of our works is the comparison of various feature detectors and the different approaches of circular feature association.

### 3.2 Circular Feature Points Detection and Association

The objectives of this section are finding local invariant salient points in successive stereo image pairs and establishing their mutual correspondences. As illustrated in Fig. 3.1, these steps provide necessary information for the following motion estimation. In our practice, we detect and associate the feature points in a circular manner as in [Geiger 2011]. For an acquired stereo image pair at time  $t - 1$  and  $t$ , circular point-to-point correspondences are established in all the four images: *current left/right images* and *previous left/right images*. This is depicted in Fig. 3.2. In this step, a feature point  $(u_l^t, v_l^t)$  detected in the current left image at time  $t$  is associated with corresponding locations in the other images as:  $(u_r^t, v_r^t)$ ,  $(u_l^{t-1}, v_l^{t-1})$ ,  $(u_r^{t-1}, v_r^{t-1})$ . Therefore, the four positions of a same point in four images form a circle:  $(u_l^t, v_l^t) \leftrightarrow (u_r^t, v_r^t) \leftrightarrow (u_l^{t-1}, v_l^{t-1}) \leftrightarrow (u_r^{t-1}, v_r^{t-1})$

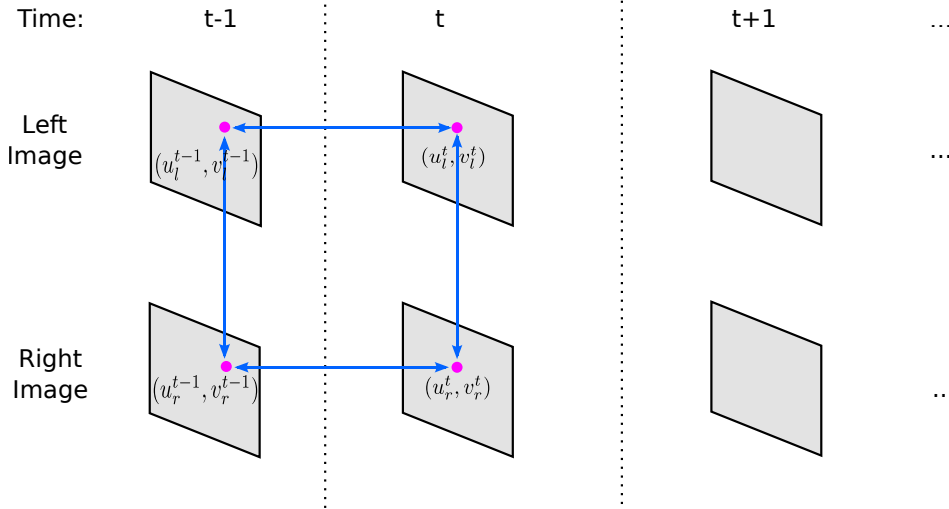


Figure 3.2: Circularly association of detected feature points in consecutive image pairs

In Sec. 2.4, we have reviewed several commonly used image local detectors, descriptors, as well as tracking and matching approaches. To seek a circular point-to-point correspondences between successive images, two distinct approaches are possible:

- The first approach is to detect feature points in one image, then, track the detected points in other images.

- The second approach is to detect feature points in all the four images, then, match these points by their feature descriptors.

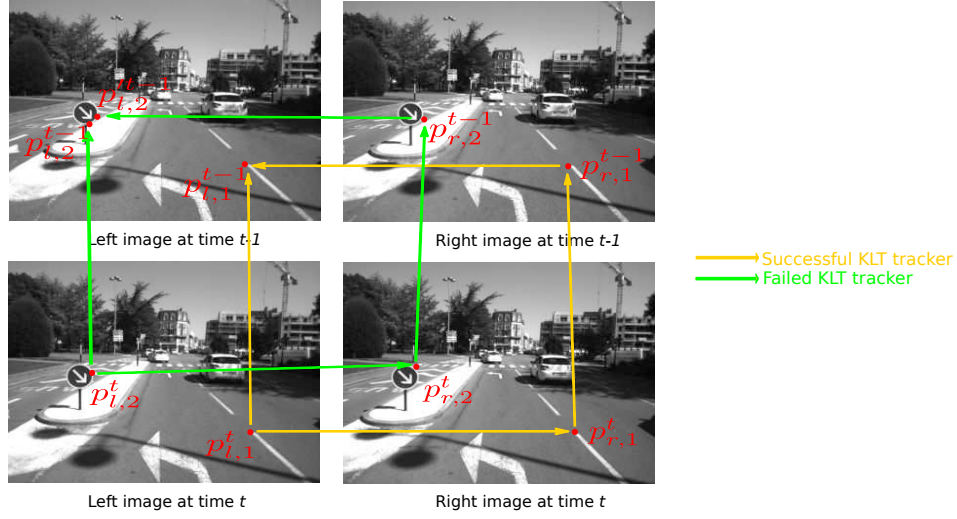
### 3.2.1 Circular Feature Point Association by KLT Tracker

Kanade-Lucas-Tomasi (KLT) tracker (described in Sec. 2.4.3.1) is the most popular point tracking method in the computer vision field. KLT tracker is able to accurately track a point, especially a corner point, by directly using spatial intensity information. In circular feature point association, feature points (e.g. Harris corners) are firstly detected in the the current left image. Then, they are tracked in two different ways to formulate a circle. Let  $p_l^t$  be a feature point in the current left image. The first way tracks  $p_l^t$  in the current right image to  $p_r^t$ , then in previous right image to  $p_r^{t-1}$  and finally in the previous left image to  $p_l^{t-1}$ ; the second way directly tracks  $p_l^t$  in the previous left image to  $p_l'^{t-1}$  from the current left image. If the displacement between  $p_l^{t-1}$  and  $p_l'^{t-1}$  is within a tolerance range, the circular tracking is succeeded. Otherwise, the KLT tracker failed to track this point within a circle. An example of KLT tracker based circular point association is demonstrated in Fig. 3.3 (a). The yellow circle shows a successfully circularly tracked feature point  $p_{l,1}^t$  from the current left image to the other three images. The green circle shows an unsuccessfully tracked circle due to unacceptable displacement.

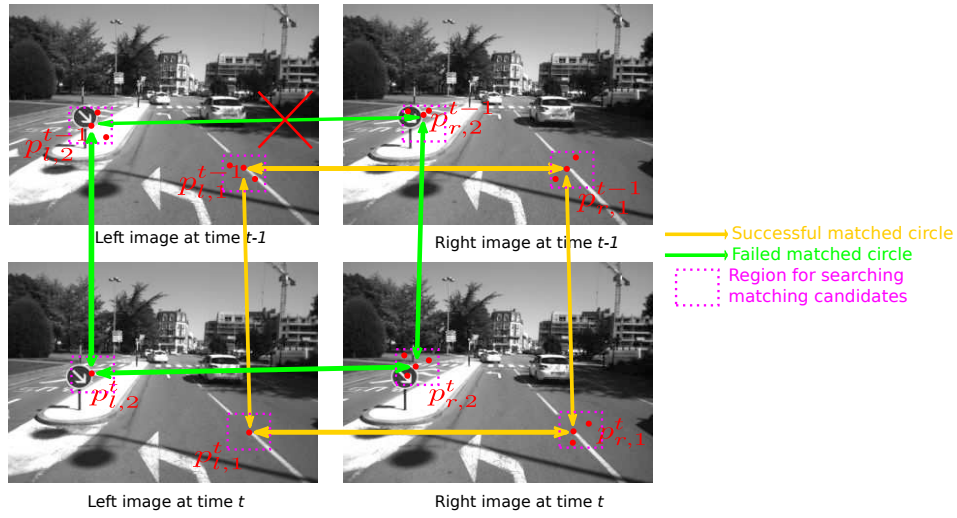
### 3.2.2 Circular Feature Point Association by Matching

As for feature matching based circular feature point association, feature points and their descriptors are computed separately in all the four images. Given a feature point  $p_l^t$  in the current left image, we search its correspondences in the other three images by comparing their descriptors. To avoid exhaustive search in the whole image, we define a region of interest (ROI) to speed-up the matching process. Since the points are matched between consecutive frames, the change between images is usually not too large. Hence, the ROI is set as a fixed rectangular area around  $p_l^t$  in all the other three images. Similarly, the matching process is refined by comparing the displacement of matched feature points. The matched points that have large distance between each other are discarded to assure the quality of matching. A typical example is depicted in Fig. 3.3 (b). The yellow circle shows a successfully circularly matched feature point  $p_{l,1}^t$  from the current left image to the other three images. The green circle shows an unsuccessfully circularly matched feature point  $p_{l,2}^t$  due to deviation. The purple dotted rectangle is the ROI to simplify searching process.

In addition, two real examples of circular feature points association by KLT tracker and feature matching are shown in Fig. 3.4 (a) and (b), respectively.



(a) KLT tracker based approach



(b) Feature matching based approach

Figure 3.3: Circular feature point association by KLT tracker (a) and feature matching (b)



(a) KLT tracker based approach (Harris corner)



(b) Feature matching based approach (SIFT detector + SIFT descriptor)

Figure 3.4: Real examples of circular feature point association

### 3.3 Ego-motion Computation

After the previous steps, sparse feature points are extracted and their correspondences are established through four images in consecutive acquiring times. Hence, for a being processed stereo image pair at time  $t$ , we get a set of associated feature points as:  $\mathcal{S}^t = \{\mathbf{s}_1^t, \mathbf{s}_2^t, \dots, \mathbf{s}_i^t, \dots, \mathbf{s}_N^t\}$ , where the subset  $\mathbf{s}_i^t | i = 1, \dots, N$  represents a successfully associated image feature point and its correspondences:  $\mathbf{s}_i^t = \{p_{l,i}^t, p_{r,i}^t, p_{l,i}^{t-1}, p_{r,i}^{t-1}\}$ , and  $N$  denotes the total number of such subsets. Referring to Sec. 3.1, given a set  $\mathcal{S}^t$  of circularly associated feature points, there are three types of geometrical constraints (3D-3D/3D-2D/2D-2D) [Scaramuzza 2011] to compute ego-motion information. In this section, we only describe how to use 3D-2D constraint to estimate ego-motion.

#### 3.3.1 3D-2D Constraint based Ego-motion Estimation

Suppose a binocular stereo rig is already calibrated and the stereo images are all well rectified, then, real 3D positions of the sparse feature points can be easily reconstructed by triangulation (Eq. 2.23). Taking the left image frame as the reference coordinate system, a subset of associated feature points  $\mathbf{s}^t = \{p_l^t, p_r^t, p_l^{t-1}, p_r^{t-1}\}$  can reconstruct two 3D positions of a same feature point in consecutive times:

$$P^t = \begin{bmatrix} X^t \\ Y^t \\ Z^t \end{bmatrix} = \begin{bmatrix} (u_l^t - c_u) \cdot b / \Delta^t \\ (v_l^t - c_v) \cdot b / \Delta^t \\ f \cdot b / \Delta^t \end{bmatrix}, \quad P^{t-1} = \begin{bmatrix} X^{t-1} \\ Y^{t-1} \\ Z^{t-1} \end{bmatrix} = \begin{bmatrix} (u_l^{t-1} - c_u) \cdot b / \Delta^{t-1} \\ (v_l^{t-1} - c_v) \cdot b / \Delta^{t-1} \\ f \cdot b / \Delta^{t-1} \end{bmatrix} \quad (3.1)$$

where  $\Delta^t = |u_l^t - u_r^t|$  and  $\Delta^{t-1} = |u_l^{t-1} - u_r^{t-1}|$  are the disparities at time  $t$  and  $t-1$  respectively.  $b, f$  and  $(c_u, c_v)$  correspond to the baseline length, focal length, the position of principal point, respectively. Suppose the displacement between  $P^{t-1}$  and  $P^t$  is entirely caused by the self-motion of the stereo rig, which amounts to assume the point is static. Then,  $P^{t-1}$  and  $P^t$  are related by a rigid 3D transformation:

$$\tilde{P}^t = \begin{bmatrix} \mathbf{R}_{t-1}^t & \mathbf{T}_{t-1}^t \\ \mathbf{0} & 1 \end{bmatrix} \tilde{P}^{t-1} = \mathbf{M}_{t-1}^t \cdot \tilde{P}^{t-1} \quad (3.2)$$

where  $\tilde{\cdot}$  denotes homogeneous coordinates,  $\mathbf{R}_{t-1}^t$  and  $\mathbf{T}_{t-1}^t$  represent the  $3 \times 3$  rotation matrix and  $3 \times 1$  translation vector from time  $t-1$  to  $t$ , respectively.  $\mathbf{M}$  is a  $4 \times 4$  augmented transformation matrix. Directly computing motion parameters from Eq. 3.2 is based on 3D-3D constraint. However, issuing from the inherent inaccuracy of 3D triangulation, this approach usually results large deviation. To make a step forward, in ideal situation, considering corresponding image position  $p^t$ , we can have:

$$\tilde{p}^t = \mathbf{K}[\mathbf{I}|\mathbf{0}]\mathbf{M}_{t-1}^t \cdot \tilde{P}^{t-1} \quad (3.3)$$

where  $\mathbf{K}$  is the  $3 \times 3$  intrinsic matrix in Eq. 2.12. Since the errors are inevitable in real problems, given a set  $\mathcal{S}^t$  containing  $N$  circularly associated feature points,

ego-motion can be solved by minimizing:

$$\arg \min_{R,T} \sum_{i=1}^N ||\tilde{p}_i^t - \mathbf{K}[\mathbf{I}|\mathbf{0}]\mathbf{M}_{t-1}^t \tilde{P}^{t-1}||^2 \quad (3.4)$$

The minimization of Eq. 3.4 is a typical non-linear least square problem. We use Gauss-Newton method (Sec. 2.3.2.1) to iteratively minimize Eq. 3.4 with respect to the transformation parameters  $(\mathbf{R}, \mathbf{T})$ .

**Ego-motion estimation by Gauss-Newton method:** In fact, the desired rotation matrix can be represented by three parameters: the *yaw* angle  $\alpha$ , *pitch* angle  $\beta$  and *roll* angle  $\gamma$ :

$$\mathbf{R}(\alpha, \beta, \gamma) = \begin{bmatrix} \cos \beta \cos \gamma & -\cos \beta \sin \gamma & \sin \beta \\ \sin \alpha \sin \beta \cos \gamma + \cos \alpha \sin \gamma & -\sin \alpha \sin \beta \sin \gamma + \cos \alpha \cos \gamma & -\sin \alpha \cos \beta \\ -\cos \alpha \sin \beta \cos \gamma + \sin \alpha \sin \gamma & \cos \alpha \sin \beta \sin \gamma + \sin \alpha \cos \gamma & \cos \alpha \cos \beta \end{bmatrix} \quad (3.5)$$

and the translation vector is:

$$\mathbf{T} = [t_x, t_y, t_z]^T \quad (3.6)$$

Accordingly, Eq. 3.3 can be expanded by substituting Eq. 3.5 and Eq. 3.6 into Eq. 3.3 as:

$$\begin{bmatrix} u^t \\ v^t \end{bmatrix} = \begin{bmatrix} f \cdot \frac{X^t}{Z^t} \\ f \cdot \frac{Y^t}{Z^t} \end{bmatrix} \quad (3.7)$$

where

$$\begin{aligned} X^t &= \cos \beta \cos \gamma \cdot X^{t-1} - \cos \beta \sin \gamma \cdot Y^{t-1} + \sin \beta Z^{t-1} + t_x \\ Y^t &= (\sin \alpha \sin \beta \cos \gamma + \cos \alpha \sin \gamma) \cdot X^{t-1} - (\sin \alpha \sin \beta \sin \gamma + \cos \alpha \cos \gamma) \cdot Y^{t-1} \\ &\quad - \sin \alpha \cos \beta \cdot Z^{t-1} + t_y \\ Z^t &= (-\cos \alpha \sin \beta \cos \gamma + \sin \alpha \sin \gamma) \cdot X^{t-1} + (\cos \alpha \sin \beta \sin \gamma + \sin \alpha \cos \gamma) \cdot Y^{t-1} \\ &\quad + \cos \alpha \cos \beta \cdot Z^{t-1} + t_z \end{aligned} \quad (3.8)$$

Therefore, for one data point, the corresponding *Jacobian matrix* is:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial u}{\partial \alpha} & \frac{\partial u}{\partial \beta} & \frac{\partial u}{\partial \gamma} & \frac{\partial u}{\partial t_x} & \frac{\partial u}{\partial t_y} & \frac{\partial u}{\partial t_z} \\ \frac{\partial v}{\partial \alpha} & \frac{\partial v}{\partial \beta} & \frac{\partial v}{\partial \gamma} & \frac{\partial v}{\partial t_x} & \frac{\partial v}{\partial t_y} & \frac{\partial v}{\partial t_z} \end{bmatrix} \quad (3.9)$$

The partial derivatives with respect to  $\alpha, \beta, \gamma, t_x, t_y, t_z$  can be easily deducted from Eq. 3.9. Given  $N$  circularly matched feature points, the final Jacobian matrix can be created by packing all corresponding Eq. 3.9 together. Then, Gauss-Newton method is performed: Given an initial guess of the rotation matrix and translation vector, the residual is calculated according to Eq. 3.4. Next, a shift vector to update the parameters is computed by Eq. 2.59. When the parameters are updated, the



residual according to the new parameters is re-computed. Ego-motion parameters are obtained by repeating such steps until the residual reaches a minimum or the maximum number of iterations is reached. In practice, we note that even if we initialize  $\mathbf{R}$  and  $\mathbf{T}$  to  $\mathbf{0}$ , a couple of iterations are sufficient for convergence.

However, in a real dynamic urban environment, the components of circularly associated feature points have three categories: a portion may belong to static surroundings, several points could be from either independent moving objects (moving pedestrians, vehicles, etc.) or inaccurate associations. For the ego-motion estimation, the last two types of associations are outliers to be removed. To be robust against outliers, we wrap previous estimation process into a RANSAC frame (Sec. 2.3.3.1). The RANSAC based stereo visual odometry is shown in Algorithm 3.

---

**Algorithm 3** Stereo Visual Odometry based on RANSAC

---

- 1: **repeat**
  - 2:   Randomly select at least 3 observation subsets from  $\mathcal{S}^t$  (three is the minimum number for solving 6 motion parameters).
  - 3:   Minimize the re-projection error in Eq. 3.4 and get new motion parameters by Gauss-Newton method.
  - 4:   Classify inliers according to the new motion estimation.
  - 5:   Update motion estimation if more inliers are obtained.
  - 6: **until** The reprojection error converge or the number of iteration reaches the maximum
  - 7: Final optimizing in all the inliers to refine the estimated parameters
-

### 3.4 Experiments of Stereo Visual Odometry

The described ego-motion estimation approach is evaluated by our experimental vehicle SetCar introduced in Sec. 1.3. The installed stereo vision system (Bumblebee XB3) observes surroundings by stereo image pairs (with a resolution of  $1280 \times 960$ ) in a frame-rate of  $13 - 15fps$  with a baseline 0.24 of meters. The whole framework is implemented in C++, based on OpenCV library <sup>1</sup>, without any acceleration technique. A laptop with a CPU Intel i7-3770 quad cores 3.40GHZ running in Linux is used to run the software.

To begin with, we firstly evaluate the performances of different approaches for circularly associating feature points. We mainly compare tracking based approaches with matching based approaches, as well as performances of different feature detectors. The best feature point detector and the best approach of circular point association are selected for final real experiments in urban environments.

#### 3.4.1 Comparing Different Feature Association Approaches

In Sec. 2.4, we have reviewed several image feature detectors and descriptors. Here, we evaluate their performances in circular feature association. The involved feature detectors are: Good feature to track (GFTT) [Shi 1994], FAST [Rosten 2005, Rosten 2006], SIFT [Lowe 2004], SURF [Bay 2008], ORB [Rublee 2011], CenSurE (STAR in OpenCV) [Agrawal 2008], BRISK [Leutenegger 2011], and feature descriptors are: SIFT [Lowe 2004], SURF [Bay 2008], ORB [Rublee 2011], BRISK [Leutenegger 2011]. For a fair comparison, all the feature detectors and descriptors use tuned parameters implemented in OpenCV.

**Evaluating feature detectors:** In [Tuytelaars 2008, Mikolajczyk 2005], several general criterions for evaluating feature detectors are given (refer to Sec. 2.4). Different to the application of object recognition, which usually has prominent scale or viewpoint changes, in our application, the detectors are used to establish point-to-point correspondences between consecutive video frames. Since the movement of our platform is stable and the frame rate of our stereo-vision system is around  $15fps$ , the scale and viewpoint changes are not large in our application. Hence, the performances of different feature detectors are evaluated from 3 aspects: *repeatability*, *uniformity* and *speed*:

- *Repeatability*: Given two images of the same scene acquired in different conditions, the percentage of features that could be found in both images is defined as repeatability. Repeatability score is define as:

$$S_r = \frac{f^-}{f^*} \quad (3.10)$$

where  $f^-$  is the number of features found in both images,  $f^*$  is the number of features detected in one of the two images, noted as reference image.

---

<sup>1</sup><http://opencv.org>

- *Uniformity*: To better estimate motion in dynamic scenes, the detected features should be distributed as even as possible. To evaluate uniformity, we divide the  $1280 \times 960$  size image into  $n_u$  disjoint identical grids. If the number of feature points located in a grid is more than a threshold, it would be marked as "filled". The uniformity score is:

$$S_u = n_u^- / n_u \quad (3.11)$$

where  $n_u^-$  is the number of "filled" grids.

- *Speed*: The processing speed is measured in millisecond (ms). Usually, the detectors with high processing speed is more preferable in real-time tasks, such as ADAS in intelligent vehicle applications.

To evaluate repeatability, we convert an image by a homography transformation, in a similar way as in [Tuytelaars 2008]. Then, feature detectors are independently performed in the two images. Next, we wrap the detected feature points in the first image to the second one by the known homography matrix. A feature point is repeated, if the wrapped position in the second image also has a detected feature point. In our experiments, randomly generated homography transformations confined in a moderate range are applied to 100 images captured by our stereoscopic system in the city of Belfort. Repeatability scores are computed as in Eq. 3.10. The performance of all the measured detectors are shown in Fig. 3.5. In this eval-

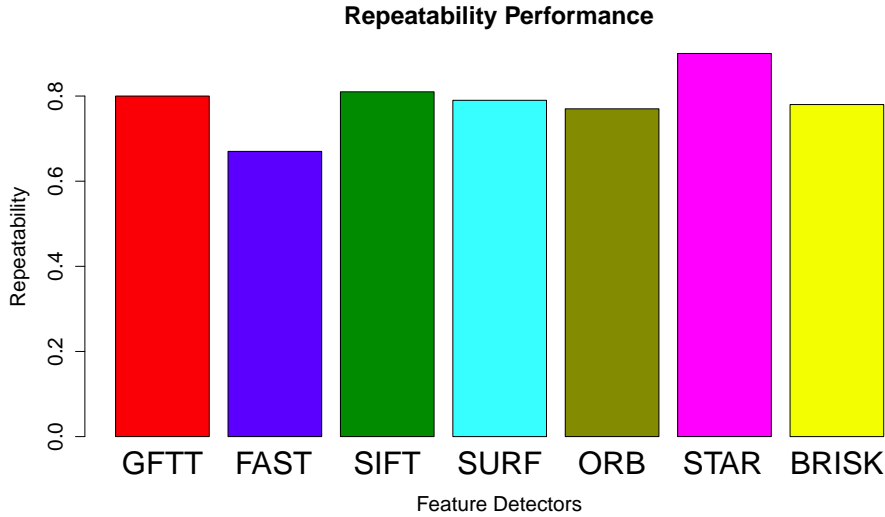
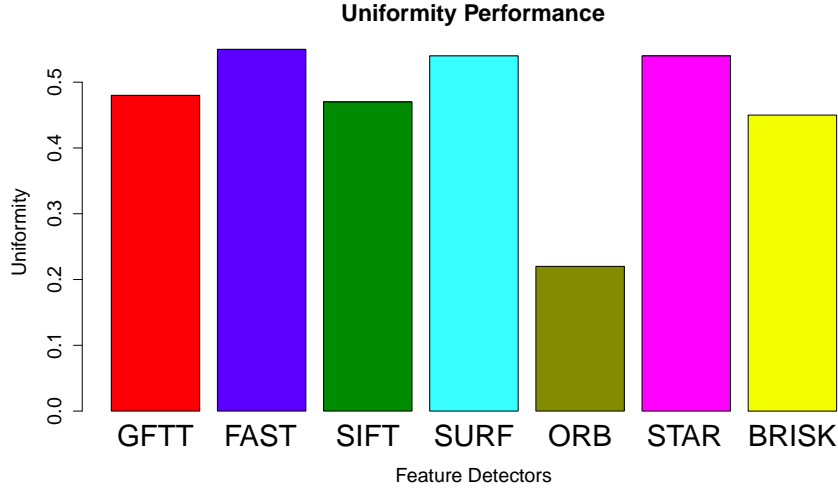


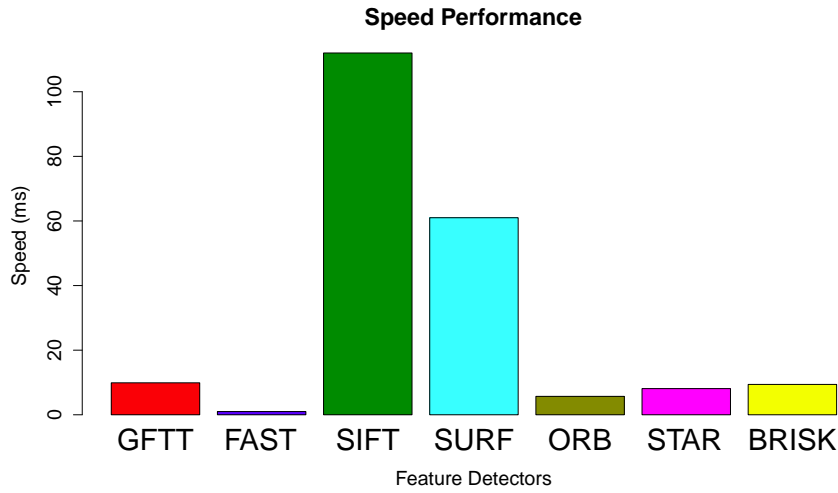
Figure 3.5: Performances of repeatability for all the tested detectors

uation, STAR performs the best, with a score of 0.9, followed by SIFT (0.82) and GFTT (0.8). FAST performs the worst, with a score of 0.64. The results reveal that the DoG (Difference of Gaussian) based methods (STAR, SIFT) attain the best repeatability score.

To evaluate uniformity score, each test image is divided into 192 squares with a size of  $80 \times 80$  pixels and the threshold is set to 2. We tested all the 100 selected images and the evaluation results are shown in Fig. 3.6 (a). It can be seen that FAST performs the best with a uniformity score of 0.67, followed by STAR, SURF, GFTT, SIFT and ORB performs the worst. The main reason that FAST reaches the best score is of its huge number of detected feature points. The results of processing



(a)



(b)

Figure 3.6: Performances of uniformity (a), and speed (b) for all the tested detectors

speed are shown in Fig. 3.6 (b). FAST is no doubt much more faster than the other detectors thanks to its extremely simplified computation. GFTT, ORB and STAR

are almost 10 times slower than FAST. SIFT is the slowest due to the computation of DoG and searching maxima in multi-scale spaces.

**Evaluating circular feature point association:** As we stated in Sec. 3.2.2, there are two approaches, the KLT tracker based and feature matching based, for circularly associating feature points in four images.

For the matching based approach, feature descriptors can be combined with almost all kinds of detectors, such as "GFTT detector + SIFT descriptor". To avoid tedious, reduplicative and similar comparison, we only take original detector-descriptor combinations into consideration, (e.g. "SIFT detector + SIFT descriptor", "SURF detector + SURF descriptor", "ORB detector + ORB descriptor", "BRISK detector + BRISK descriptor" ). For the KLT tracker based approach, we evaluate following detectors: GFTT, FAST and STAR. Also, we propose two criterions to evaluate their performances:

- *Survival rate*: Both KLT tracker and matching based methods would eliminate a part of detected feature points due to inaccurate tracking or matching. When tracking or matching feature points through four images, such phenomenon is amplified. The "survived" associated feature points maybe too few to be utilized. Hence, we use *survival rate*  $S_s$  to describe the performance:

$$S_s = \frac{n_{left}}{n_{total}} \quad (3.12)$$

where  $n_{total}$  is the number of all detected feature points in a reference image and  $n_{left}$  is the number of successfully tracked or matched feature points in the other images.

- *Accuracy*: As we have described in Algorithm 3, a RANSAC scheme is used to exclude outliers caused by independent moving objects and imprecisely associated feature points. Although RANSAC is excellent enough to cope with outliers, we still hope that, when moving in a static environment the portion of outliers is as small as possible. Therefore, we introduce an accuracy factor: the ratio of the inliers to number the whole feature points.

$$S_a = \frac{n_{inlier}}{n_{total}} \quad (3.13)$$

In our evaluation, the images without independent moving objects are used.

Except the proposed two criterions, the performance of speed is also very important.

- *Speed*: For KLT tracker based method, the processing time is counted including all the four tracking times (as illustrated in Fig. 3.3). For matching based approach, the operation time is measured for all the four matching processes, which is also drawn in Fig. 3.3.

In our evaluation, 100 stereo image circular subsets captured in static scenes are used for evaluation. For a fair comparison, the number of detected feature points

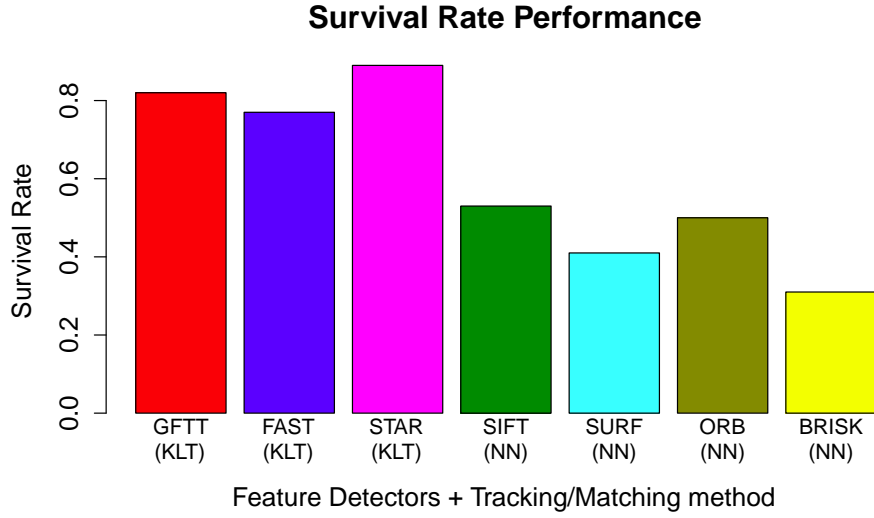


Figure 3.7: Survival rate performances for all the tested methods

in each image is controlled between 400-500. Fig. 3.7 shows the results of survival rate for all the considered KLT tracker based and matching based methods. Among all the methods, KLT based STAR detector achieves the best performance (survival rate: 0.88), followed by GFTT+KLT and FAST+KLT. The BRISK detector and descriptor get the worst results. An interesting finding is that: in survival rate comparison, all the KLT based approaches perform better than feature matching based method. This fact could be explained by the repeatability in Fig. 3.5. For KLT based methods, feature points are detected once in one image and tracked to other images. While for feature matching based methods, features points are independently detected in four images. Therefore, suppose we choose SIFT detector, only 40% percent ( $0.8^4 \sim 0.4$ ) of extracted points re-appear in all the four images. This problem is not serious in KLT based methods, due to KLT tracker's high accuracy and robustness.

The results of accuracy evaluation are shown in Fig. 3.8. All the KLT tracker based methods perform better than feature matching based methods. STAR based KLT tracker achieves the best accuracy score, followed by GFTT and FAST based KLT trackers. Fig. 3.9 reveals the comparison results in terms of speed. Again, the performances of the KLT based methods are beyond feature matching based process. Matching 128-length feature descriptors of SIFT is no doubt the slowest approach. Although SURF reduce the length of descriptor to 64, ORB and BRISK step forward by substituting with a binary descriptor, they still require much more processing time than KLT tracker based method. The main reason is due to the repeated feature detection processes.

**Choose the Best Approach of Circular Feature Point Association:** We have compared different methods to achieve circular feature point association. It is obvi-

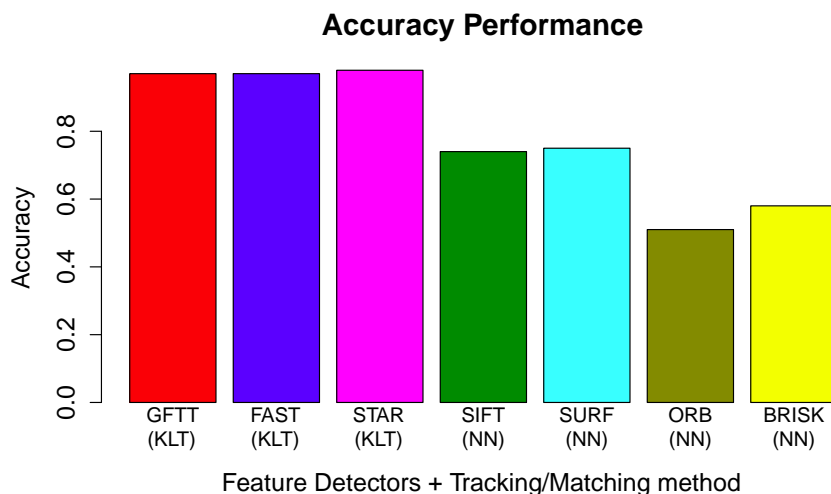


Figure 3.8: Accuracy performances for all the tested methods

ous that KLT tracker based methods are better than feature matching methods. The results are due to the characteristic of unremarkable changes between consecutive images, which makes KLT tracker more suitable than feature descriptor matching – the latter has advantages when viewpoint has prominent changes. Another interesting finding is the poor performances of binary descriptors (ORB, BRISK). The improvement of matching speed for binary descriptors is maybe obtained in sacrifice of accuracy, as shown in Fig. 3.8

Comparing feature detectors, we notice that STAR (CenSurE) detector performs the best in global evaluation. It is quick for computation, stable and accurate for cross-image detection and tracking. Although FAST feature detector is extremely fast for computation, it lacks robustness and accuracy for tracking, compared with STAR or GFTT. GFTT can be regarded as the second best choice, since it is just slightly worse than STAR. Therefore, in our application, we choose STAR feature detector to accurately localize feature points. KLT tracker is used for establishing point-to-point correspondences through consecutive four stereo images.

### 3.4.2 Stereo Visual Odometry in Urban Environment

After picking up the best method for circular feature point association, we apply the described stereo visual odometry method in several real datasets. At first, we demonstrate the experimental results in a well-known open benchmark. Then, more experiments performed by our platform SetCar are conducted in the city of Belfort, France.

**Test in KITTI Vision Benchmark Suite:** KITTI Vision Benchmark Suite <sup>2</sup> [Geiger 2012] is a state-of-art vision dataset provided by Karlsruhe Institute of

<sup>2</sup><http://www.cvlibs.net/datasets/kitti/>

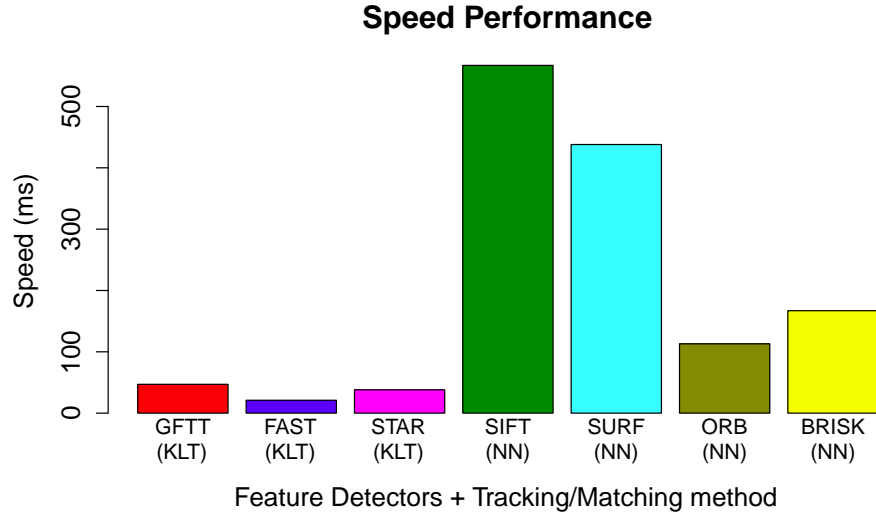


Figure 3.9: Processing speed performances for all the tested methods

Technology (KIT)’s intelligent vehicle platform. The benchmark suite contains large amount of data for stereo matching, optical flow, visual odometry/SLAM, 3D object recognition. In addition, it also provides MATLAB/C++ development kits for easy access.

In our experiments, we utilize the dataset of visual odometry. The dataset consists of 11 rectified stereo video sequences with a resolution of  $1241 \times 376$  pixels acquired with a frame rate  $10fps$ , in Karlsruhe, Germany. The calibration parameters and ground truth provided by a RTK-GPS are also given. The described method is tested with the dataset and several results are shown in Fig. 3.10



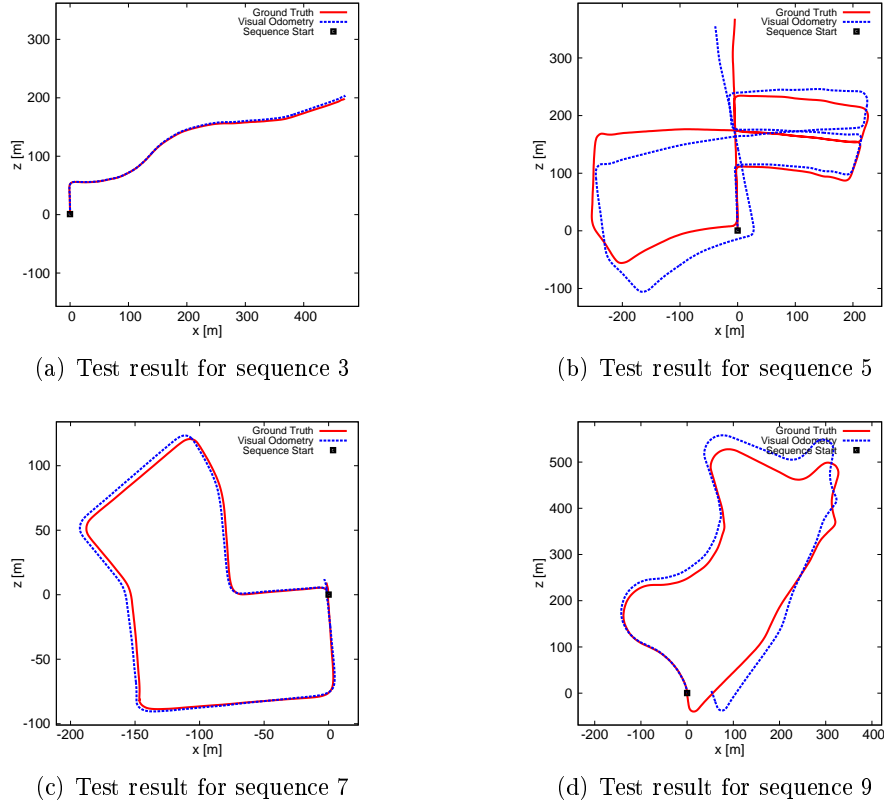


Figure 3.10: Four examples of stereo visual odometry experiments for KITTI dataset

In Fig. 3.10, 4 experimental results with KITTI visual odometry dataset are shown. Red lines represent ground truths of the trajectories acquired by RTK-GPS. The blue dot-lines are the trajectories estimated by stereo visual odometry. It can be found that, the performance of visual odometry deteriorates quickly when the cameras are turning. Also, the errors increase and accumulate along with the driving distance. More quantitative error analyzing results are shown in Fig. 3.11. Translation error analyses for these 4 sequences are given in Fig. 3.11 (a) - (h). Fig. 3.11 (i) - (j) illustrate the translation errors in average with regard to the path distance and driving speed for all the 11 sequences. It can be found from the results that, within the range of 400 meters, the translational deviation is usually around 3 - 4 percentage of the total distance. However, the translation errors increase by the path distance, this is due to the errors in visual odometry coming from errors in feature points' positions, 3D triangulation and calibration parameters. In fact, long-distance and accurate visual odometry is still an open question for computer vision community. Another interesting finding is that the suitable driving speed for applying visual odometry is within 25km/h to 40km/h.

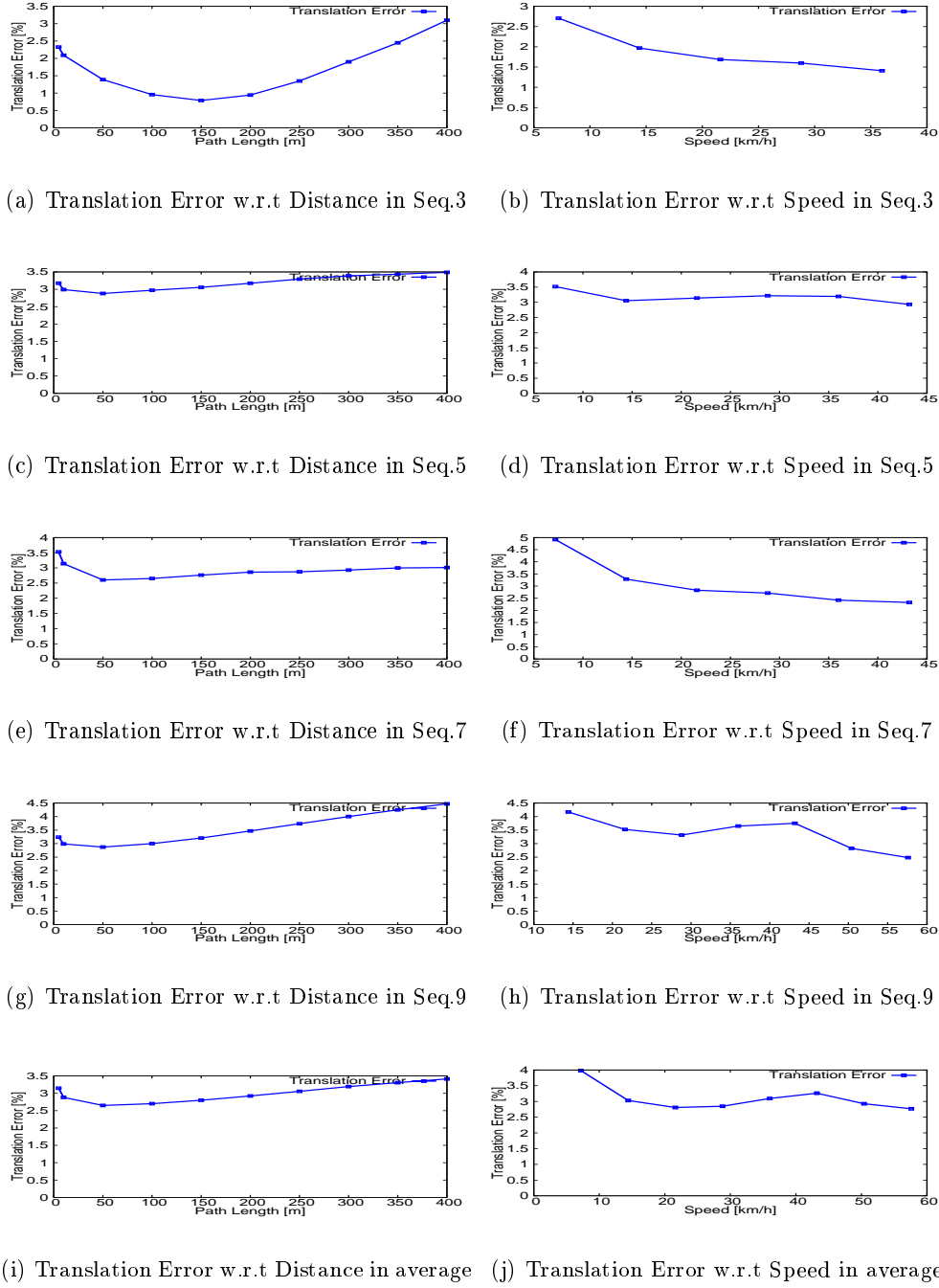


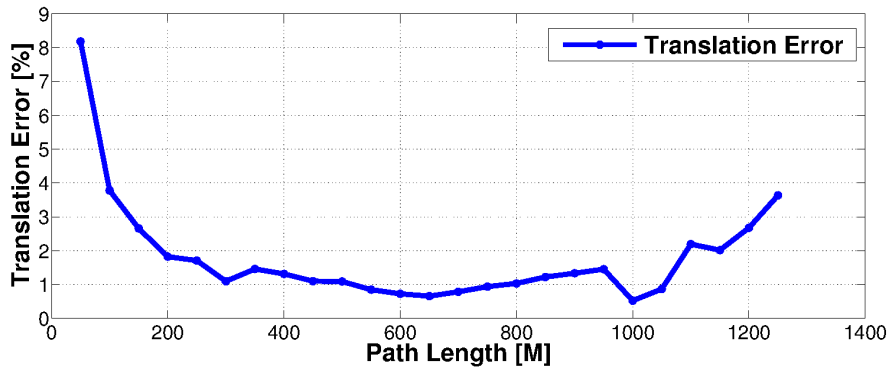
Figure 3.11: Quantitative translation error analyses

**Test in our platform SetCar:** The test sequences provided by KITTI benchmark are confined in a short distance. To evaluate our adopted stereo visual odometry method, we apply it using our platform SetCar, described in Sec. 1.3. The platform was driven in the city of Belfort in much longer distances (more than 1000 meters).

All the test sequences were captured by our Blumbeebie XB3 stereo vision system. The stereo image pairs are well rectified after a stereo calibration process. The positions localized by the equipped RTK-GPS are regarded as ground truth and compared with the results of the described visual odometry method. Four real experimental results are shown in Fig: 3.12, 3.13, 3.14, 3.15.



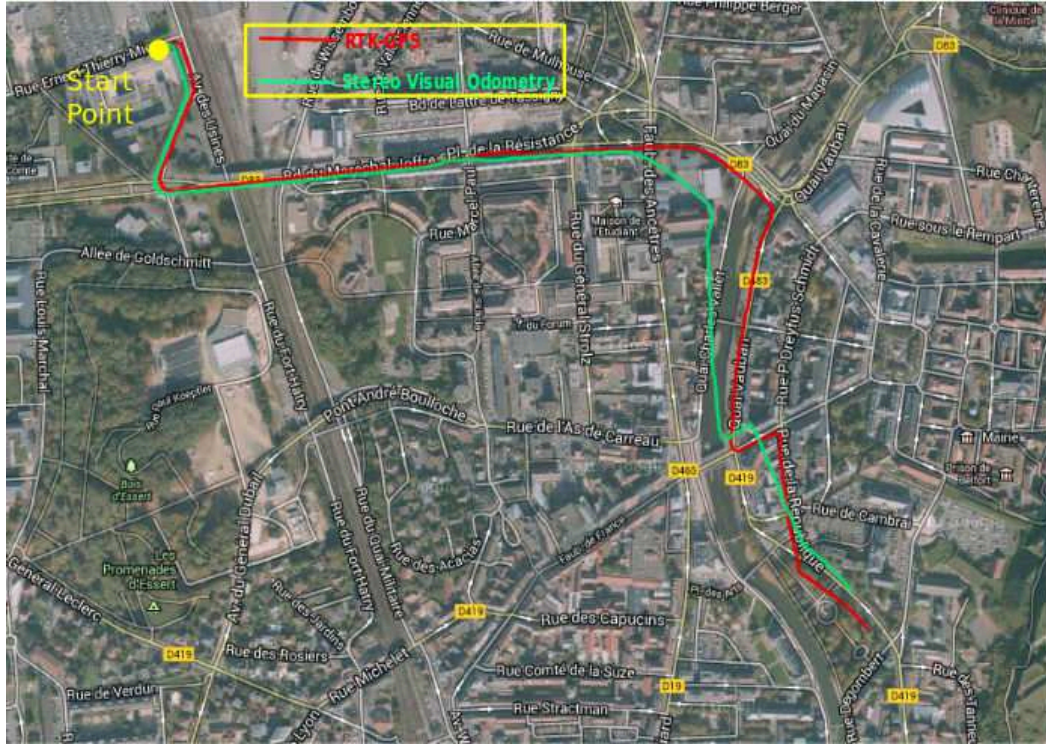
(a) Comparison between RTK-GPS and Stereo Visual Odometry



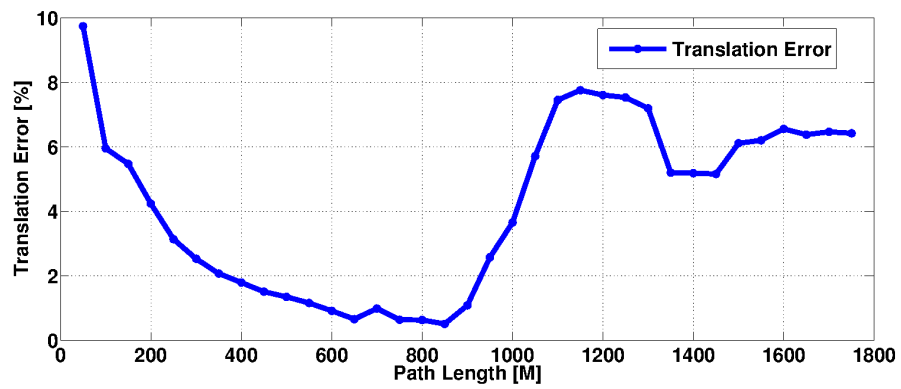
(b) Translation error w.r.t path distance

Figure 3.12: The first experimental results with our platform

From the results of our experiments, it can be seen that the accuracy of the proposed visual odometry method is still unable to compare with RTK-GPS. In long distance situations, translation errors are slowly accumulated. The average translation error rate within average of 1000 meters is below 8%. The



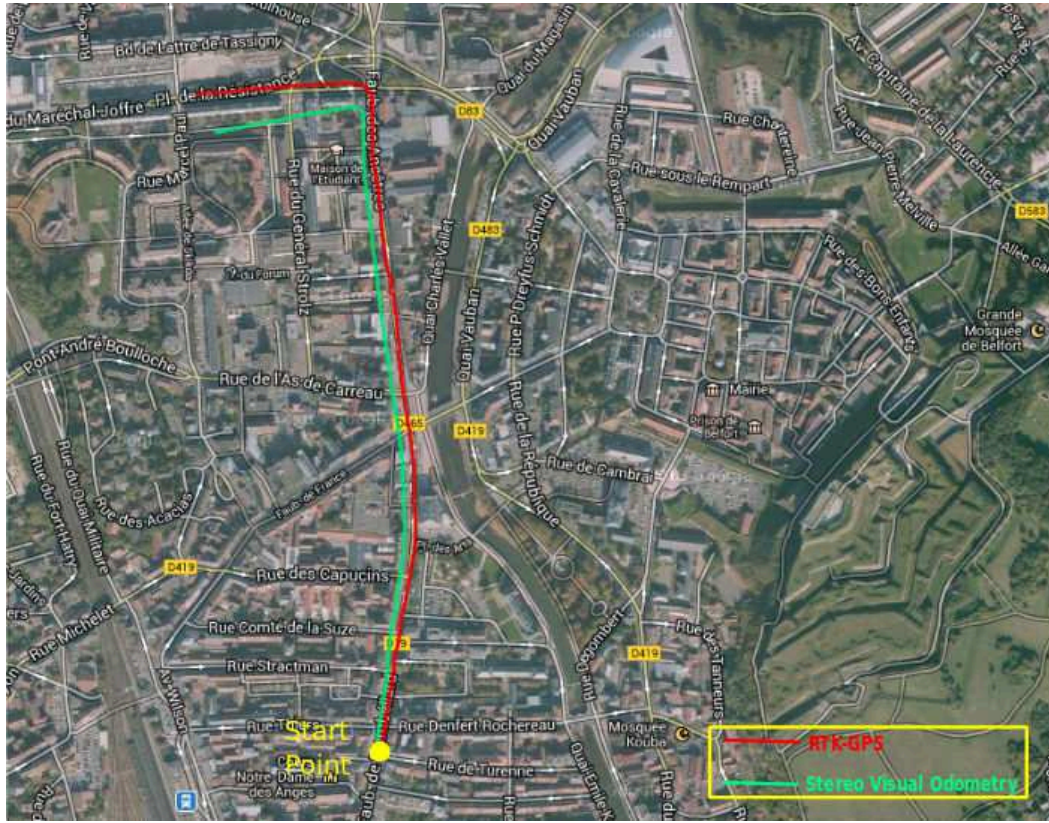
(a) Comparison between RTK-GPS and Stereo Visual Odometry



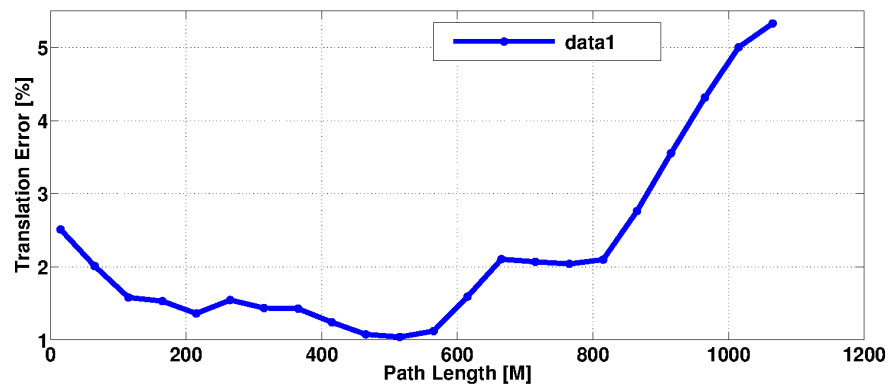
(b) Translation error w.r.t path distance

Figure 3.13: The second experimental results with our platform





(a) Comparison between RTK-GPS and Stereo Visual Odometry

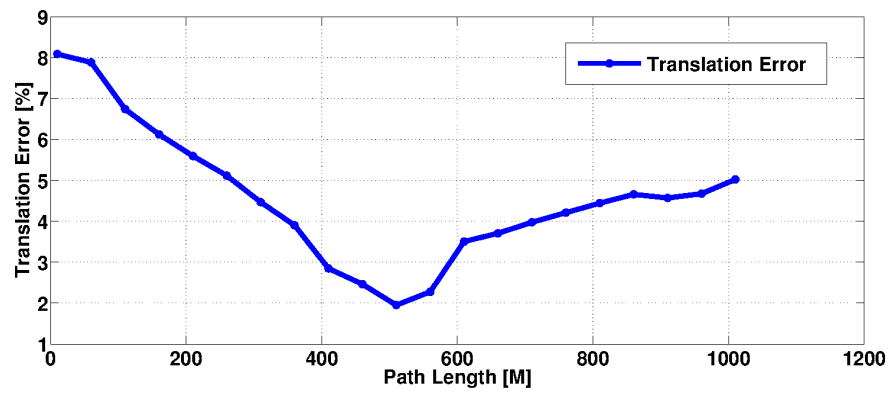


(b) Translation error w.r.t path distance

Figure 3.14: The third experimental results with our platform



(a) Comparison between RTK-GPS and Stereo Visual Odometry



(b) Translation error w.r.t path distance

Figure 3.15: The fourth experimental results in our platform

### 3.5 Conclusion and Future Works

In this chapter, we presented a stereo vision based method to estimate ego-motion for a moving intelligent vehicle. The ego-motion information of the vehicle itself (visual odometry) is firstly investigated by utilizing quad-associated feature points in consecutive stereo images. To improve the accuracy of visual odometry, various image features and methods of feature association are tested and compared. The experimental comparisons showed that the combination of CenSurE and KLT tracker achieves the best performance. Real experiments of visual odometry are tested with an open benchmark, as well as our platform. From the experiments, we can see that, compared with RTK-GPS, visual odometry still lacks of accuracy. The future works would focus on improving accuracy of visual odometry.

To improve the accuracy of feature based visual odometry, following approaches could be attempted:

- Improve the accuracy of key points locations by computing sub-pixel positions. In our experiments, the positions of feature points are in discretized coordinates. However, real world is continues. The usage of sub-pixel position of feature points maybe increase the accuracy of visual odometry.
- Model the error of key points locations. In our experiments, the weights of all feature points are the same. Modeling the error of different key points according to distance or other criterions maybe improve the accuracy of stereo visual odometry.

# Independent Moving Object Detection, Segmentation and Recognition

---

## Contents

<b>4.1 Independent Moving Object Detection and Segmentation .</b>	<b>71</b>
4.1.1 Introduction . . . . .	71
4.1.2 UV-Disparity Based Independent Moving Objectxs Detection and Segmentation . . . . .	73
4.1.3 Experimental results of U-disparity image based Independent Moving Object Detection . . . . .	77
<b>4.2 Moving Object Recognition using Spatial Information . . .</b>	<b>80</b>
4.2.1 Introduction . . . . .	80
4.2.2 Spatial Feature Extraction and Classification . . . . .	81
4.2.3 Experimental Results . . . . .	84
<b>4.3 Conclusion and Future Works . . . . .</b>	<b>89</b>

---

In the previous chapter, we presented a method of estimating ego-motion information for a moving intelligent vehicle. However, real urban environments are complex in full of independent moving objects, such as moving pedestrians, vehicles, cyclists, etc. Therefore, the perception system of an intelligent vehicle requires an ability to analyze the moving objects when it is driven. In this chapter, we firstly present a method to detect and segment moving objects from moving stereo vision system and then, we introduce a recognition method to recognize the segmented objects based on spatial information.

## 4.1 Independent Moving Object Detection and Segmentation

### 4.1.1 Introduction

Vision based moving object detection is an old but still dynamic computer vision area. It provides a classification of the pixels in video sequence into either foreground (moving objects) or background. When the camera, or vision system, is static, one of



the most common approach to achieve such moving object detection is background subtraction [Wren 1997, Toyama 1999]: each video frame is compared with a reference or background model, then pixels that significantly differ from the background model are considered as parts of moving objects. The background model could be from the simplest difference of temporally adjacent frames to complex mixture Gaussians [Stauffer 2000].

However, the problem becomes rather tricky when the vision system is moving, which is usually the case in the field of robotics, intelligent vehicles. Similar background model is very difficult to acquire under such quickly changed environments. To cope with this problem, proposed approaches in the literature could be roughly divided into two categories.

1. The first category uses global motion compensation to generate a background model as utilized in motion detection in static cameras cases [Kaucic 2005]. This method suffers from severe limitations in the assumption of homography transform between consecutive images. Whereas, in real moving situation, the images are changed by perspective transformation. Although several following improvements have been introduced in [Sawhney 2000, Kang 2005]. It is still lack of accuracy and is unable to deal with complex real environment.
2. The second category is generally based on analyzing the displacement of pixels in image plane (optical flow). For example, [Thompson 1990] incorporated dense optical flow with information of camera motion or scene structure to achieve moving object detection when the camera is also moving. [Rabe 2007a] proposed a moving object detection method based on analysis of individually tracked image points (optical flow), which provides a motion metric to indicate their motion status. In [Dey 2012], a multiframe monocular epipolar constraint of camera motion was derived for monocular moving camera. This constraint is combined with optical flow, which captures all motion in the video across multiple frames. As an extension of optical flow from 2D image plane to 3D world, scene flow is able to detect moving objects in 3D space, as introduced in [Lenz 2011, Wedel 2009]. This kind of methods usually involves joint estimation of ego-motion as well as objects movement. Benefit of the second category is that there is no assumption for specific environment.

In this section, we present a sparse optical flow based moving object detection method. This method firstly computes sparse optical flow for detected feature points. Then, motion analysis of the sparse points is achieved by the RANSAC based visual odometry method described in Sec. 3.3: the feature points used to calculate ego-motion are assumed to be static points, while the remaining points are either from independent moving objects or noises. Based on the "suspected" feature points from moving objects, we use U-disparity image to detect real moving objects. At last, based on the detecting results in U-disparity image, the moving objects are segmented in dense disparity map.

### 4.1.2 UV-Disparity Based Independent Moving Objects Detection and Segmentation

#### 4.1.2.1 Disparity map and UV-disparity map

For a stereo image pair  $(I_l, I_r)$ , its corresponding disparity image  $I_\Delta$  refers to the apparent pixel difference or motion between  $I_l$  and  $I_r$ . It can be simply converted to depth map by  $\text{Disparity} = 1/\text{Depth}$ . Therefore, disparity image is usually calculated as a foundation of 3D reconstruction. Numerous algorithms have been proposed to fast and accurately compute the disparity image, for example: Block Matching [Konolige 1998], Semi-Global Block Matching (SGBM) [Hirschmüller 2008], Graph Cut [Kolmogorov 2002], Belief Propagation [Sun 2002]. After a balance between computational speed and quality, we choose SGBM to compute dense disparity image for our application. An example of SGBM results is shown in Fig. 4.1 (b). The figure is pseudo-colored: the color ranges from red to blue, which denotes that the disparity is from small to big. As mentioned before, our stereo image pairs are well rectified. Hence, given the disparity image, pixel-wise 3D reconstruction can be achieved by triangulation, according to Eq. 2.23.

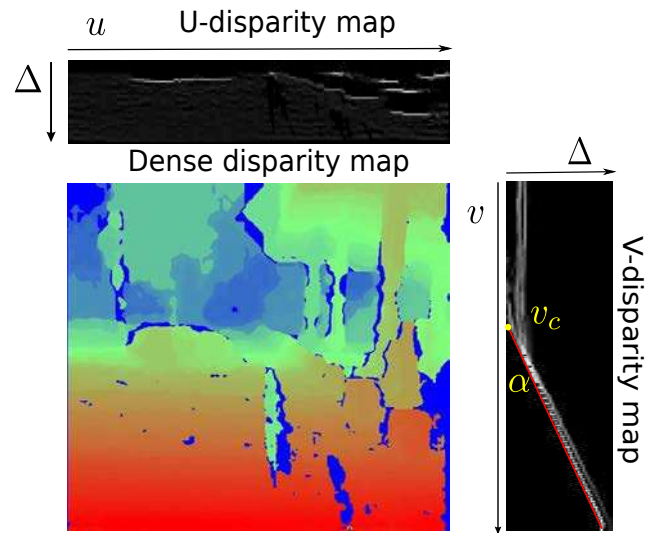
Apart from dense disparity map, U-disparity map  $I_u(u, \Delta)$  and V-disparity map  $I_v(\Delta, v)$  [Labayarde 2002] are also calculated for further scene understanding. In the field of intelligent vehicle, U-V disparity images are practical tools for scene understanding (obstacle detection [Wang 2006], ground plane detection [Soquet 2007], etc.). U-V disparity images are accumulative projections of dense disparity image  $I_\Delta$  to rows/columns. For example, U-disparity image is built by accumulating the pixels with the same disparity in  $I_\Delta$  in each column ( $u$ -axis). Suppose a point  $p_u = (u_i, \Delta_i)$  in  $I_u$  with intensity  $d_u$ , it means that in the  $i$ th column of disparity image  $I_\Delta$ , there are  $d_u$  pixels with the same disparity  $\Delta_i$ . Similarly, V-disparity image is obtained symmetrically. A point  $p_v = (\Delta_j, v_j)$  in  $I_v$  with intensity  $d_v$  means there are  $d_v$  pixels with the same disparity  $\Delta_j$  in the  $j$ th row of  $I_\Delta$ . Examples are shown in Fig. 4.1 (b). Actually, the U-disparity image could be viewed as a bird's view disparity image of the scene. As in the Fig. 4.1 (b), all the obstacles standing in the ground plane are mapped to a bright line in the U-disparity image. In the V-disparity image, ground plane is mapped to a quasi-line (marked as a red line). All the obstacles are projected as bright lines "grow up" from the red line. These attributes of U-V disparity images make them as convenient tools to estimate obstacles and ground plane. In our works, the U-disparity image is used for moving objects detection and segmentation, while the V-disparity image is used for estimating the ground pitch angle introduced in next chapter.

#### 4.1.2.2 Independent Moving Object Segmentation in U-disparity Map

In most cases of driving in urban environments, the vehicle moves on a flat ground plane with a certain pitch angle with respect to the vision system. Suppose in the image, there is a plane perpendicular to the ground plane. Since the ground surface is almost in the same distance to the vision system, the disparities of the



(a) An image of road scene



(b) Dense disparity map and U-V disparity maps

Figure 4.1: Dense disparity image and its UV-disparity images

pixels belonging to the ground plane are almost the same. Hence, the pixels of the ground plane in the same column are transformed to the same point in U-disparity image. This effect illustrates the reason why an object perpendicular to the ground is imaged as a bright line in U-disparity image. In Fig. 4.1, the vehicle, stone, pedestrian and pillar are all mapped as white lines in the U-disparity image. Note that the white lines in the U-disparity image are of the same column positions with the corresponding obstacles in the disparity map.

Based on the characteristic of U-disparity image, detecting moving objects is able to be achieved by detecting the white lines in U-disparity image. Recall to Chapter 3, the extracted feature points are used to compute ego-motion. Within all the associated feature points, inliers that are classified by RANSAC scheme come from the static scene, outliers correspond to independent moving objects or wrongly associated feature points. When projecting the outliers into the U-disparity image, they are always located in the bright lines. This property of U-disparity image implies that a flood-fill segmentation method [Gonzalez 1992] is feasible to detect and segment moving objects. An example is drawn in Fig. 4.2. Fig. 4.2 (a) shows the quad-associated feature points, the red points are inliers used for estimating ego-motion, the blue points are outliers coming from independent moving objects or wrongly matched feature points. Fig. 4.2 (b) shows the projection of the inliers and outliers into the U-disparity map.

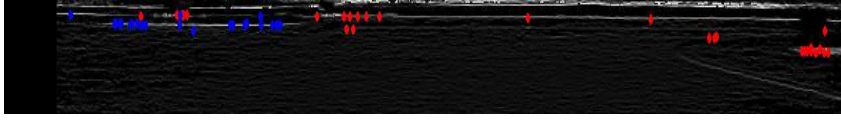
Before performing flood-fill segmentation in U-disparity map, a preprocessing should be conducted to improve the performance. Restricted by the resolution of image, the moving objects too far away are usually almost the same between two consecutive frames. Moreover, the detected feature points in such area are usually not stable for analyzing. Therefore, we define an effective range for detecting moving objects. For the used Bumblebee XB3 stereo vision system with a baseline length of  $0.24m$  and a focal length 1007 pixels, according to Eq. 2.23, the maximum detected distance is  $1007 \times 0.24/1 = 241.68m$ . Whereas, the distance is too far away to effectively distinguish changes. In our application, we set the maximum detected distance as 30 meters. Hence, the pixels with disparities larger than  $1007 \times 0.24/30 \approx 8$  pixels are used for further analyzing.

In addition to the effective detected range, another issue should be correctly addressed. Usually, an object close to the stereo-vision system would be always captured in more pixels than an object far from the system. When transformed in U-disparity map, objects with large disparities (in small distances) are always "brighter" than objects with small disparities (in large distance), as in Fig. 4.1 (b), where the white line representing the vehicle in the left part of the U-disparity image is "darker" than the white line representing the stone block in the bottom right point, due to the distance. This phenomenon would cause difficulties for further segmentation. To make the gray value distribution in U-disparity map more even, we employ an intensity adjustment as a preprocessing step. A modified sigmoid logistic function  $\mathcal{S}(\cdot)$  is used to adjust the intensity of U-disparity map:

$$I'_u = I_u \cdot \mathcal{S}(\Delta) = I_u \cdot \frac{r}{1 + e^{\Delta \cdot c}} \quad (4.1)$$



(a) The inliers (red) and outliers (blue) classified by ego-motion estimation



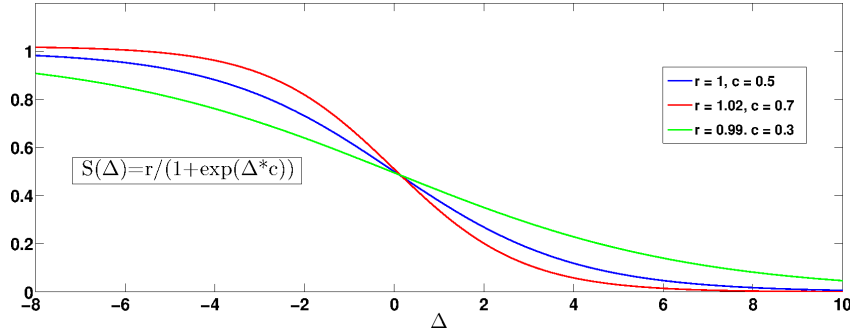
(b) The projection of outliers and inliers in U-disparity image

Figure 4.2: Properties of U-disparity for obstacle detection

where  $I_u$  and  $I'_u$  are intensities of a pixel in U-disparity map before and after adjustment, respectively.  $r$  and  $c$  represent control coefficients.  $\Delta$  is the row in U-disparity map. Sigmoid function is able to smoothly restrain the intensity of areas near the stereo vision system and amplify the intensity of the areas far away. Note that  $\Delta$  is bigger when an object is closer. An illustrative example is given in Fig. 4.3 (a), where three sigmoid-like functions with different parameters are given. With tuned parameters, an example of U-disparity map after intensity adjustment is shown in Fig. 4.3 (b), (c). We can see that the intensities of all potential objects are adjusted similar to each other, regardless of the distance.

Based on the corresponding feature points and adjusted U-disparity map, we segment the independent moving objects in a stereo image pair acquired at time  $t$  as follows:

1. Project all the outliers into the adjusted U-disparity map according to their disparities.
2. Take the new locations of outliers in the adjusted U-disparity map as seeds. Then, a flood fill segmentation algorithm is performed to segment image patches with similar intensities to the seeds.



(a) Sigmoid functions for intensity adjustment



(b) U-disparity image before adjustment



(c) U-disparity image after adjustment

Figure 4.3: Intensity adjustment of U-disparity image

3. After obtaining all the candidate segmentation patches, a merging process is employed to merge all the segments which are mutually overlapped.
4. Since the outliers comprise inaccurate tracked feature points appearing in static obstacles or noises, incorrect segments would lie in candidate segments. To overcome this problem, a double-phase post-processing refinement is applied. In the first phase, each candidate segment, if it contains an inlier projection in the U-disparity map, it is rejected. In the second phase, the surviving segments are compared to stored candidate segments in previous time  $t - 1$ . If a segment has an overlapped region with a previous segment, it passes the refinement and is confirmed as independent moving object in the U-disparity map. The remaining candidate segments are stored for usage in the next frame.
5. At last, confirmed segments in the U-disparity map are back projected to the dense disparity map to get independent moving objects in the dense disparity map.

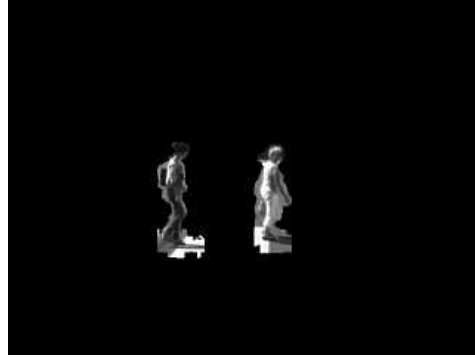
#### 4.1.3 Experimental results of U-disparity image based Independent Moving Object Detection

The proposed method is evaluated by our experimental vehicle SetCar introduced in Sec. 1.3. All the stereo sequences were captured by our Blumblebee XB3 stereo

vision system and well calibrated and rectified. From tens of thousands acquired images, we select several typical test sequences containing independent moving objects for evaluating our method. The programme is written in C++ based on the OpenCV library. The parameters of adjusting U-disparity image are set as  $r = 8$  and  $c = 0.02$ . Five examples are shown in Fig. 4.4 and Fig. 4.5. Note that the segmentation is performed in disparity map, it is hard to be identical to object's real border in original image.



(a) An image of road scene



(b) Moving object segmentation result



(c) An image of road scene



(d) Moving object segmentation result

Figure 4.4: Several moving object segmentation examples tested in our platform

To better evaluate the proposed method, we quantitatively analyze the moving detecting results by comparing them with ground truth. In this experiment, five test sequences (each of them is composed of more than 100 frames) are used for evaluation. We compute the Type I errors<sup>1</sup> (true positive rate and false positive rate) for each sequence, the results are as follows: From Tab. 4.1, we could see that the proposed framework performs well when there are moving vehicles in the scenario. However, the performance degenerates when pedestrians appear. The major reason is that a moving pedestrian is slower than a moving vehicle. Hence, several detected feature points within a moving pedestrian are classified as static. This problem would cause undetected moving pedestrians.

<sup>1</sup>[http://en.wikipedia.org/wiki/Type\\_I\\_and\\_type\\_II\\_errors](http://en.wikipedia.org/wiki/Type_I_and_type_II_errors)



(a) An image of road scene



(b) Moving object segmentation result



(c) An image of road scene



(d) Moving object segmentation result



(e) An image of road scene



(f) Moving object segmentation result

Figure 4.5: More moving object segmentation examples



	True Positive	False Positive	Sequence Length
Sequence 1	97.5%	2.5%	121
Sequence 2	96.5%	3.5%	135
Sequence 3	85.2%	14.8%	217
Sequence 4	93.4%	6.6%	102
Sequence 5	88.9%	11.1%	177

Table 4.1: Type I errors based evaluation

## 4.2 Moving Object Recognition using Spatial Information

In the previous sections, ego-motion of the vehicle itself and independent moving objects are investigated. In this section, we propose a recognition method to classify moving objects (such as pedestrian, vehicle, cyclist) by sparse spatial information.

### 4.2.1 Introduction

Vision-based vehicular perception systems usually consist of two distinct levels of functions which rely on two distinct levels of information, respectively. For the first level, spatial information, either dense or sparse, is always utilized to deal with the problems of motion analysis, including ego-motion estimation, moving object detection and tracking [Nedevski 2007] [Rodriguez F 2009] [Rabe 2007b]. For the second level, some complex appearance features are usually exploited in object recognition and semantic segmentation [Dalal 2005] [Viola 2004] [Leibe 2006]. In the last decades, one trend of vision-based perception systems for intelligent vehicles is breaking through the boundary of the two levels. In [Leibe 2008a], the authors integrated both structure information and complex appearance features (implicit shape model (ISM) detector [Leibe 2008b]) for object detection and tracking. In [Hoiem 2008], the authors improved vehicle, pedestrian detection by considering 3D perspective information. In [Brostow 2008], semantic segmentation of traffic scenarios is completed by only structure and motion information.

Classification of particular types of objects is a dynamic research field. A common scheme of solution contains two steps: At first, extracting and processing certain appearance features (as SIFT [Lowe 1999], HOG [Dalal 2005], Haar-like features [Viola 2004]) from labeled dataset. Second, a supervised machine learning method (as SVM [Burges 1998], AdaBoost [Freund 1995], Random Forest [Breiman 2001], Gradient Boosting [Friedman 2000]) is then applied to train a good classifier for further prediction. However, since the previous approaches heavily rely on appearance features, several researchers tried to introduce geometric information into this field. In [Brostow 2008], the author improves semantic segmentation and recognition by taking advantages of 3D geometric features. In [Hoiem 2008], vehicle and pedestrian detection results are improved with the help of perspective information. In [Zhu 2010], a range image generated by a laser range finder is

segmented and classified by only depth information.

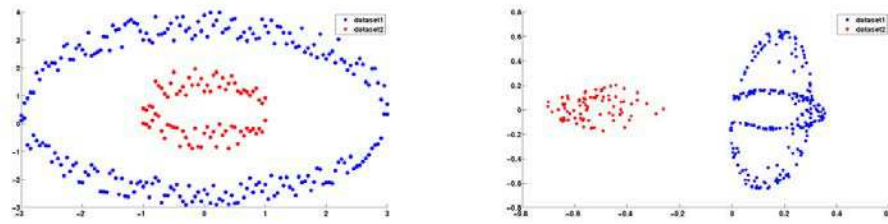
Our proposed method is inspired from [Brostow 2008] and [Zhu 2010]. Our objective is attempting to classify moving objects in urban environments solely by spatial information. The spatial information is generated from segmented pixels in stereo images. Afterward, several basic spatial features as well as advanced spatial features (based on Kernel Principal Component Analysis (KPCA) [Schölkopf 1998]) are extracted. Finally, a classifier which has been trained off-line, is used to predict the categories of the moving objects.

#### 4.2.2 Spatial Feature Extraction and Classification

After the previous steps, a segment of moving object is extracted as a set of pixels. By triangulation, it can be represented as a cloud of 3D points:  $C = \{P_1, \dots, P_M\}$ . In this section, several spatial features are extracted and trained to recognize common moving objects in urban environments.

##### 4.2.2.1 Spatial Feature Extraction

The authors in [Brostow 2008] proposed 5 simple structure features of every 3D point for semantic segmentation. For a cluster of 3D points, [Zhu 2010] utilized 5 simple structure features and 2 statistical features. Although improvements are achieved by these methods, they still leave large space to work in spatial features. In fact, the limitations of their spatial features arise from the intuitive representations of points in Euclidean space. The statistical normal features in [Zhu 2010] is in fact the main directions of a 3D points cluster which is equivalent to doing Principal Component Analysis (PCA) for the points cluster. Traditional PCA is a powerful method for extracting structures from high-dimensional data. However, data of a 3D points cluster is low-dimensional (Euclidean space).



(a) Two linearly inseparable co-centric point clusters (b) The projections of the two clusters in a transformed higher dimensional space using the first two principal components

Figure 4.6: The advantage of data analysis in high dimensional space

With the aid of kernel methods, more and advanced spatial features can be extracted by transforming the feature space from 3D Euclidean space into a higher dimensional space. The reason stems from that two linearly inseparable clusters can

sometimes become linearly separable in a transformed higher space, as shown in Fig. 4.6. The left image shows two clusters which can not be divided by a linear method, while after transforming them into a higher space (by Gaussian kernel), the two clusters are easily to be separated from their projections into the first two principal components. In fact, if  $N$  points cannot be linearly separated in  $d < N$  dimensional space, they always can be almost linearly separated in  $d' \geq N$  dimensional space. For instance, given  $N$  points  $P_i, i = 1, \dots, N$  in  $d$  dimension, if we map them into an  $N$ -dimensional space with by a mapping  $\Phi(\cdot)$ :

$$\Phi(P_i) = \delta_{ij} \quad (4.2)$$

where  $\Phi: \mathbb{R}^d \rightarrow \mathbb{R}^N$  and  $\delta_{ij}$  is the Kronecker delta. It is not difficult to construct a hyperplane that divides the points into arbitrary clusters (see in Fig. 4.6 (b)).

Inspired by this phenomenon, we utilize a Kernel PCA (KPCA) based method [Bo 2011] to improve the features in [Zhu 2010]. The method firstly transforms the 3D points cluster into a higher dimensional space by KPCA. Then, the histogram of top  $L$  eigenvalues of kernel matrix is calculated and taken as advanced spatial features. KPCA is an extension of PCA using kernel methods to map data from low dimension into high dimension. For a points cluster  $\mathbf{C} = \{P_1, \dots, P_M\}$ , where  $P_i = (X_i, Y_i, Z_i)^T$ . The steps of KPCA are as follows:

1. Building the  $M \times M$  kernel matrix  $\mathbf{K}$ , where:

$$\mathbf{K}(i, j) = (\Phi(P_i) \cdot \Phi(P_j)) = \Phi(P_i)^T \Phi(P_j) \quad (4.3)$$

In practice, the commonly used kernel functions are: polynomial kernel:  $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + a)^c$  Gaussian kernel:  $k(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)$ , exponential kernel:  $k(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|)$ .

2. Centering the kernel matrix by:

$$\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_M \mathbf{K} - \mathbf{K} \mathbf{1}_M + \mathbf{1}_M \mathbf{K} \mathbf{1}_M \quad (4.4)$$

where  $\mathbf{1}_M$  is a  $M \times M$  matrix with  $(\mathbf{1}_M)_{ij} = 1/M$ .

3. SVD decomposition of the centered kernel matrix:

$$\tilde{\mathbf{K}} = \mathbf{U} \mathbf{S} \mathbf{V}^* \quad (4.5)$$

4. In the matrix  $\mathbf{S}$ , the singular values of  $\tilde{\mathbf{K}}$  are arranged in the diagonal with decreasing order. The top  $L$  singular values together with their corresponding column vectors in  $\mathbf{S}$  represent the main directions of a transformed points cluster. Hence, two kinds of objects in different shapes would have different distribution of these main directions, as shown in Fig. 4.7. In practice, the top  $L$  singular values  $\lambda_i, i = 1, \dots, L$  are selected and normalized as:

$$f_i = \lambda_i / \sum_i^L \lambda_i \quad (4.6)$$

In Table. 4.2, all the features extracted from the cluster in practice are listed.

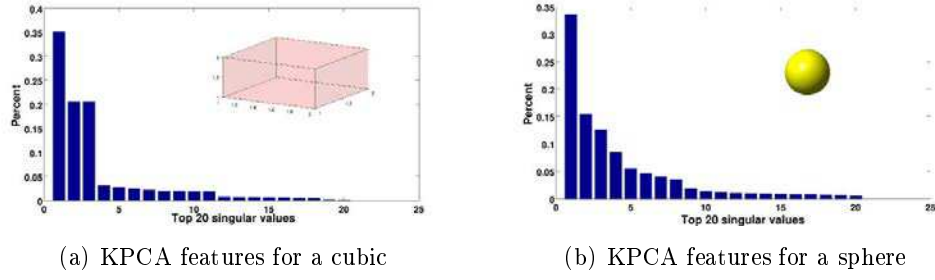


Figure 4.7: The KPCA features for two different objects (Gaussian kernel)

Feature	Definition
$f_1, f_2, f_3$	Cluster center in x,y,z directions
$f_4, f_5, f_6$	Cluster variance in x,y,z directions
$f_7$	Maximal height value in the cluster
$f_8, f_9, \dots, f_{27}$	Top 20 kernel shape descriptors

Table 4.2: Extracted Features

#### 4.2.2.2 Classification

The objective of classification aims to recognize the common moving objects in urban environments, including vehicles, pedestrians and cyclists .... Based on the off-the-shelf machine learning methods, together with the spatial features extracted in the previous step, we adopt two popular machine learning methods (Random Forest [Breiman 2001] and Gradient Boosting Trees [Friedman 2000] [Friedman 1999]) to train classifiers based on the extracted spatial features.

**Random Forest:** Random forest is an ensemble learning framework for classification (or regression) that operates by constructing a multitude of decision trees at training time. The first formal introduction of random forest was given in [Breiman 2001], which describes a method of building a forest of uncorrelated trees using classification and regression trees(CART) [Breiman 1984] like decision trees, combined with randomized node optimization and bagging. In his paper, each tree is constructed using the following steps:

- Let  $T$  be the number of training cases, and the number of variables in the classifier be  $V$ .
- $v$  denotes as the number of input variables to be used to determine the decision at a node of the tree;  $v \ll V$ .
- Choose a training set for this tree by choosing  $n$  times with replacement from all  $T$  available training cases (e.g. take a bootstrap sample). Use the rest of the cases to estimate the error of the tree, by predicting their classes.
- For each node of the tree, randomly choose  $m$  variables on which to base the

decision at that node. Calculate the best split based on these  $m$  variables in the training set.

- Each tree is fully grown and not pruned (as may be done in constructing a normal tree classifier).

For prediction, a new sample is pushed down the tree. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the mode vote of all trees is reported as the random forest prediction.

**Gradient Boosting Trees:** Gradient boosting is another type of ensemble learning method. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. A pseudo code of general gradient boosting is given by Algorithm 4:

---

**Algorithm 4** Generic gradient boosting

---

- 1: Input: training set  $\{(x_i, y_i), i = 1, \dots, n\}$ , a differentiable loss function  $L(y, F(x))$ , number of iterations  $K$
- 2: Initialize the model with a constant value:

$$F_0(x) = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma) \quad (4.7)$$

- 3: **for**  $m=1$  to  $M$ : **do**
- 4:   Compute *pseudo-residuals*:  $r_{im} = -[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}]_{F(x)=F_{m-1}(x)}$  *for*  $i = 1, \dots, n$ .
- 5:   Fit a base learner  $h_m(x)$  to pseudo-residuals.
- 6:   Compute multiplier  $\gamma_m$  by solving the following one dimensional optimization problem:

$$\gamma_m = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)) \quad (4.8)$$

- 7:   Update the model:  $F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$
  - 8: **end for**
  - 9: Output  $F_M(x)$ .
- 

Gradient boosting trees is a special case of gradient boosting by using fix-sized decision trees (especially CART trees) as base learners.

### 4.2.3 Experimental Results

#### 4.2.3.1 Dataset

We tested the proposed recognition method both on KITTI dataset and our dataset. All the stereo images are rectified and the intrinsic and extrinsic parameters of the cameras are also provided. An interactive labeling tool is developed to label the

category of a moving object in the dataset. In all, 2191 samples from 3 categories (Vehicle, Pedestrian, Motor/Bike) are collected. Within the dataset, nearly 70% of samples are treated as training samples, almost 30% are taken as testing samples, see in Table. 4.3. All the testing samples are from our dataset, while the training samples are from both KITTI dataset and our dataset.

	Training samples	Testing samples	Total
Vehicle	696	298	994
Pedestrian	363	156	519
Motor/Bike	380	163	543
Overall	1439	617	2056

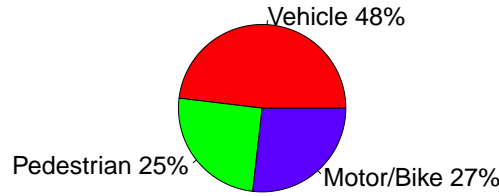


Table 4.3: Experimental dataset

#### 4.2.3.2 Moving Object Recognition Results

The segmentation results from Sec. 4.1 are used for further recognition. For each segment, we reconstruct its pixels' 3D positions by triangulation. The spatial features are extracted according to Tab. 4.2 and sent to trained classifiers for recognition. Three commonly used kernel functions (Polynomial, Gaussian and Exponential Kernel) were adopted. where  $c = 2$ ,  $a = 1$ ,  $\gamma = 0.5$  (see Sec. 4.2.2). We trained two types of classifiers (Random Forest, Gradient Boosting Trees) using the training dataset. The parameters of the two kinds of classifiers (as the depth of a decision tree, the maximum number of trees, the loss function type ...) are optimized by cross validation between the training and test data. Moreover, we also implemented the proposed features proposed in [Zhu 2010] for a comparison. Since the detailed implementation of [Zhu 2010] is not given, we simulate his method as well as we can. All the results are shown in Table. 4.4, which illustrates:

- In theory, Random forest is an example of bagging approach, less prone to overfit. Gradient boosted trees represent the other one. Both are very successful in many applications. While in our case, the Gradient Tree Boosting

classifier performs better than Random Forest classifier in all the cases in our dataset.

- The classification results based on the three kinds of kernels are almost the same. However, the Gaussian kernel performs slightly better than the other two kernels.
- Compared with Zhu's method [Zhu 2010], the proposed kernel shape descriptors improve the classification results thanks to transforming the features into a higher dimensional space. In average, the precision based on the proposed kernel shape descriptors increases by around 10% than in [Zhu 2010].
- The Confusion matrices reveal that pedestrians and cyclists samples are apt to be confused. This is because they are similar in both size and shape.

The experimental results for the moving object recognition are shown in Fig. 4.8.

(a) Classification results and comparison to [Zhu 2010]

	Random Forest (Polynomial Kernel)	Gradient Boosting Tree (Polynomial Kernel)	Random Forest (Gaussian Kernel)	Gradient Boosting Tree (Gaussian Kernel)	Random Forest (Exponential Kernel)	Gradient Boosting Tree (Exponential Kernel)	Random Forest (Zhu's method) [Zhu 2010]	Gradient Boosting Tree (Zhu's method) [Zhu 2010]
Vehicle	83.71%	92.34%	82.25%	97.41%	86.32%	90.66%	81.01%	88.26%
Pedestrian	78.91%	88.68%	79.24%	82.48%	83.58%	80.56%	73.11%	78.11%
Motor/Bike	80.16%	85.64%	78.92%	85.63%	78.97%	82.28%	65.68%	64.31%
<b>Overall</b>	82.98%	89.64%	80.60%	90.47%	83.70%	85.85%	74.97%	79.394%

(b) Confusion matrix for Gradient Tree Boosting

	Vehicle	Pedestrian	Motor/Bike
Vehicle	97.41%	0.55%	2.04%
Pedestrian	1.9%	82.48%	15.62%
Motor/Bike	3.69%	10.68%	85.63%

(c) Confusion matrix for Random Forest

	Vehicle	Pedestrian	Motor/Bike
Vehicle	82.25%	9.01%	8.74%
Pedestrian	6.09%	79.24%	14.67%
Motor/Bike	4.98%	16.10%	78.92%

Table 4.4: The classification results based on the dataset



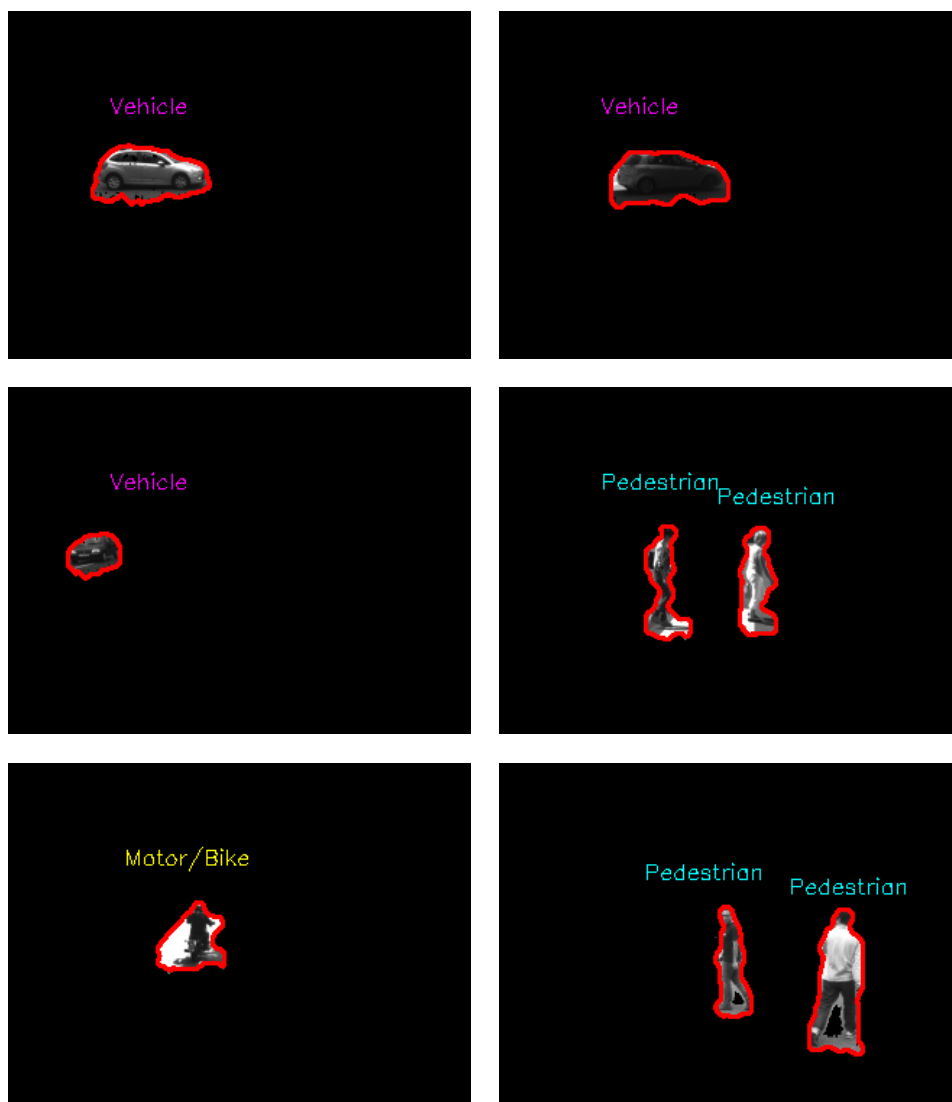


Figure 4.8: Recognition results

### 4.3 Conclusion and Future Works

In this chapter, we proposed stereo vision based methods to detect, segment and recognize independent moving objects, for a moving intelligent vehicle. These methods are based on the ego-motion estimation results in Chap. 3.

In the first part, we presented a stereo vision based method to detect and segment independent moving objects from a moving intelligent vehicle. The method relies on the "outliers" generated in the step of visual odometry. The "outliers" mainly consist of matching noises and feature points from moving objects. Next, they are projected to U-disparity image as seeds for moving object detection and segmentation. This method is tested on our datasets and the experimental results are analyzed. However, the performance of this method degenerates when trying to detect moving pedestrians. The main reason of this problem is that moving pedestrians usually move too slow to effectively detect their motion between two consecutive frames. Future works would be around this problem. One possible solution to this problem is using pedestrian detection methods detect pedestrians at first. Then, if there are any detected moving feature points in detected pedestrian region, it could be classified as a moving pedestrian.

In the second part, a recognition approach based on learning spatial information is proposed. We tried to classify the segmented moving objects acquired by the previous step. In this method, a kernel PCA algorithm is used to select the spatial features in high dimensional space. Next, we trained two kinds of classifiers: Random Forest and Gradient Boosting Trees to recognize three kinds of moving objects: Pedestrian, Vehicle and Motor/Bike. The experimental results showed good recognition rates, especially for the Gaussian kernel based KPCA features combining with Gradient Boosting Trees. Also, we show that with the boosted kernel PCA features, our method outperforms another method by around 10% in terms of precision. In future works, we consider combining spatial features with appearance features to reach a better recognition results. For example, the histogram of oriented gradients (HoG) or local binary patterns (LBP) could be introduced to improve the recognition performance.



# Extrinsic Calibration between a Stereo Vision System and a Lidar

---

## Contents

---

<b>5.1</b>	<b>Introduction</b>	<b>91</b>
5.1.1	Related works	92
5.1.2	Problem Formulation	93
5.1.3	Performance Evaluation	94
<b>5.2</b>	<b>3D Plane Reconstruction based Extrinsic Calibration</b>	<b>95</b>
5.2.1	Corner Points Triangulation	96
5.2.2	3D Plane Estimation	96
5.2.3	Automatic Lidar Measurements Extraction	100
5.2.4	Estimating Rigid Transformation between the lidar and the Stereoscopic System	101
5.2.5	Summary of the Calibration Procedure	103
<b>5.3</b>	<b>Experimental Results</b>	<b>104</b>
5.3.1	Computer Simulations	104
5.3.2	Real Data Test	110
5.3.3	Outdoor Experiments	111
<b>5.4</b>	<b>Conclusion And Future Work</b>	<b>116</b>

---

The previous chapters focus in solving several perception missions only by stereo vision system. A real intelligent vehicle, as introduced in Sec. 1, usually equips with multiple sensors. In this thesis, we also use a 2D lidar to perform the perception tasks. In order to fuse the two kinds of sensor measurement, an extrinsic calibration method aimed to estimate a 3D rigid transformation between the two sensors is proposed and performed.

## 5.1 Introduction

As we introduced in the first chapter, we plan to use stereoscopic system and 2D lidar to analyze the environment for an intelligent vehicle. However, despite many impressive progresses in both computer vision, and range sensors in past decades, no technique based on single kind of sensors can be reliably and robustly performed in

real complex environments. In fact, visual sensors are always limited in narrow field of views (FOV) and require huge computational cost. On the other hand, providing only depth information, the range sensors are not enough for more complex tasks in deep level. Hence, achieving effective and robust environments understanding requires the fusion of different kinds of sensors. Indeed, fusing visual and range data has attracted more and more attentions for intelligent vehicles. In [Mahlisch 2006, Fox 2007, Hwang 2007], various information fusion methods are proposed.

The problem of fusing data between visual cameras and LIDARs arises from the heterogeneity of the sensors. Lidar data is depth information in a certain scan plan, while the data of a camera can be viewed as a projection of a 3D world into an image plane. Thus, how to bridge the gap between a visual camera and a LIDAR has to be solved before performing data fusion. An initial solution is to estimate a rigid geometrical transformation (a  $3 \times 3$  rotation matrix and a  $3 \times 1$  translation vector) between lidar coordinate system and stereo vision coordinate system – which is usually called extrinsic calibration between lidar and stereo vision system. In general, it is implemented as the first step for further information integration of the two sensors, as illustrated in [Perrollaz 2006, Cristiano 2009, Huang 2009b].

### 5.1.1 Related works

In literature, most proposed methods for extrinsic calibration between a lidar and a camera can be roughly divided into three categories:

1. *Methods based on auxiliary sensors.* In [Aliakbarpour 2009], with the aid of an IMU (Inertial Measurement Unit) and a visible laser point, the Euclidean 3-dimensional transformation between a lidar and a camera is acquired by rotating the scan plane, where the angular information is estimated by the inertial sensor. Nunez *et al.* [Nunez 2009] also use an IMU to construct geometric constraints between a lidar and a camera system for calculating the transformation. However, being equipped with an IMU is a prerequisite for this kind of approaches.
2. *Methods based on specially designed calibration boards.* Li *et al.* [Li 2007] design a right-angled triangular checkerboard and employ the invisible intersection points of the lidar's scan plane with the edges of the checkerboard to set up constraint equations. Rodriguez *et al.* [Rodriguez F 2008] adopt a calibration board with two concentric circles where the inner circle is hollowed. The authors use the correspondence of the center of the inner circle between several different poses to achieve extrinsic calibration. By making use of special features in the designed calibration board, such approaches enable one to acquire the geometric relationship between a lidar and a camera. However, specially designed calibration boards impede themselves for more widespread uses.
3. *Methods based on common calibration chessboards.* Zhang and Pless [Zhang 2004] firstly proposed a method utilizing an ordinary calibration chessboard which is widely used in camera calibration. This method is based on

the fact that lidar measurements on the calibration chessboard satisfy certain geometric constraints with the plane of the chessboard. Following Zhang’s idea [Zhang 2004], Huang and Barth [Huang 2009a] utilize similar geometric constraints and generalize them into a situation of multi-layer lidar. Vasconcelos et al [Vasconcelos 2012] improve Zhang’s method [Zhang 2004] by reducing the minimum number of various poses from 6 to 5. This kind of methods can handle with intrinsic calibration for cameras and extrinsic calibration between a lidar and a camera at the same time. Also, these methods require no extra equipments or specially designed calibration boards. Thus, in our point of view, they are more practical in real applications.

According to the above analysis, we prefer the third approach for its simplicity. Since stereovision can provide a complete three-dimensional view of a scene, its application for intelligent vehicles is more and more comprehensive nowadays. However, all the kinds of methods mentioned above are applied between a monocular camera and a 2D lidar. Calibrating a stereo vision system with a lidar can be performed by calibrating each camera with the lidar separately, as all existing methods perform, or more reasonably, by calibrating the considered stereovision system with the lidar. Based on 3D reconstruction of a common calibration board, we propose a novel calibration approach for obtaining the relative position and orientation between a stereo vision system and a lidar. It is worthy to mention that the advantages of our approach are threefold: Compared with the first and second categories of methods, our method is more convenient since it uses one standard chessboard; Compared with the third category, we will show that owing to the consideration of the whole stereovision system, the proposed extrinsic calibration algorithm performs more precisely. Moreover, the introduction of Mahalanobis distance constraints and the consideration of sensor noise models make the calibration results more accurate and robust.

### 5.1.2 Problem Formulation

A geometric model of the multisensor system together with the 2D chessboard is given in Fig. 5.1. As in Sec. 2.1.1 and Sec. 2.1.3.1, we set  $\mathbb{R}_{left}^3$ ,  $\mathbb{R}_{right}^3$ ,  $\mathbb{R}_{lidar}^3$  as left camera, right camera and lidar coordinate system,  $\mathbb{R}_{left}^2$  and  $\mathbb{R}_{right}^2$  as left image, right image coordinate system, respectively. Let the left camera coordinate system be the stereo vision coordinate system:  $\mathbb{R}_{stereo}^3 = \mathbb{R}_{left}^3$ .  $\mathbb{R}_{right}^3$  is linked to the stereoscopic coordinate system  $\mathbb{R}_{stereo}^3$  by a rigid transformation composed of a rotation matrix  $\mathbf{R}_r^l$  and a translation vector  $\mathbf{T}_r^l$ . Suppose a point  $P$  in the right camera coordinates as:  $P_r = (X_r \ Y_r \ Z_r)^T$ , then, its corresponding coordinates in the stereoscopic system  $\mathbb{R}_{stereo}^3$  are:

$$P_{stereo} = P_l = \mathbf{R}_r^l \cdot P_r + \mathbf{T}_r^l \quad (5.1)$$

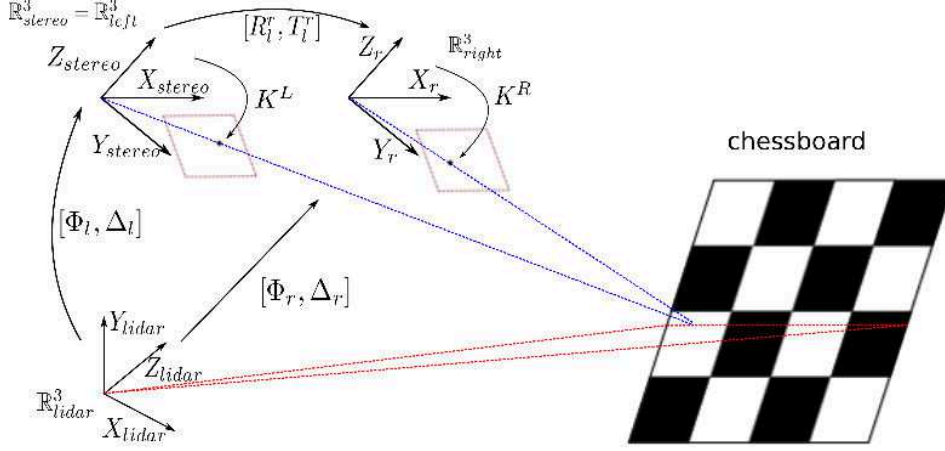


Figure 5.1: Multi-sensor system: geometric relationships between the sensors coordinate systems

In a similar way, a point in the stereoscopic coordinate system can be mapped into the right camera coordinate system as Eq. (5.1), by substituting  $\mathbf{R}_r^l$  and  $\mathbf{T}_r^l$  as:

$$\begin{aligned}\mathbf{R}_l^r &= \mathbf{R}_r^l{}^T \\ \mathbf{T}_l^r &= -\mathbf{R}_r^l{}^T \cdot \mathbf{T}_r^l\end{aligned}\tag{5.2}$$

Let  $(\Phi_l, \Delta_l)$  and  $(\Phi_r, \Delta_r)$  be the 3D rigid transformation from the LIDAR coordinate system  $\mathbb{R}_{lidar}^3$  to the left and right camera coordinate systems  $\mathbb{R}_{left}^3$  and  $\mathbb{R}_{right}^3$  respectively, where  $\Phi_l$  and  $\Phi_r$  are orthogonal rotation matrixes,  $\Delta_l$  and  $\Delta_r$  are translation vectors. Suppose a 3D point  $P$  observed by both the LIDAR and the stereoscopic system, and denoted as  $P_{lidar} = (X_{lidar} \ Y_{lidar} \ Z_{lidar})^T$  in  $\mathbb{R}_{lidar}^3$ ,  $P_l$  and  $P_r$  in the left and right camera coordinate systems. The three coordinates are connected by:

$$\begin{aligned}P_l &= \Phi_l \cdot P_{lidar} + \Delta_l \\ P_r &= \Phi_r \cdot P_{lidar} + \Delta_r \\ P_l &= \mathbf{R}_r^l \cdot P_r + \mathbf{T}_r^l\end{aligned}\tag{5.3}$$

### 5.1.3 Performance Evaluation

In practice, it is difficult to obtain the ground truth of the real extrinsic parameters between a camera and a lidar. In [Zhang 2004], [Huang 2009b], the extrinsic calibration methods are evaluated just by intuition (whether the projected laser points in images are reasonable or not). In this paper, we define an indicator  $\varepsilon$  as a precision measure of the multi-sensor calibration system:

$$\varepsilon^2 = \frac{1}{L} \sum_i^L \|\mathbf{R}_r^l \cdot (\Phi_r P_{lidar}^i + \Delta_r) + \mathbf{T}_r^l - (\Phi_l P_{lidar}^i + \Delta_l)\|^2\tag{5.4}$$

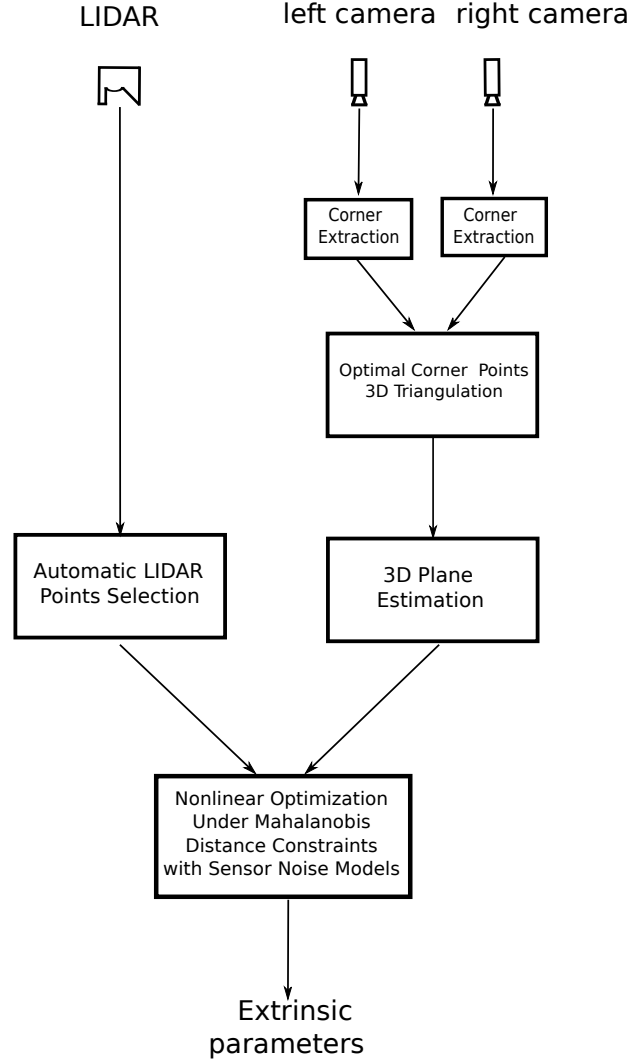


Figure 5.2: Flow diagram of the proposed calibration method

where  $L$  refers to the number of lidar points on the chessboard used in the extrinsic calibration.  $\varepsilon$  is not a direct error with respect to the ground truth. Indeed, it can be viewed as a precision measure evaluating the conformity of the results with respect to the geometric relationships within the multi-sensor system. We hope  $\varepsilon$  to be as small as possible.

## 5.2 3D Plane Reconstruction based Extrinsic Calibration

A flowchart in Fig. 5.2 illustrates all the steps of our method. It requires putting a calibration chessboard in front of all the sensors at different positions and orientations and make sure that all the sensors can detect it. For each pose, acquired



images are undistorted and rectified as in Sec. 2.2.3.2. Afterwards, we automatically extract the corner points of the chessboard in the images of the two cameras, and triangulate their 3D positions in the stereoscopic coordinate system. Then, from the reconstructed points, a PCA based method is used to estimate the 3D plane, which best fits the reconstructed points of the chessboard in the stereoscopic coordinate system. Finally, considering the geometrical constraints, a non-linear 3D minimization is carried out in order to calculate the extrinsic parameters. Our stereoscopic system delivers rectified stereo image pairs automatically, all the intrinsic/extrinsic parameters of the stereo-rig are calculated by [Zhang 2000]’s method (implemented in [Bouguet 2010]). Therefore, the intrinsic/extrinsic parameters  $\mathbf{K}_l, \mathbf{K}_r, \mathbf{R}_r^l, \mathbf{T}_r^l$ , are known and  $\mathbf{K}_l = \mathbf{K}_r$ .

### 5.2.1 Corner Points Triangulation

In Sec. 2.2.4, a two-view triangulation method in ideal situation is given. However, in general cases, the two cameras can be rotated with respect to each other. Furthermore, due to the inevitable imaging noises, the two rays  $O_lP$  and  $O_rP$  are not guaranteed to be intersected. The process of 3D triangulation becomes complicate in such cases. A common method (middle point method) [Beardsley 1994] is to use the middle point of the perpendicular to the two rays, as drawn in Fig. 5.3 (a). Beyond a simple middle point method, [Hartley 1997] proposes an optimal triangulation method to minimize projection errors of corresponding points under epipolar constraint. In [Kanatani 2008], a method is proposed to correct the positions of corresponding to achieve optimized triangulation results. Fig. 5.3 (b) illustrates the case of optimal triangulation. More details about 3D triangulation in general cases could be found in [Hartley 2004].

In 3D computer vision practices, rectifying stereo image pairs is a necessary preprocessing step to simplify the succeeding 3D reconstruction. In our approach, input stereo images are all undistorted and rectified. Afterward, we apply a sub-pixel corner point extraction [Bouguet 2010] to get sets of corresponding pairs of corner points  $p_l = (u_l, v_l)$  and  $p_r = (u_r, v_r)$ . 3D reconstruction is performed according to Eq. 2.23. Let  $P_l$  be the 3D point coordinates in the left camera coordinate system,  $P_r$  be its corresponding coordinates in the right camera coordinate system.

By triangulating all the corner point pairs extracted from the two views, the corresponding 3D coordinates attached to the left and right camera coordinate systems are acquired. Although errors exist in the results of triangulation, the 3D points cloud is nearly flat. The next step is to estimate a 3D plane that best fits all the reconstructed 3D points.

### 5.2.2 3D Plane Estimation

A plane in 3D Euclidean space can be defined by a known point on the plane and a normal vector perpendicular to its surface. In homogeneous coordinate system, the

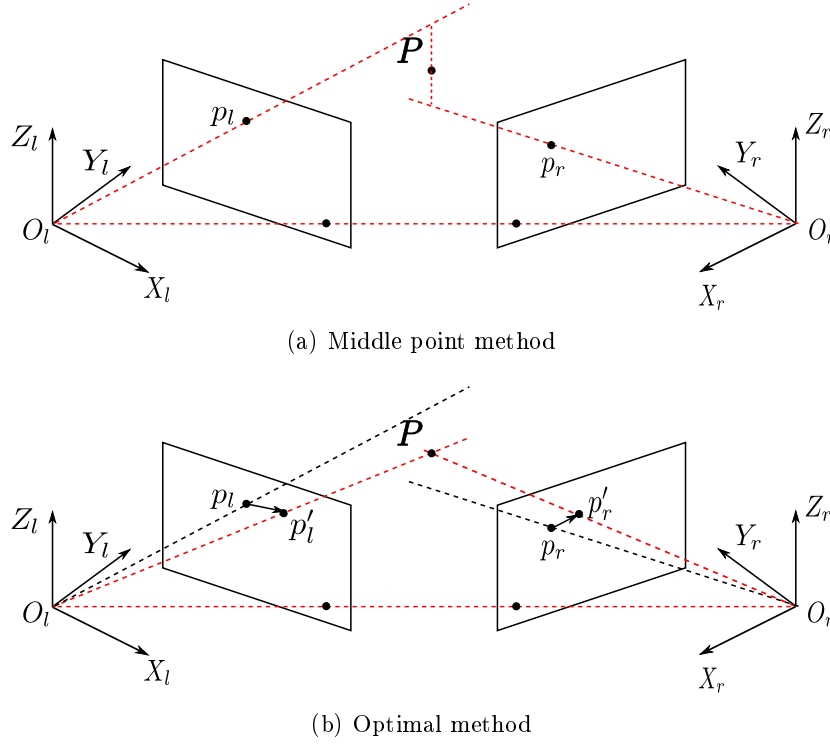


Figure 5.3: Middle point and optimal triangulation methods

equation of the plane can be expressed as:

$$[\vec{n}, d] \cdot [x, y, z, 1]^T = 0 \quad (5.5)$$

where  $\vec{n} = [a, b, c]$  is a  $1 \times 3$  normal vector ( $\sqrt{a^2 + b^2 + c^2} = 1$ ),  $d = -\vec{n} \cdot [x_0 \ y_0 \ z_0]^T$ ,  $[x_0 \ y_0 \ z_0]$  is a known point on the plane.  $d$  represents the perpendicular distance from the origin to the plane. More simple, Eq. 5.5 can be written as:

$$ax + by + cz + d = 0 \quad (5.6)$$

In mathematics, given a set of 3D points  $\mathbf{S} = \{P^1, \dots, P^i, \dots, P^N | i = 1, \dots, N\}$ , evaluating its best fitting plane is a classic linear regression problem. As introduced in Sec. 2.3.1, linear least squares methods can be applied to solve this problem.

**Ordinary Least Squares Plane Fitting:** As described in Sec. 2.3.1.1, ordinary least squares (OLS) assumes that only the predicted variables have errors. In 3D plane fitting, the problem becomes to finding plane coefficients to minimize a summation of residuals:

$$\sum_{i=1}^N |z_i - (a'x_i + b'y_i + d')|^2 \quad (5.7)$$

where  $a' = -a/c, b' = -b/c, d' = -d/c$ . For convenience, the plane equation represents in Eq. 5.6 is changed as

$$z = a'x + b'y + d' \quad (5.8)$$

According to Eq. 2.32, let  $\mathbf{P} = [\mathbf{x}, \mathbf{y}, \mathbf{1}]^T$  be a  $N \times 3$  observational matrix,  $\mathbf{z}$  be a  $N \times 1$  predicted vector. The OLS solution of 3D plane fitting is:

$$\begin{bmatrix} a' \\ b' \\ d' \end{bmatrix} = (\mathbf{P}^T \mathbf{P})^{-1} \mathbf{P}^T \mathbf{z} \quad (5.9)$$

Supposing the predicted variable  $z$  is contaminated by independent and identically distributed Gaussian noises with variance  $\sigma$ , the covariance of the plane coefficients is given by Eq. 2.34 as:

$$\mathbf{Q} = (\mathbf{P}^T \mathbf{P})^{-1} \sigma^2 \quad (5.10)$$

**Total Least Squares Plane Fitting:** Recall to Sec. 2.3.1.3, total least squares extends the assumption of ordinary least squares by considering errors in all variables. According to Eq. 2.49, TLS computes a right singular vector corresponding to the minimum singular value of an augmented matrix. The TLS model for 3D plane fitting is:

$$\mathbf{z} + \mathbf{n}_z = (a', b', d') \cdot \left( \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ 1 \end{bmatrix} + \begin{bmatrix} \mathbf{n}_x \\ \mathbf{n}_y \\ 0 \end{bmatrix} \right) \quad (5.11)$$

where  $\mathbf{n}_x, \mathbf{n}_y, \mathbf{n}_z$  are the errors in all the observational data  $(X, Y, Z)$ . Recall to Eq. 2.48, the TLS 3D plane fitting problem is:

$$\begin{aligned} & \underset{\mathbf{n}_x, \mathbf{n}_y, \mathbf{n}_z}{\text{minimize}} \quad \|[\mathbf{n}_x, \mathbf{n}_y, \mathbf{n}_z]\|_F \\ & \text{subject to} \quad \mathbf{z} + \mathbf{n}_z = (a', b', d') \cdot \left( \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ 1 \end{bmatrix} + \begin{bmatrix} \mathbf{n}_x \\ \mathbf{n}_y \\ 0 \end{bmatrix} \right) \end{aligned} \quad (5.12)$$

Let  $\mathbf{C} = [\mathbf{x}, \mathbf{y}, \mathbf{1}, \mathbf{z}]$  be a  $N \times 4$  augmented matrix comprising all the observational data. Then, we perform a singular value decomposition to  $\mathbf{C}$ :

$$\mathbf{C} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T \quad (5.13)$$

where  $\mathbf{U}$  and  $\mathbf{V}$  are  $4 \times 4$  real *unitary matrices*.  $\mathbf{\Sigma}$  is a  $4 \times 4$  diagonal matrix with eigenvalues on the diagonal in decreasing order:  $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ . Let  $\mathbf{v}_i$  be the  $i$ th column of  $\mathbf{V}$ . Then,  $\mathbf{v}_4$  is the eigenvector corresponding to the minimum eigenvalue. According to Eq. 2.49, the TLS solution for 3D plane fitting is:

$$\begin{bmatrix} a' \\ b' \\ d' \end{bmatrix} = -\frac{\mathbf{t}}{\alpha} \quad (5.14)$$

where  $\mathbf{t}$  is the first three elements of  $\mathbf{v}_4$ ,  $\alpha$  is the fourth element of  $\mathbf{v}_4$ .

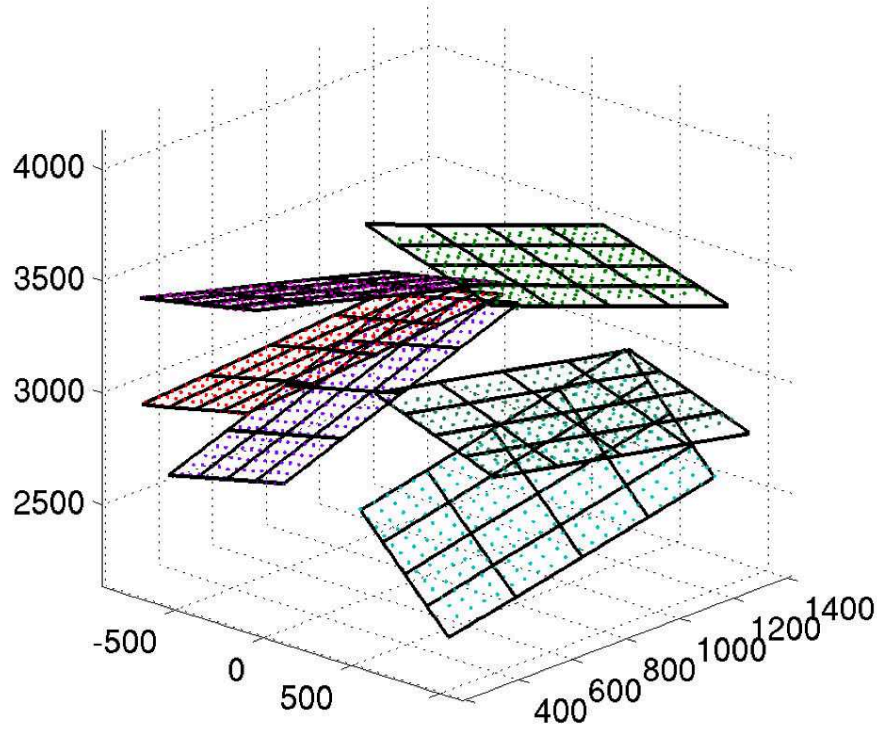


Figure 5.4: Estimated planes for different clouds of 3D points by total least squares

Assuming the errors in all observational data  $(X, Y, Z)$  are independent identical distributed zero-mean Gaussian noises:  $\mathbf{n}_x, \mathbf{n}_y, \mathbf{n}_z \sim \mathcal{N}(0, \sigma)$ , the covariance matrix of TLS estimation results is approximated according to Eq .2.50:

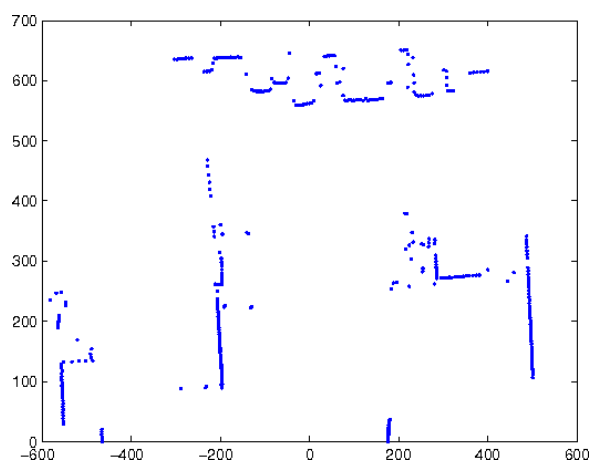
$$\mathbf{Q} = \frac{1}{\gamma} \sigma^2 \|(a', b', d')\|^2 (\mathbf{M}^T \mathbf{M})^{-1} \quad (5.15)$$

where  $M = [\mathbf{x}, \mathbf{y}, \mathbf{1}]$  is a  $N \times 3$  matrix. In fact, the TLS method in 3D plane fitting equals to principal Component Analysis (PCA) [Wall 2012].

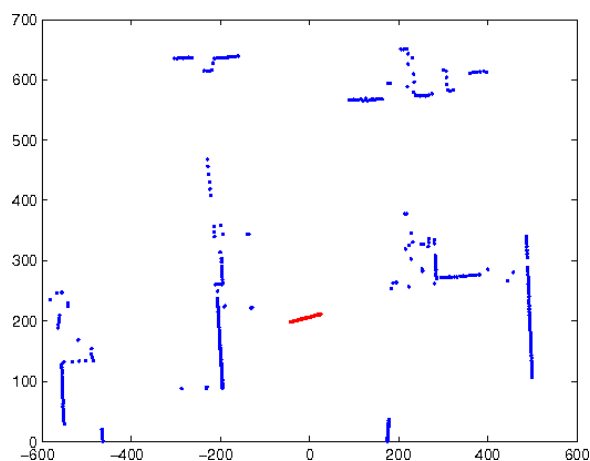
Applying the method described above, 3D plane estimation is performed for various positions and orientations of the calibration board. Fig. 5.4 illustrates an example of six 3D planes estimated according to six different sets of reconstructed 3D points. The black frames represent the estimated planes and the colored points within the frames are the reconstructed 3D corner points.

### 5.2.3 Automatic Lidar Measurements Extraction

In [Zhang 2004] and [Huang 2009b], the ways of extracting lidar points on the calibration board are not mentioned. Here, we apply an automatic extraction approach by differencing the acquired measurements and background data in a static environment. The background lidar measurements are acquired before starting calibration. Then, the measurements acquired during the calibration process are compared with the background by a simple subtraction operation to get the lidar points on the chessboard. According to [Ye 2002], when a laser ray hits on the edge of the chessboard, the measurement would be the average distance between the chessboard and the background. Consequently, the lidar points in the edge of the chessboard which have unreasonable measurements should be discarded. The final results are shown in Fig. 5.5.



(a) The background



(b) Detected foreground (in red point)

Figure 5.5: Automatic lidar measurements extraction

### 5.2.4 Estimating Rigid Transformation between the lidar and the Stereoscopic System

The steps presented above allow us to estimate 3D planes of the calibration board with various poses ( $j = 1, \dots, M$ ). In ideal situations, every lidar measurement on the calibration board should satisfy the plane equation of the calibration board:

$$\vec{N} \cdot \tilde{P}_{stereo} = \vec{N} \cdot \mathbf{H} \cdot \tilde{P}_{lidar} = 0 \quad (5.16)$$

where  $\mathbf{H} = \begin{bmatrix} \Phi_l & \Delta_l \\ 0 & 1 \end{bmatrix}$  denotes 3D rigid transformation which we want to estimate.  $\vec{N} = [\vec{n}, d] = [a, b, c, d]$  is the estimated 3D plane of the chessboard.

However, in real experiments, Eq. 5.16 is rarely satisfied. Given an estimated rigid transformation  $\mathbf{H}$ , all the residuals can be represented as:

$$\mathbf{e} = \{\vec{N}_1 \mathbf{H} \tilde{P}_{lidar}^{1,1} \dots \vec{N}_1 \mathbf{H} \tilde{P}_{lidar}^{L_1,1} \dots \vec{N}_M \mathbf{H} \tilde{P}_{lidar}^{1,M} \dots \vec{N}_M \mathbf{H} \tilde{P}_{lidar}^{L_M,M}\} \quad (5.17)$$

where  $\tilde{P}_{lidar}^{k,j} (k = 1, \dots, L_j, j = 1, \dots, M)$  is the homogeneous coordinate of the  $k$ th lidar point on the chessboard at the  $j$ th pose.  $\vec{N}_j (j = 1, \dots, M)$  is the 3D plane of the chessboard put at the  $j$ th pose.

Although directly minimizing a summation of squared errors in Eq.(5.17) makes sense, it doesn't take into account enough the errors produced either from the 3D plane equations estimated by the stereovision system or from the lidar itself. Considering the error of 3D triangulation grows quadratically, this could make results worse when the calibration board is far away from the stereovision system. To make the method more robust against errors, we propose to minimize a summation of squared Mahalanobis distance, rather than Euclidean distance. Compared with Euclidean metric, the Mahalanobis distance benefits can be viewed as follows: firstly, it corrects the scale factor. In our work, the errors come from both stereoscopic system and lidar. Euclidean distance is sensitive to the scale of various kinds of variables involved. For Mahalanobis distance, thanks to the measure of covariance, the problem of scale factor is resolved. Secondly, Mahalanobis distance corrects the correlation between variables. This advantage is also a consequence of calculating covariance. Assuming  $\mathbf{e} \sim \mathcal{N}(0, \mathbf{C})$ , then the optimization of extrinsic parameters based on Mahalanobis distance becomes:

$$H(\Phi_l, \Delta_l) = \arg \min_{\Phi_l, \Delta_l} \{\mathbf{e}^T \mathbf{C}^{-1} \mathbf{e}\} \quad (5.18)$$

where  $\mathbf{C}$  is the covariance matrix of  $\mathbf{e}$ . Assuming the elements of  $e$  in Eq.(5.17) are

mutually independent, then  $\mathbf{C}$  is a diagonal matrix as:

$$\mathbf{C} = \begin{bmatrix} C_{1,1} & & & & & \\ & \ddots & & & & \\ & & C_{L_1,1} & & & \\ & & & \ddots & & \\ & & & & C_{1,M} & \\ & 0 & & & & \ddots \\ & & & & & & C_{L_M,M} \end{bmatrix} \quad (5.19)$$

where  $C_{k,j} = \text{var}(\vec{N}_j \mathbf{H} \tilde{P}_{lidar}^{k,j})$ . Meanwhile, assuming the plane estimation  $\vec{N}$  and measurement  $P_{lidar}$  are two independent random vectors with multinormal distribution  $\mathcal{N}(\vec{N}, \Sigma_{\vec{N}})$  and  $\mathcal{N}(\bar{P}_{lidar}, \Sigma_{P_{lidar}})$ . A closed-form solution for every element of  $C$  is complex. Noticing that we don't know  $\mathbf{H}$  before computation. Meanwhile,  $\mathbf{H}$  plays a role as scale factor when expanding  $C_{k,j}$ . Here, we approximate  $C$  by ignoring the middle term  $\mathbf{H}$ :

$$\begin{aligned} C_{k,j} &\approx \text{var}(\vec{N}_j \tilde{P}_{lidar}^{k,j}) \\ &= \vec{N}_j^T \Sigma_{P_{lidar}^{k,j}} \vec{N}_j + \bar{P}_{lidar}^{k,jT} \Sigma_{\vec{N}_j} \bar{P}_{lidar}^{k,j} + \text{Tr}(\Sigma_{P_{lidar}^{k,j}} \Sigma_{\vec{N}_j}) \end{aligned} \quad (5.20)$$

where  $\text{Tr}(\ast)$  is the trace of a matrix. Hence, the covariance matrix can be approximated with the distributions of  $\mathcal{N}(\vec{N}, \Sigma_{\vec{N}})$  and  $\mathcal{N}(\bar{P}_{lidar}, \Sigma_{P_{lidar}})$ . The covariance matrix of lidar measurement is inferred in Sec. 2.1.2 and given in Eq. 2.6. For the covariance matrix of estimated 3D plane coefficients, Eq. 5.15 gives a covariance matrix for the coefficients  $(a', b', d')$ . Since in our work, the plane coefficient  $c$  is always set to  $-1$ , corresponding covariance matrix of plane coefficients is:

$$\Sigma_{\vec{N}} = \begin{bmatrix} Q_{11} & Q_{12} & 0 & Q_{13} \\ Q_{21} & Q_{22} & 0 & Q_{23} \\ 0 & 0 & 0 & 0 \\ Q_{31} & Q_{32} & 0 & Q_{33} \end{bmatrix} \quad (5.21)$$

where  $Q_{ij}$  refers to the  $(i, j)$ th element in Eq. 5.15.

After obtaining the covariance matrix  $\mathbf{C}$  in Eq. 5.19, we adopt the Levenberg-Marquardt algorithm to find the optimized  $\mathbf{H}$ , which minimizing the summation of residuals as in Eq. 5.18.

### 5.2.5 Summary of the Calibration Procedure

The proposed calibration procedure is summarized below in the form of Algorithm 5.

---

**Algorithm 5** 3D Plane Reconstruction Based Extrinsic Calibration

---

- 1: **for**  $j = 1$  to  $M$  (number of poses) **do**
  - 2:   Reconstruct their 3D position in the stereoscopic coordinate system by optimal triangulation, as presented in section 5.2.1
  - 3:   Estimate the best fitting plane  $[\vec{N}_j, d_j]$ , as stated in section 5.2.2
  - 4:   Automatically select the points  $\{P_{lidar}^{k,j} | k = 1, 2, \dots, L_j\}$  on the calibration board from lidar measurements
  - 5:   For every selected lidar point and the estimated plane, compute their covariance matrixes according to Eq. (2.6) and Eq. (5.21), respectively
  - 6: **end for**
  - 7: Given a first guess,  $[\Phi_0, \Delta_0]$ , use LM-algorithm to reach a convergence with  $[\Phi, \Delta]$  minimizing the objective function of Eq. (5.18)
  - 8: **Return**  $[\Phi, \Delta]$
-



## 5.3 Experimental Results

The proposed algorithm has been implemented with Matlab and tested using both computer simulations and real data sets. Moreover, a comparison between our approach and a popular camera/lidar calibration method [Zhang 2004] is given.

### 5.3.1 Computer Simulations

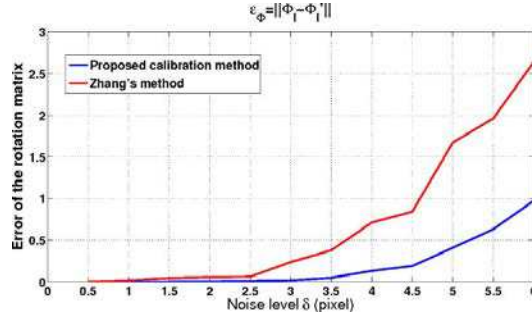
Ground truth of the 3D rigid transformation  $(\Phi_l, \Delta_l)$  is set by rotation vector  $[0.5^\circ, 2^\circ, -3^\circ]^T$ , and the translation vector  $\Delta_l = [0.5, 1.2, -0.3]^T$  (in meters). The intrinsic parameters of the two cameras are the same:  $f_x = 1040, f_y = 1050$  (focal lengths in pixels), principal point is  $(600, 450)$  (in pixels) and skewness coefficient  $\gamma = 0$ . The extrinsic parameters within the stereoscopic system are set by the rotation vector  $[0.1^\circ, -0.1^\circ, 0.2^\circ]^T$ , and the translation vector  $\mathbf{T}_l^r = [0.28, 0.04, -0.02]^T$  (in meters). We set several virtual 3D planes and select some points from each plane, project them into the images of the left and right cameras. The 3D points whose projections are within the left and right images are collected. Virtual lidar measurements on calibration board planes are acquired at the intersection of the chessboard plane and the scan plane of the lidar.

### 5.3.1.1 Performance w.r.t the image noise

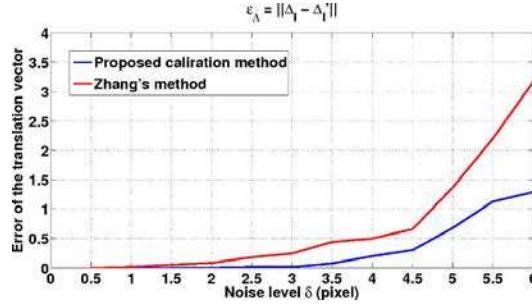
Gaussian noise with zero mean and standard deviation  $\sigma$  (from 0.5 to 6 pixels) is added to the virtual corner points. Based on those "noised" corner points, the proposed method is compared with Zhang's method [Zhang 2004]. The errors are computed as:

$$\varepsilon_{\Phi_l} = \|\Phi_l - \Phi'_l\|, \quad \varepsilon_{\Delta_l} = \|\Delta_l - \Delta'_l\| \quad (5.22)$$

where  $\Phi_l$  and  $\Delta_l$  are the ground truth values,  $\Phi'_l$  and  $\Delta'_l$  are the estimated values. Since the results concerning with the right camera are almost the same, we only measure the error of the left camera. The results are shown in Fig. 5.6 (a), (b). It's clear to see that our calibration method performs the best and tolerance of image noises is around 3 pixels.



(a) Rotation matrix error w.r.t noise level

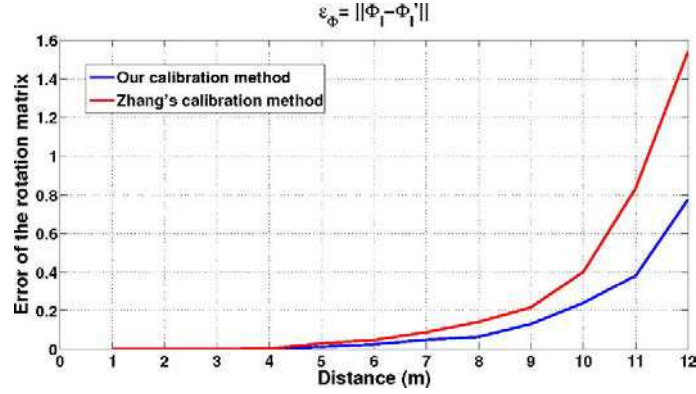


(b) Translation vector error w.r.t noise level

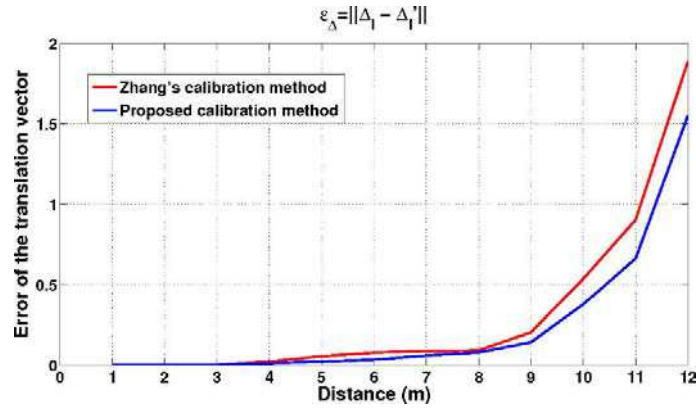
Figure 5.6: Rotation and translation errors w.r.t the noise level

### 5.3.1.2 Performance w.r.t the distance

An experiment is conducted to explore an proper range for the position of the calibration board. We set 6 different calibration board poses within a range of 1m. This is repeated at different distances within the range  $[2m, 12m]$  from the stereoscopic system. The calibration results are shown in Fig. 5.7, which reveals that the best range is within 8 meters.



(a) Rotation matrix error w.r.t distance

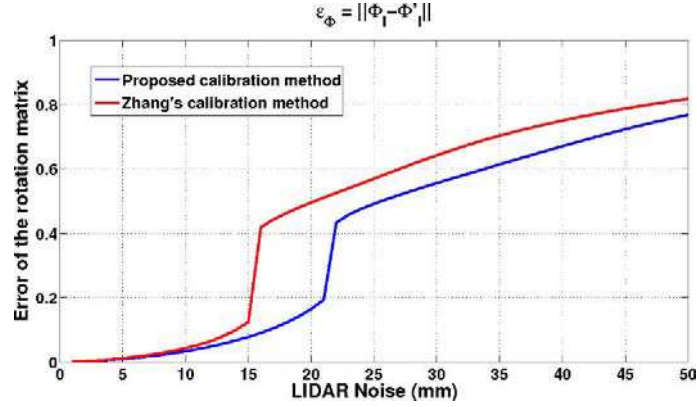


(b) Translation vector error w.r.t distance

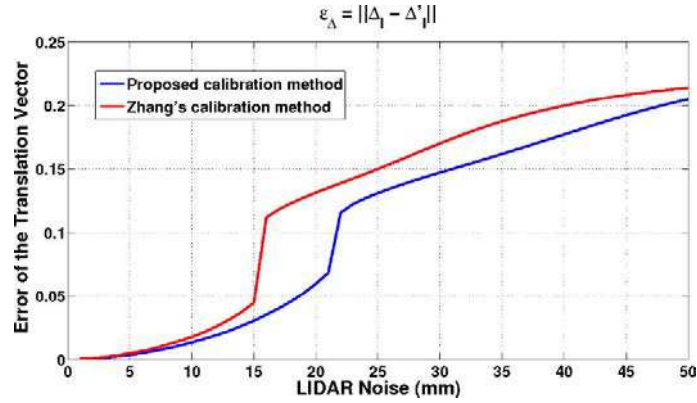
Figure 5.7: Rotation and translation errors w.r.t the noise level

### 5.3.1.3 Performance w.r.t lidar noise

The lidar points are "polluted" by zero-mean Gaussian noises of range and angle measurements  $n_r$  and  $n_\theta$  with variances  $\delta_r$  and  $\delta_\theta$ , respectively.  $\delta_\theta$  is set to 0.0001 degree while  $\delta_r$  varies from 0 to 50 (mm). The simulation results are shown in Fig. 5.8. One can see that our method makes the calibration results more robust, when compared with Zhang's method [Zhang 2004].



(a) Rotation matrix error w.r.t lidar noise

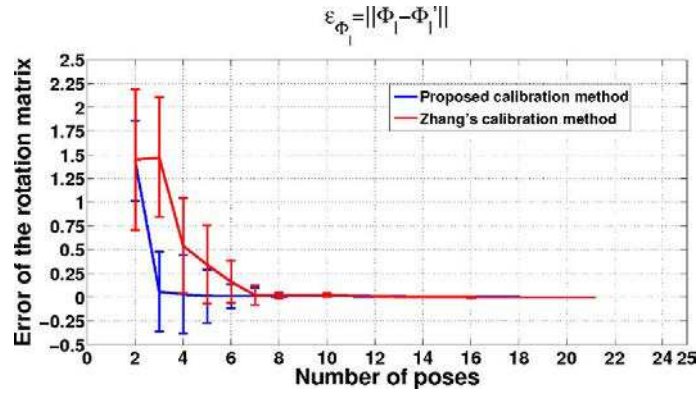


(b) Translation vector error w.r.t lidar noise

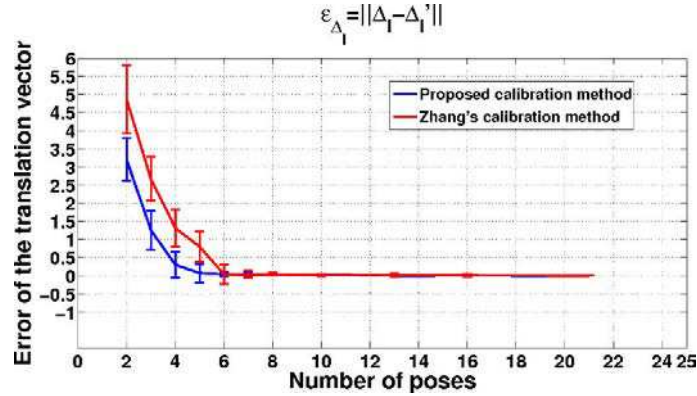
Figure 5.8: Rotation and translation errors w.r.t the noise level

#### 5.3.1.4 Performance w.r.t the number of calibration board poses

We randomly generate 22 different 3D calibration board planes 10 times. By gradually increasing the number of various poses of the chessboard from 2 to 22 in each run, we record the results and calculate the errors. We also test Zhang's method [Zhang 2004] for comparison. We run the tests 10 times with various sets of chessboard poses. The results are averaged, and shown in Fig. 5.9. The error bars represent the variance of 10 different tests. Good results can be obtained when the number of poses is larger than 6.



(a) Rotation matrix error w.r.t the number of calibration board poses



(b) Translation vector error w.r.t the number of calibration board poses

Figure 5.9: Rotation and translation errors w.r.t the number of poses

### 5.3.1.5 Performance w.r.t the first guess in LM algorithm

Although Levenberg-Marquardt (LM) algorithm is very robust, it is required to evaluate the influence of the first guess on the final results. In this experiment, since it is impossible to exhaustively try all the random first guesses, we test all the factors in a first guess independently. For the initial rotation guesses, we start from a  $3 \times 3$  identity matrix, then independently change the yaw, pitch and roll angles [wik 2012] from  $0^\circ$  to  $90^\circ$  with an interval of  $5^\circ$  as:

$$R_{guess} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot R_x(\gamma) \cdot R_y(\alpha) \cdot R_z(\beta) \quad (5.23)$$

where  $\gamma, \alpha, \beta$  are the yaw, pitch and roll angles. For the initial translation guesses, we start from  $[0, 0, 0]^T$  (in meter), then gradually increase the  $X, Y, Z$  values independently to  $[10, 10, 10]^T$  (in meter) with an interval of 0.5 meter. Results of errors according to various first guess values are shown in Fig. 5.10. It is shown that LM algorithm is quite robust. Indeed, even for a first guess that is far from the expected value, it always produces solutions close to the ground truth.

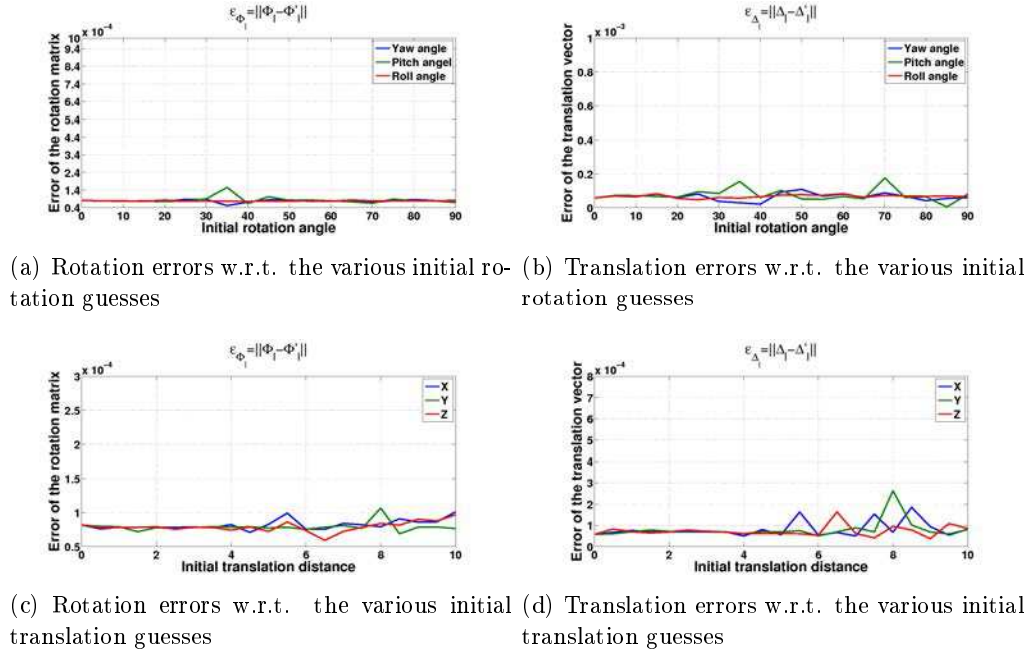


Figure 5.10: Rotation and translation errors w.r.t the initial guess

### 5.3.2 Real Data Test

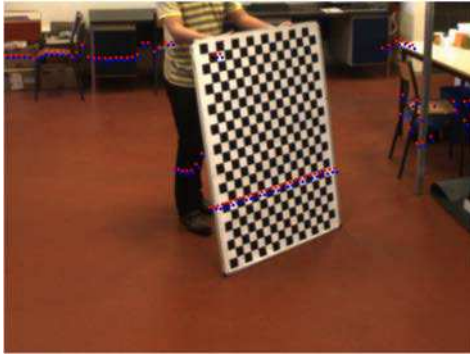
The calibration process is performed in an indoor environment at first, then, the results are verified in outdoor experiments. The calibration pattern we made is of  $19 \times 12$  squares with equal size  $50\text{mm} \times 50\text{mm}$ . The chessboard is detected by the two sensors meanwhile with 20 various poses. We both apply our method and Zhang's approach [Zhang 2004] for comparison. The projection of the lidar measurements into the images of the stereoscopic system is shown in Fig. 5.11. It seems that



(a) Projected lidar points in the left image by two methods



(b) Projected lidar points in the right image by two methods



(c) Partial enlargement view of the left image



(d) Partial enlargement view of the right image

Figure 5.11: Comparison of the proposed method with Zhang's method [Zhang 2004]. In (c) and (d), the blue points (by the proposed method) are almost at the same height, while the red points (by Zhang's method [Zhang 2004]) are not.

both of the results are quite reasonable. While from the partial enlargement views, our method gets more stable and precise results. Indeed, blue points (calculated by our method) in the pair of images have almost the same height, while the red ones (calculated by Zhang's method [Zhang 2004]) have obvious deviation between the two images. The estimated extrinsic parameters using our method and Zhang's method are shown in Tab. 5.1. It shows that our algorithm performs more precisely

than Zhang’s method [Zhang 2004]. The measure  $\varepsilon$  in Eq. 5.4 explains the deviation of red points in Fig. 5.11.

Table 5.1: Extrinsic Parameters and Precision Measure Calculated in Real Data

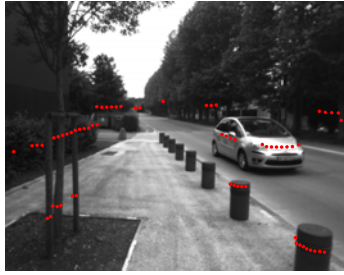
	The proposed method	Zhang’s method [Zhang 2004]
$\Phi_I$	[0.1474, −0.0251, 0.0083]	[0.1520, −0.0284, 0.0045]
$\Delta_I$	[0.1195, −1.4543, −1.2328]	[0.1245, −1.4566, −1.2297]
$\Phi_R$	[0.1509, −0.0210, 0.0067]	[0.1409, −0.0240, −0.0101]
$\Delta_R$	[0.3516, −1.4533, −1.2398]	[0.3637, −1.4583, −1.2427]
$\varepsilon$	0.5405mm	3.6mm

### 5.3.3 Outdoor Experiments

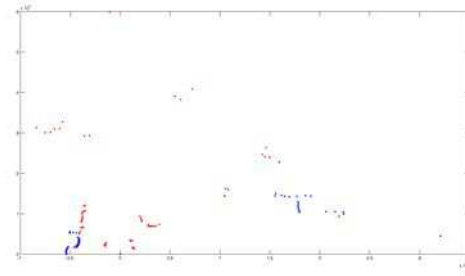
More experiments are used to test the calibration method in outdoor environments, as shown in Fig. 5.12 and Fig. 5.13. The experiments are performed in the city of Belfort, France. Left columns of Fig. 5.12 and 5.13 show six typical scenes in urban area captured by the left camera, as well as the corresponding lidar measurements within its view. Right columns of Fig. 5.12 and 5.13 show lidar measurements (the 2D lidar mounted on the bottom of our platform). The red points are within the view of the left camera, and then are projected into corresponding images as shown in the left column.

We also apply the same algorithm for the 2D lidar mounted on the top of our platform. Fig. 5.14 (b),(c) show lidar measurements from the bottom and top. Fig. 5.14 (a) shows the projections of lidar points into the left camera’s field of view by the output of the proposed algorithm. Another similar example is shown in Fig. 5.15. The lidar points within the camera’s field of view are marked by red color. We can see that the projections are quite reasonable, since the geometric characteristics of the scenes detected by the camera coincide closely with lidar data.

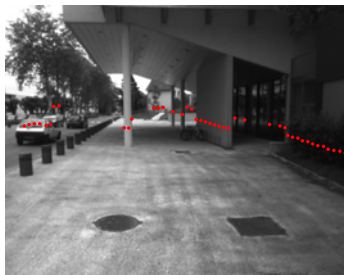




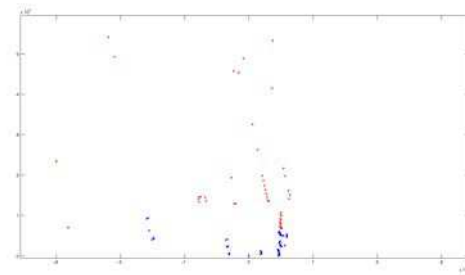
(a)



(b)



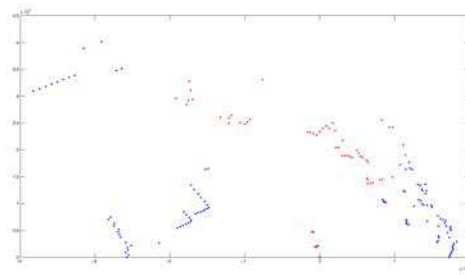
(c)



(d)



(e)

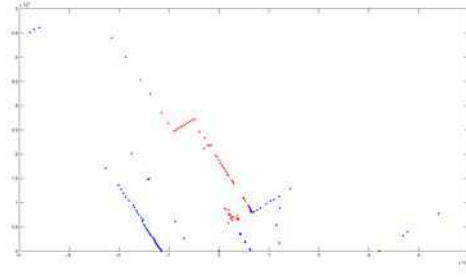


(f)

Figure 5.12: Experimental results in outdoor environments. *left* column shows the images and projecting lidar measurements, *right* column shows the corresponding lidar data, red points denote within the view of the left camera.



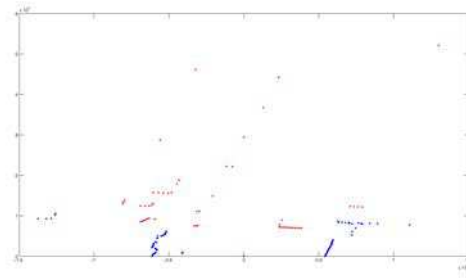
(a)



(b)



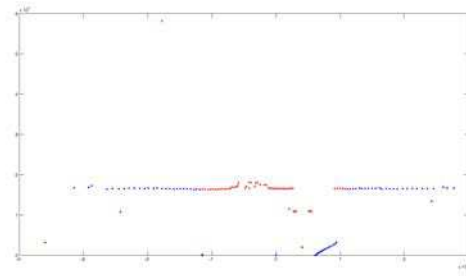
(c)



(d)



(e)

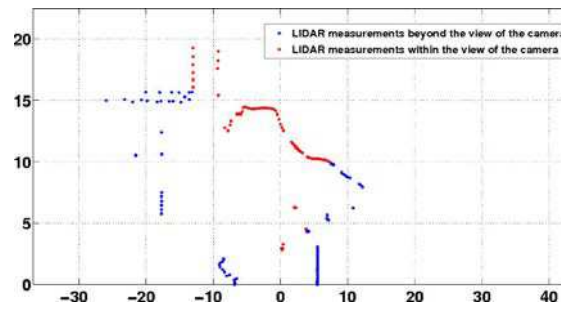


(f)

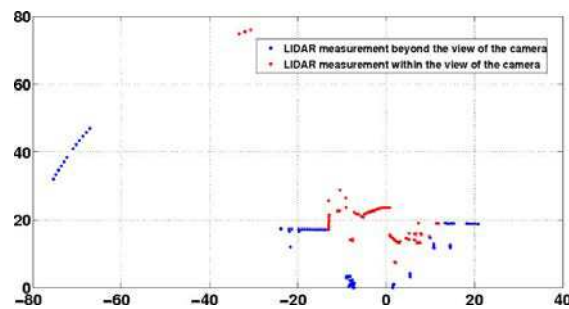
Figure 5.13: More Experimental results in outdoor environments. *left* column shows the images and projecting lidar measurements, *right* column shows the corresponding lidar data, red points denote within the view of the left camera.



(a) Scene 1: Projection of the lidar points into the left image

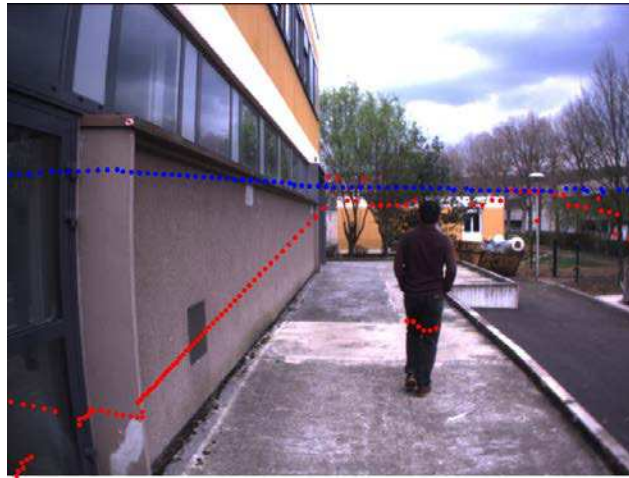


(b) Bottom lidar measurements in scene 1

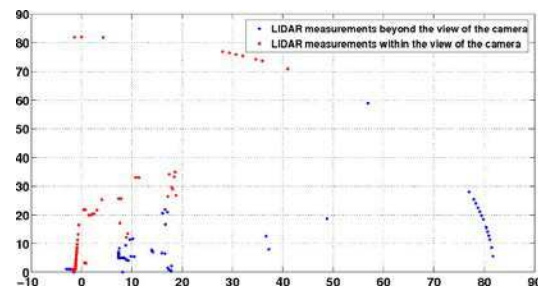


(c) Top lidar measurements in scene 1

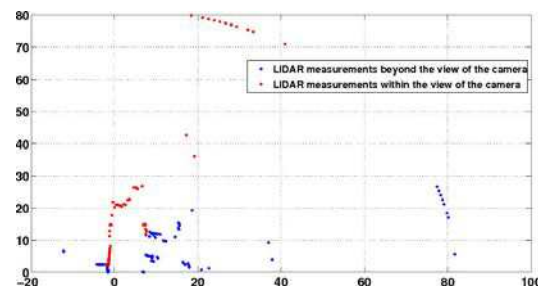
Figure 5.14: More experimental results in outdoor environments



(a) Scene 2: Projection of the lidar points into the left image



(b) Bottom lidar measurements in scene 2



(c) Top lidar measurements in scene 2

Figure 5.15: More experimental results in outdoor environments

## **5.4 Conclusion And Future Work**

In this chapter, we presented a novel extrinsic calibration method for integrating a lidar and a binocular stereoscopic system. Since the geometric relationship between the two cameras in the stereoscopic system is considered, real data experiments showed that our method is more exact and precise than Zhang's method which is widely used. Meanwhile, the introduction of sensor noise models and non-linear Mahalanobis distance optimization makes our method much more robust than Zhang's method. This is verified by computer simulation results. The described method can be applied for any multi-sensor fusion system consisting of multiple cameras and multiple lidars. Future work will be mainly focused on the calibration of a stereovision system with multi-layer lidars. Also, we plan to develop an improved self-calibration method.

# Occupancy Grid Mapping of Environments

---

## Contents

---

<b>6.1</b>	<b>Occupancy Grid Mapping by Stereo Vision . . . . .</b>	<b>120</b>
6.1.1	Introduction . . . . .	120
6.1.2	Foundations . . . . .	121
6.1.3	Ground Plane Analysis . . . . .	122
6.1.4	Building Occupancy Grid Map . . . . .	128
6.1.5	Experimental Results of Stereo Vision based Dynamic Occupancy Grid Mapping . . . . .	131
<b>6.2</b>	<b>Occupancy Grid Mapping by Lidar . . . . .</b>	<b>136</b>
6.2.1	Introduction . . . . .	136
6.2.2	Ray Casting . . . . .	136
6.2.3	Lidar Inverse Model . . . . .	137
6.2.4	Experimental Results . . . . .	139
<b>6.3</b>	<b>Occupancy Grid Mapping by Fusing Lidar and Stereo Vision</b>	<b>142</b>
6.3.1	Introduction . . . . .	142
6.3.2	Fusing by Linear Opinion Pool . . . . .	142
6.3.3	Experimental Results . . . . .	144
<b>6.4</b>	<b>Conclusion and Future Works . . . . .</b>	<b>148</b>

---

In the field of intelligent vehicles, many tasks, such as localization, collision avoidance, path planning, are usually performed based on well represented maps [Leonard 2008, Nguyen 2012]. Mapping is the process of transforming sensor measurement to an image of the environment. In the early days of robotic research, many tasks, such as localization/navigation, depend on the map being defined as a finite sized set of landmarks. However, physical sensors do not usually detect landmarks unambiguously. In order to use landmark based algorithms, sensor readings must be pre-processed in a separate step to convert the raw sensor data into a set of detected landmarks, such as in [Leonard 1991]. The additional step introduces more error into any algorithm, as well as discarding much of the sensor information which does not contain any landmark.

One of the primary drawbacks of landmark based maps is data association problem. Because raw sensor data are not labelled with correct landmarks detected, the

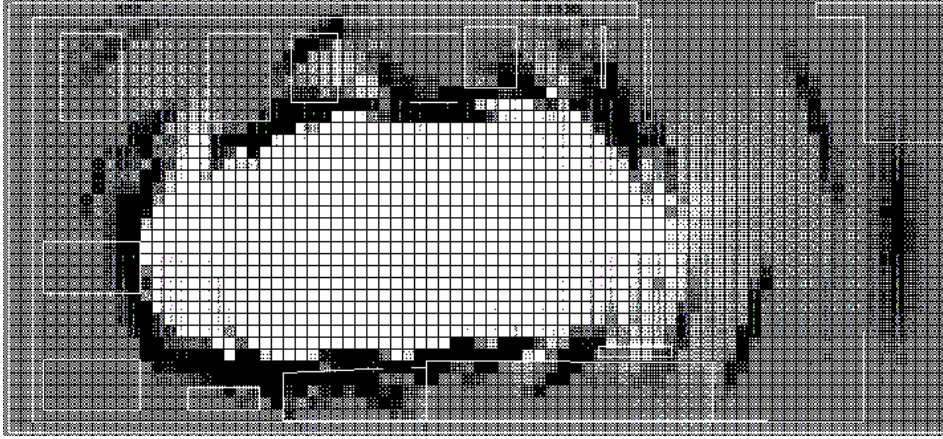


Figure 6.1: Example of occupancy grid map [Miroslav 1999]

sensor processing must somehow determine exactly which landmarks were observed. If mistakes are made, the localization and mapping algorithms, which depend on sensor data, will fail. In order to compensate the data association problem, many localization and SLAM (Simultaneous Localization And Mapping) algorithms include methods for determining associations between sensor data and landmarks. However, these techniques increase significantly the complexity of the solutions. Also, they do not solve the problem of actually finding landmarks in raw sensor readings. Some examples of these algorithms include GraphSLAM [Folkesson 2004] and Sparse Extended Information Filters [Thrun. 2004].

A common technique for map representation that does not suffer from data associations is occupancy grid map. In fact, occupancy grid map (OGM) [Moravec 1985, Thrun 2003] is one of the most popular environmental representation tool. It maps the environment around a vehicle as a field of uniformly distributed binary/ternary variables indicating status of grids (occupied, free or undetected). Its applications could be dated back to 1980s, when researchers utilized sonars or lidars to illustrate environments by occupancy grids. The first idea of employing occupancy grid map originated from the work of [Elfes 1989]. Fig. 6.1 shows an example of an occupancy grid map built by a sonar. In [Thrun 2005], occupancy grid map is regarded as the result of the mapping process and is exhaustively outlined. In addition, occupancy grid map provides an unified environmental representation for integrating different sensor measurements (radar, lidar, vision system).

An occupancy grid map  $\mathcal{M}$  is defined as an evenly distributed rectangular grids within a cover area  $\mathcal{F}$ . Each grid of the map is indexed and denoted as  $C_{i,j}$ . The sizes of all grids are equal and set as  $s_c$ . The occupancy probability of a cell  $C_{i,j}$  is determined by corresponding sensor measurements. In order to do this efficiently, we usually assume that the cells are independent, which greatly simplifies the mapping process. Hence, the occupancy probability is acquired as:

$$P_{i,j}(O|Y); Y = Y_1, \dots, Y_k \quad (6.1)$$

where  $Y$  denotes all the possible measurements that can be associated to cell  $C_{i,j}$ .

This chapter presents a framework to create occupancy grid maps in a dynamic environment, based on a stereo vision system and a single layer lidar. Furthermore, a fusion method to take advantages of the two sensors is presented. It has to be noted that, in previous researches [Moravec 1985, Thrun 2003], occupancy grid mapping is produced in a relatively fixed environment (such as rooms, buildings) for a mobile robot. After generating a global map, it is stored for afterward usage. Under these circumstances, it is reasonable to build a global map or perform SLAM, since the robot moves in a closed loop. Whereas the situation changes in the applications of intelligent vehicle. In our application, the intelligent vehicle has to self-drive successfully in a dynamic unknown urban area. Therefore, the vehicle faces an open, unknown and dynamic environment. It emphasizes the ability of mapping in dynamic environments in real time without prior information. In our works, we do not address SLAM but try to separate mapping and localization as two tasks. Hence, the occupancy grid mapping techniques that will be presented in this chapter are about building a current local map.

The chapter is organized as follows: Sec. 6.1 proposes an occupancy grid mapping method by stereo vision. In Sec. 6.2, we describe lidar based occupancy grid mapping method. At last, a fusion of lidar and stereo vision system to build occupancy grid map is presented in Sec. 6.3



## 6.1 Occupancy Grid Mapping by Stereo Vision

### 6.1.1 Introduction

In literature, range sensors, such as lidar and radar are usually used for creating occupancy grid maps. The characteristic of measuring distance directly makes occupancy grid mapping easy to be performed. Usually, under a given sensor measurement model (such as inverse sensor model [Thrun 2003]), probabilistic occupancy grid mapping is able to be quickly calculated with the measurements. While in our application, we also use stereo vision system to observe the surroundings, which offers abundant appearance information. Nevertheless, the different measurement characteristics of stereo vision system lead to a different processing approach to get occupancy grid map.

Several existing approaches are listed as follows. In [Murray 2000], the authors regard stereo sensor as a range finder, taking the first encountered object in each column as an occupied grid. [Brailon 2006] firstly estimates ground plane in front of a stereo-camera, then clusters the detected points above the plane as occupied grids. Three different types of occupancy grids are analyzed and compared at length in [Badino 2007], which furthermore proposes three kinds of occupancy likelihood functions modeled by Gaussian distribution. Quite similar to [Badino 2007], the method proposed in [Nguyen 2012] introduces an inverse sensor model for stereo-camera. In [Danesu 2009], occupancy grid map is generated from a digital elevation map after filtering out road and isle cells according to height information. In [Perrollaz 2010], the authors directly calculate occupancy grids by several effective probabilistic occupancy grid models in obstacle u-disparity image. In addition, this method requires a pre-performed road-obstacle separation.

However, many of the aforementioned papers do not treat the problem of moving objects in dynamic environments. Relied on the motion analysis of a moving intelligent vehicle (ego-motion estimation, moving object detection and recognition) described in Chapter. 3, 4, we propose a framework of stereo vision based dynamic occupancy grid mapping in urban environments. Dynamic occupancy grid map models real environments by evenly distributed rectangular grids, which contain both occupancy and motion information. The proposed framework mainly comprises two components (motion analysis for the vehicle-itself and independent moving objects, dynamic occupancy grid mapping) within two parallel process (sparse feature points processing between two consecutive stereo image pairs, dense stereo processing). Fig. 6.2 visualizes the whole framework. For every incoming stereo image pairs, sparse image feature points are extracted and tracked in a circular manner between the current and previous image pairs. The successfully tracked feature points are used to estimate ego-motion of the vehicle itself and independent motions of surrounding moving objects. Meanwhile, dense stereo disparity is calculated from each stereo image pair. A pixel-wise moving objects segmentation is performed in U-disparity map. Finally, the dense stereo information, together with the moving information, are used to create a dynamic occupancy grid map.

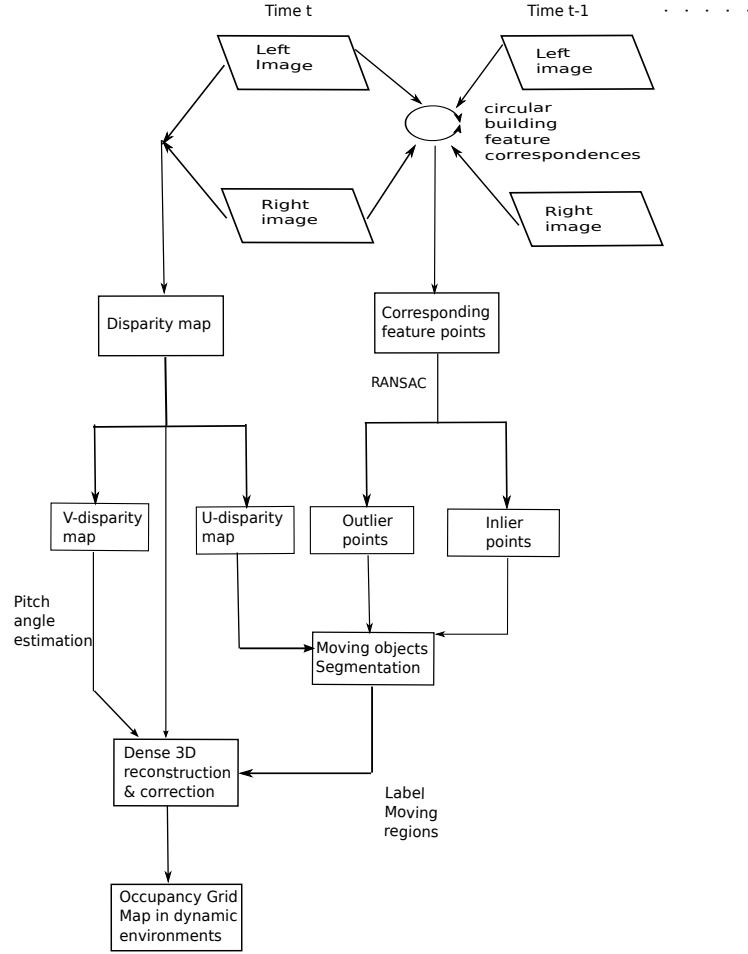


Figure 6.2: Flow diagram of the dynamic occupancy grid mapping

Since the motion analysis has been already described in previous chapters, in this section, we will emphasize the process of creating dynamic occupancy grid map based on stereo vision.

### 6.1.2 Foundations

The sensor used for drawing occupancy grid map is still the Bumblebee XB3 stereo vision system installed in our platform SetCar. The stereo vision model used here is slightly different to the model introduced in Sec. 2.1.1, as we also consider the pitch angle between the ground plane and the stereo vision system. The model is illustrated in Fig. 6.3. As usual, the stereo vision system is previously calibrated and rectified. Therefore, the left and right cameras are viewed as identical and modeled by classic pinhole model  $(f, c_u, c_v)$ , where  $f$  is the focal length,  $(c_u, c_v)$  is the position of principal point. The ground plane is assumed to be a flat plane under the stereo vision system, as drawn in Fig. 6.3. The stereoscopic coordinate system is assumed to be originated from  $\mathcal{O}_s$ , the middle point of the baseline. The

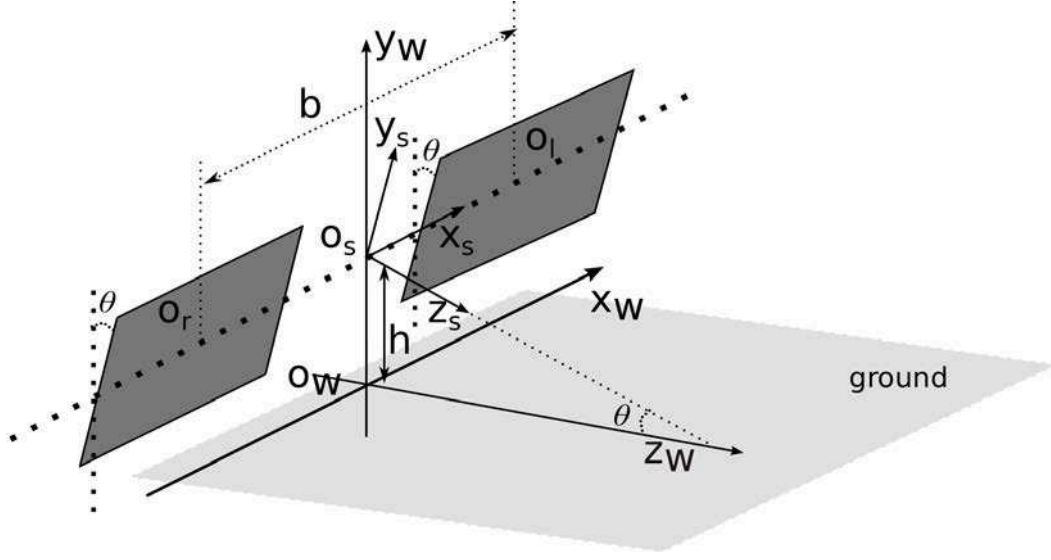


Figure 6.3: Geometric model of the stereo vision system

world coordinate system is set as originated from the point  $\mathcal{O}_w$ , the projection of  $\mathcal{O}_s$  in the ground. The left and right camera frames are assumed to have the same pitch angle  $\theta$  with regard to the ground plane. x-y-z directions are illustrated as well in Fig. 6.3. Therefore, 3D position of a point  $(X_s, Y_s, Z_s)$  in the stereoscopic coordinate system can be triangulated from its projections  $(u_l, v_l)$  and  $(u_r, v_r)$  in the left and right image planes as:

$$\begin{bmatrix} X_s \\ Y_s \\ Z_s \end{bmatrix} = \begin{bmatrix} (u_l - c_u) \cdot b/\Delta - b/2 \\ (v_l - c_v) \cdot b/\Delta \\ b \cdot f/\Delta \end{bmatrix} \quad (6.2)$$

where  $\Delta$  is the disparity. Considering the pitch angle between the stereoscopic system and the ground, the corrected coordinates in the world coordinate system are:

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X_s \\ Y_s \\ Z_s \end{bmatrix} + \begin{bmatrix} 0 \\ h \\ 0 \end{bmatrix} \quad (6.3)$$

where  $\theta$  is the pitch angle,  $h$  is the height between the stereoscopic system and the ground plane.

### 6.1.3 Ground Plane Analysis

In urban environment, ground plane (road, parking field, etc.) can be regarded as a supporting plane to obstacles. Usually, an obstacle roots from ground plane and is perpendicular to the plane. To successfully mapping the obstacles in surrounding environment, analyzing the ground plane is crucial. The first component of ground plane analysis is the pitch angle estimation.

### 6.1.3.1 Estimating Pitch Angle between Ground Plane and Stereoscopic System

In most of the existing vision-based occupancy grid mapping methods (e.g. [Badino 2007], [Nguyen 2012], [Perrollaz 2010]) the stereoscopic system is assumed to be parallel to the ground. However, this assumption does not hold true in our practice, since in our platform, the stereo vision system is mounted with a pitch angle to the ground. The pitch angle is crucial for mapping the surrounding environment. The importance of estimating the pitch angle between the stereo vision system and the ground plane is illustrated in Fig. 6.4. Since the reconstructed 3D

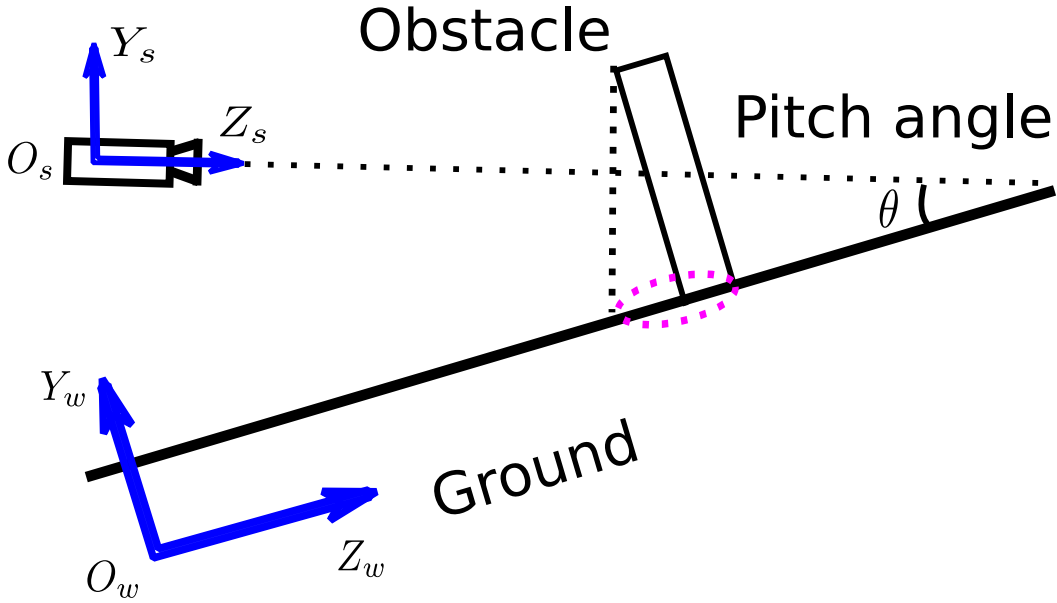


Figure 6.4: The importance of estimating pitch angle

points are in the coordinate of stereo vision system  $O_s$ , assume there is an obstacle perpendicular to the ground, its projection on the ground plane is like a shadow (drawn in purple ellipse) caused by the pitch angle. Therefore, to make an accurate grid map of the environment, the pitch angle has to be computed and compensated.

**V-disparity based Pitch Angle Estimation:** As we mentioned in Sec. 4.1.2.1, one attribute of V-disparity is that, if the ground is a planar plane, it would be projected as a line, as drawn in Fig. 6.5. Furthermore, the pitch angle could be computed directly from V-disparity map. Here, we show the derivation of this property. According to the model in Fig. 6.3 and Eq. 6.3, given a 3D point in the world coordinate system  $O_w$  with homogeneous coordinates  $\tilde{\mathbf{P}} = (X, Y, Z, 1)^T$  and

its image coordinates  $(u, v, 1)^T$  in the left image, we have:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{M}_l \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (6.4)$$

where  $\lambda$  is a scale factor,  $\mathbf{M}_l$  is the transformation matrix formed by 3D rigid transformation and intrinsic projection matrix:

$$\mathbf{M}_l = \begin{bmatrix} f & c_u \sin \theta & c_u \cos \theta & b/2 \\ 0 & f \cos \theta + c_v \sin \theta & -f \sin \theta + c_v \cos \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \end{bmatrix} \quad (6.5)$$

From Eq. 6.4 and 6.5, we can have:

$$v = c_v + f \frac{(Y + h) \cos \theta - Z \sin \theta}{(Y + h) \sin \theta + Z \cos \theta} \quad (6.6)$$

and

$$\Delta = |u_l - u_r| = \frac{fb}{(Y + h) \sin \theta + Z \cos \theta} \quad (6.7)$$

Assume the ground around the vehicle is flat as drawn in Fig. 6.3, then its equation in  $\mathcal{O}_w$  is  $Y = 0$ . Combining the ground plane equation with Eq. 6.6 and Eq. 6.7, we can deduce the following linear equation with respect to the left image:

$$\frac{h}{b} \Delta = f \sin \theta + (v - c_v) \cos \theta \quad (6.8)$$

Equation 6.8 shows that a horizontal ground plane in  $\mathcal{O}_w$  will be projected as a straight line in V-disparity image  $I(\Delta, v)$ . Considering the intercept  $v_c$  in  $v$ -axis when  $\Delta = 0$ , we have:

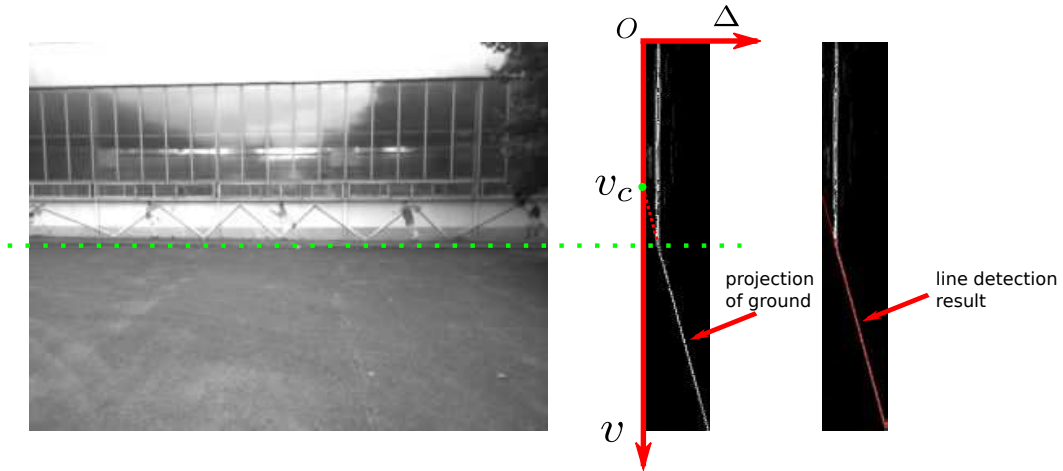


Figure 6.5: Estimating pitch angle to the ground

$$0 = f \sin \theta + (v_c - c_v) \cos \theta \quad (6.9)$$

Therefore, for a planar ground, the pitch angle can be deduced as:

$$\theta = \arctan\left(\frac{c_v - v_c}{f}\right) \quad (6.10)$$

The estimation of pitch angle is illustrated in Fig. 6.5. It requires to find the line generated by the ground plane at first, and then, compute the intercept of this line to  $v$ -axis. Then, following Eq. 6.10 gets the pitch angle estimation.

In real environments, the ground plane is not ideally flat and the property of "line like" projection is hard to maintain. Hence, we use a poly-line model which consists of two-line segments to approximate real ground projection. In this poly-line model, two lines  $(l_1, l_2)$  correspond to two segments of ground plane according to its distance to the stereoscopic system. For example, we use one line to represent the ground within 10 meters in front of the stereoscopic system, and the other line is used to model the ground plane more than 10 meters away. We use Hough transform<sup>1</sup> to extract the poly-line in V-disparity image. An example is shown in Fig. 6.6. The pitch angles of the two section of ground,  $\theta_1, \theta_2$  are estimated separately according to Eq. 6.10.

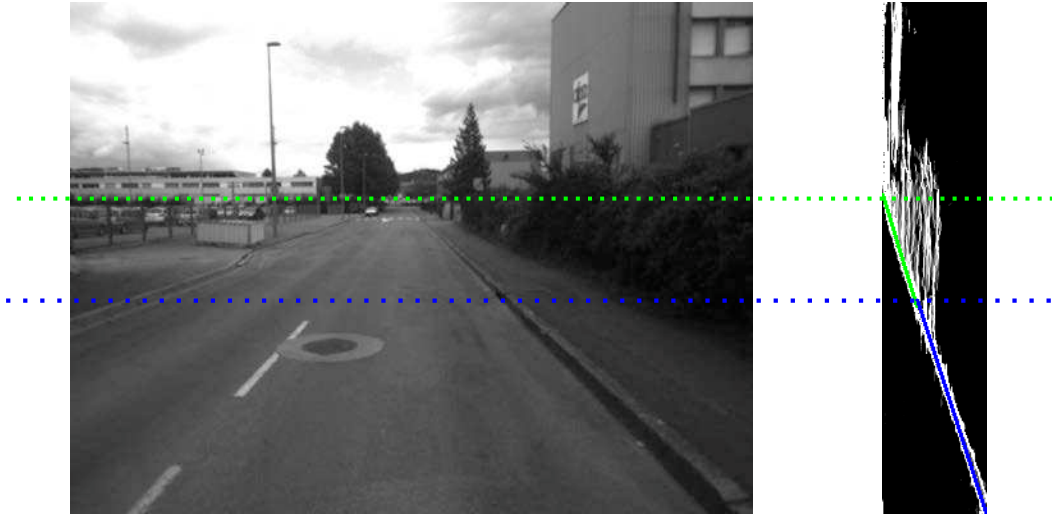


Figure 6.6: Using two line segments to approximate ground projection in real environment. The green line represents ground projection more than 10 meters away. The blue line is the projection of ground within 10 meters.

**Sequentially estimate pitch angles:** Using two-line segments is able to estimate the pitch angles of ground sections from one stereo image pair. While in our application, the occupancy grid map is sequentially computed when the vehicle is driving in real urban area. Many factors (e.g. vibration of vehicle, ascent or downhill of the road) would contribute to the sudden changes of pitch angles. We use Kalman

<sup>1</sup>[http://en.wikipedia.org/wiki/Hough\\_transform](http://en.wikipedia.org/wiki/Hough_transform)

filter [Kalman 1960] to filter out the noise and converge toward to the true values. In our case, we assume the pitch angles  $\theta_1$  and  $\theta_2$  as constant variables polluted by Gaussian noise and a state space model is used. Taking  $\theta_1$  for example, its state transition model is:

$$\theta_{k,1} = \theta_{k-1,1} + w_{k,1} \quad (6.11)$$

where  $\theta_{k,1}$  is the current pitch angle  $\theta_1$  at time  $k$ ,  $\theta_{k-1,1}$  the state at the last time,  $w_{k,1}$  is a zero mean Gaussian noise with variance  $\sigma_{w_{k,1}}$ . The measurement model is:

$$z_{k,1} = \theta_{k,1} + v_{k,1} \quad (6.12)$$

where  $z_{k,1}$  is the measurement value and  $v_{k,1}$  is a zero mean Gaussian noise with variance  $\sigma_{v_{k,1}}$ . Kalman filter [Kalman 1960] is able to optimally estimate variables' posteriors in a linear dynamical system and all the error terms and measurements have a Gaussian distribution. It works in a two-step process. In the *prediction step*, the Kalman filter predicts estimates of the current state, along with their uncertainties. When the next measurement is observed, these predictions are updated by weighted averaging in the *update step*, as:

$$\hat{\theta}_k = \hat{\theta}_k^- + W(z_k - \hat{\theta}_k^-) \quad (6.13)$$

where  $\hat{\theta}_k$  is the posterior estimation of  $\theta_k$ ,  $\hat{\theta}_k^-$  is the prediction from state transition model.  $z_k - \hat{\theta}_k^-$  is the difference between measured and predicted values,  $W$  is a weight factor. The essence of Kalman filter is a framework of recursively estimating  $W$  from the uncertainties of all the errors. The principle is to give more weight to estimates with higher certainty. More details about Kalman filter can be found in [Welch 1995].

An example of pitch angle filtering is shown in Fig. 6.7, with an experimental result for a video sequence with almost 700 frames. The blue line is the original estimation of pitch angle  $\theta_1$ , red line is the result after Kalman filtering. The initial variances of  $w_{0,1}$  and  $v_{0,1}$  are set as 0.0005 and 0.001, respectively. The Kalman filter configuration for  $\theta_2$  is the same as for  $\theta_1$ . It is clear that after Kalman filtering, the pitch angle estimation is more smooth than before.

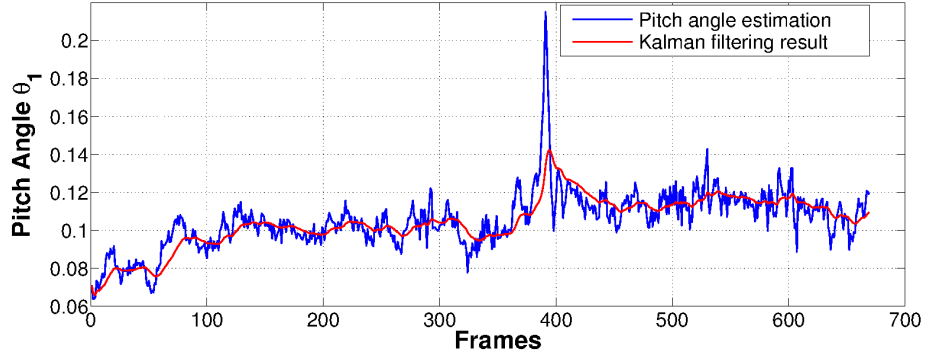
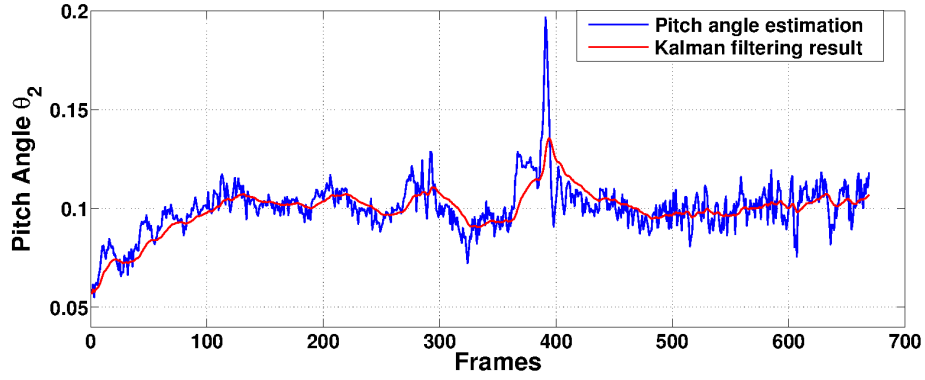
(a) Kalman filtering results for  $\theta_1$ (b) Kalman filtering results for  $\theta_2$ 

Figure 6.7: Kalman filtering pitch angles

### 6.1.3.2 Ground Subtraction

Ground plane is a free space in an environment. To better map obstacles, ground areas are detected and subtracted before mapping. Detecting and subtracting ground areas are based on the fitted poly-line in V-disparity map as in Sec. 6.1.3.1. As drawn in Fig. 6.8, the area below the poly-line represents ground field. Given a pixel  $p(\Delta, v)$  in V-disparity image, we compute its v-distance  $d_v$  to the estimated poly-line as  $d_v = v - v_g$ , where  $v_g$  is its projection in the poly-line. The classification of ground and non-ground is judged by:

$$p(\Delta, v) \in \begin{cases} \text{ground} & \text{if } d_v > d_h \\ \text{non-ground} & \text{if } d_v \leq d_h \end{cases} \quad (6.14)$$

where  $d_h$  is a threshold that restricts the ground region in V-disparity image. An example of the defined ground region is shown in Fig. 6.8. The area under the red poly-line in Fig. 6.8 (a) represents the ground region. Fig. 6.8 (b) shows the result of ground subtraction. In the process of computing occupancy probability, grids from the ground segments are directly labelled as free areas.



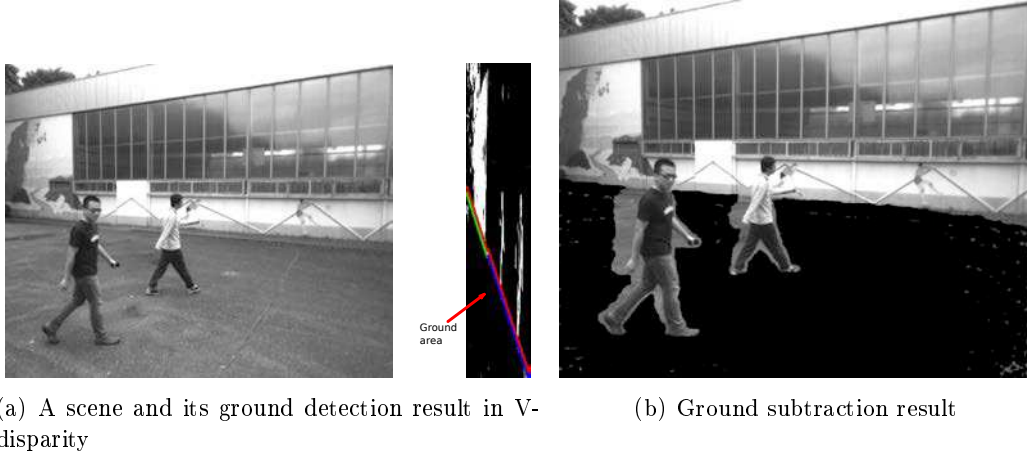


Figure 6.8: Example of ground subtraction ( $d_h = -10$ )

#### 6.1.4 Building Occupancy Grid Map

In this section, we will describe how to build an occupancy grid map based on stereo vision system. An occupancy grid map  $\mathcal{M}$  is defined as an evenly distributed rectangular grids  $C_{i,j}$  array within a predefined area  $\mathcal{F}$ , where  $i$  and  $j$  are indices of a grid in the map. The sizes of all grids are equal and set to  $s_c$ . Every grid  $C_{i,j}$  holds a ternary occupancy indicator  $O_{i,j}$  for three states:

$$O_{i,j} = \begin{cases} \text{undetected} \\ \text{occupied} \\ \text{free} \end{cases} \quad (6.15)$$

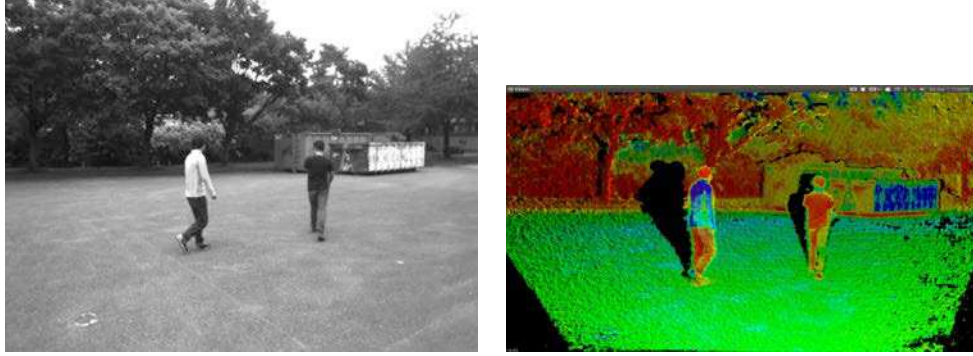
The process of building occupancy grid map is to estimate the status of the occupancy indicator from stereo measures. The probability of a cell  $C_{i,j}$  to be occupied given stereo observations  $Y_{stereo}$  is defined as  $P_{i,j}(O|Y_{stereo})$ . In the following, an approach of computing  $P_{i,j}(O|Y_{stereo})$  is given.

##### 6.1.4.1 Reconstruct and Correct 3D Points

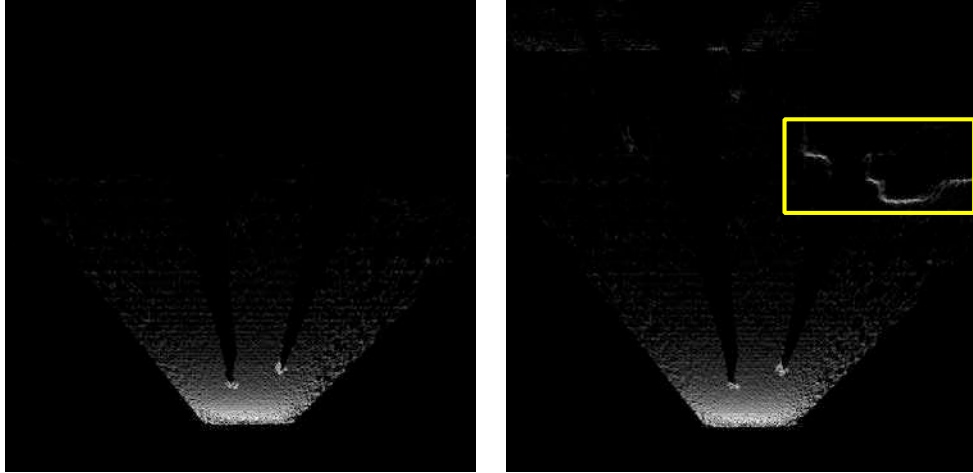
Before computing an occupancy grid map from stereo vision, a dense disparity map is calculated by Semi-Global Block Matching (SGBM) algorithm. The left pixels after ground subtraction are reconstructed by triangulation (Eq. 2.23). The obtained 3D points are corrected by the pitch angles estimated previously, according to Eq. 6.3. The corrected 3D points within a region of interest (ROI) are used for computing occupancy grid map.

The benefit of pitch angle correction is illustrated in Fig. 6.9 (we show the ground for further demonstrative purpose). Fig. 6.9 (a) - (b) show a 3D reconstruction example from a stereo image pair. The projection of all the 3D points in the ground plane without and with pitch angle correction is shown in Fig. 6.9

(c) (d), respectively. The intensity of a pixel in (c) or (d) denotes the number of points in one grid. If a grid has more points, it has higher gray value. It is clear to see that the regions belonging to the obstacles (in yellow box) become much more "brighter" after correction. This is because after correction, the reconstructed 3D points located on the surface of the obstacle become more vertical to the ground and hence more concentrated in one area.



(a) A scene consisting of two pedestrians and obstacles (b) 3D reconstruction from disparity image



(c) Projection of all the 3D points to the ground before pitch angle correction (d) Projection of all the 3D points to the ground after pitch angle correction

Figure 6.9: Effect of pitch angle correction

#### 6.1.4.2 Computing Occupancy Probability

The reconstructed and corrected 3D points within ROI are assigned to corresponding grids in map  $\mathcal{M}$  with respect to their positions. In each grid  $C_{i,j}$ , the number of assigned points  $n_{i,j}$  is counted.

Assuming that all obstacles are perpendicular to the planar ground, the more the number of points a grid holds, the bigger the probability it is occupied. Conse-

quently, the occupancy probabilities  $P_{i,j}(O|Y_{stereo})$  are closely related to the number of associated 3D points in each grid.

However, the distribution of the associated number of 3D points is not uniform in the field of view: the places more close to the stereoscopic system has more stereo measurements. As shown in Fig. 6.9, the ground near the stereoscopic system usually contains more associated 3D points than a real obstacle far away from the stereoscopic system. Estimating occupancy probability directly from absolute number of points would always lead to false decisions. In literature, [Badino 2007] and [Nguyen 2012] don't mention this problem, while [Perrollaz 2010] and [Danescu 2009] avoid it by a previous separation of road pixels. In our method, this problem is solved by sigmoid function based adjustment described in Sec. 4.1. The absolute number of points for grid  $C_{i,j}$  is adjusted as:

$$n'_{i,j} = n_{i,j} \cdot \mathcal{S}(d_{i,j}) = n_{i,j} \cdot \frac{r}{1 + e^{d_{i,j} \cdot c}} \quad (6.16)$$

where  $n_{i,j}$  and  $n'_{i,j}$  are the absolute and adjusted numbers of points in grid  $C_{i,j}$  respectively.  $d_{i,j}$  is the distance from stereo vision system to the grid.  $r$  and  $c$  are control coefficients. The occupancy probability with respect to the number of points is modeled as:

$$P_{i,j}(O|Y_{stereo}) = 1 - e^{-(n'_{i,j}/\delta_n)} \quad (6.17)$$

where  $\delta_n$  is a scale factor. However, it is not convenient to directly use the probability in decision making. The log-odds of the probability is then adopted:

$$l_{i,j}(O|Y_{stereo}) = \log\left(\frac{P_{i,j}(O|Y_{stereo})}{1 - P_{i,j}(O|Y_{stereo})}\right) \quad (6.18)$$

Based on the log-odds of each grid  $C_{i,j}$ , the occupancy indicator  $O_{i,j}$  is decided as:

$$O_{i,j} = \begin{cases} undetected & \text{if } n'_{i,j} < n_t \\ occupied & \text{if } l_{i,j}(O) \geq l_t \\ free & \text{if } l_{i,j}(O) < l_t \text{ or } O_{i,j} \text{ is located in the ground segments} \end{cases} \quad (6.19)$$

where  $n_t$  and  $l_t$  are thresholds manually set for making decision.

#### 6.1.4.3 Moving Object Detection and Recognition in Occupancy Grid Map

In chapter 4, a framework of ego-motion analysis, moving object detection, segmentation and recognition is presented. With the processing results of moving object detection and recognition, the occupancy grid map is enriched with more information. For a grid  $C_{i,j}$  in grid map, a state vector replaces the former occupancy indicator:  $S_{i,j} = (O_{i,j}, M_{i,j}, T_{i,j})$ , where  $M_{i,j}$  is a binary variable indicating whether the grid is moving or not,  $T_{i,j}$  represents the three possible types of moving objects,

i.e. pedestrian, vehicle and Motor/Bike. The values of  $M_{i,j}$  and  $T_{i,j}$  are mainly decided by the results in chapter 4.

To label the independent grids in occupancy grid map, we distinguish the reconstructed 3D points from static environment and segmented moving objects. The 3D points from static scenes are accumulated in the grid as before and their number is represented by  $n_{i,j}^s$ . Similarly,  $n_{i,j}^d$  is the number of points from dynamic objects, we have:

$$n_{i,j} = n_{i,j}^s + n_{i,j}^d \quad (6.20)$$

The motion indicator  $M_{i,j}$  is decided by comparing  $n_{i,j}^d$  with  $n_{i,j}^s$ .

$$M_{i,j} = \begin{cases} \text{dynamic} & \text{if } n_{i,j}^d > n_{i,j}^s \\ \text{static} & \text{otherwise} \end{cases} \quad (6.21)$$

The type indicator  $T_{i,j}$  is directly set by the recognition results from Sec. 4.2.

### 6.1.5 Experimental Results of Stereo Vision based Dynamic Occupancy Grid Mapping

The proposed stereo vision based dynamic occupancy grid mapping is tested using the dataset acquired by our experimental vehicle SetCar introduced in Sec. 1.3. The whole method is implemented in C++ and performed in a desktop with a CPU Intel i7-3770 quad cores 3.40GHZ. The region of interest (ROI) for the grid map is set to  $30m \times 30m$ , with a maximum height of  $3m$ . The parameters used to calculate the occupancy indicator are set as  $\delta_n = 0.2, n_t = 2, l_t = 7, r = 8, c = 0.02$ . The whole computation time including motion analysis and dynamic occupancy grid mapping is 0.5s in average for each image pair. Four sequential results are shown in Fig. 6.10, 6.11, 6.12 and 6.13. In the figures, red areas represent detected independent moving objects, and the recognized categories are labelled outside the bounding box of the moving objects.

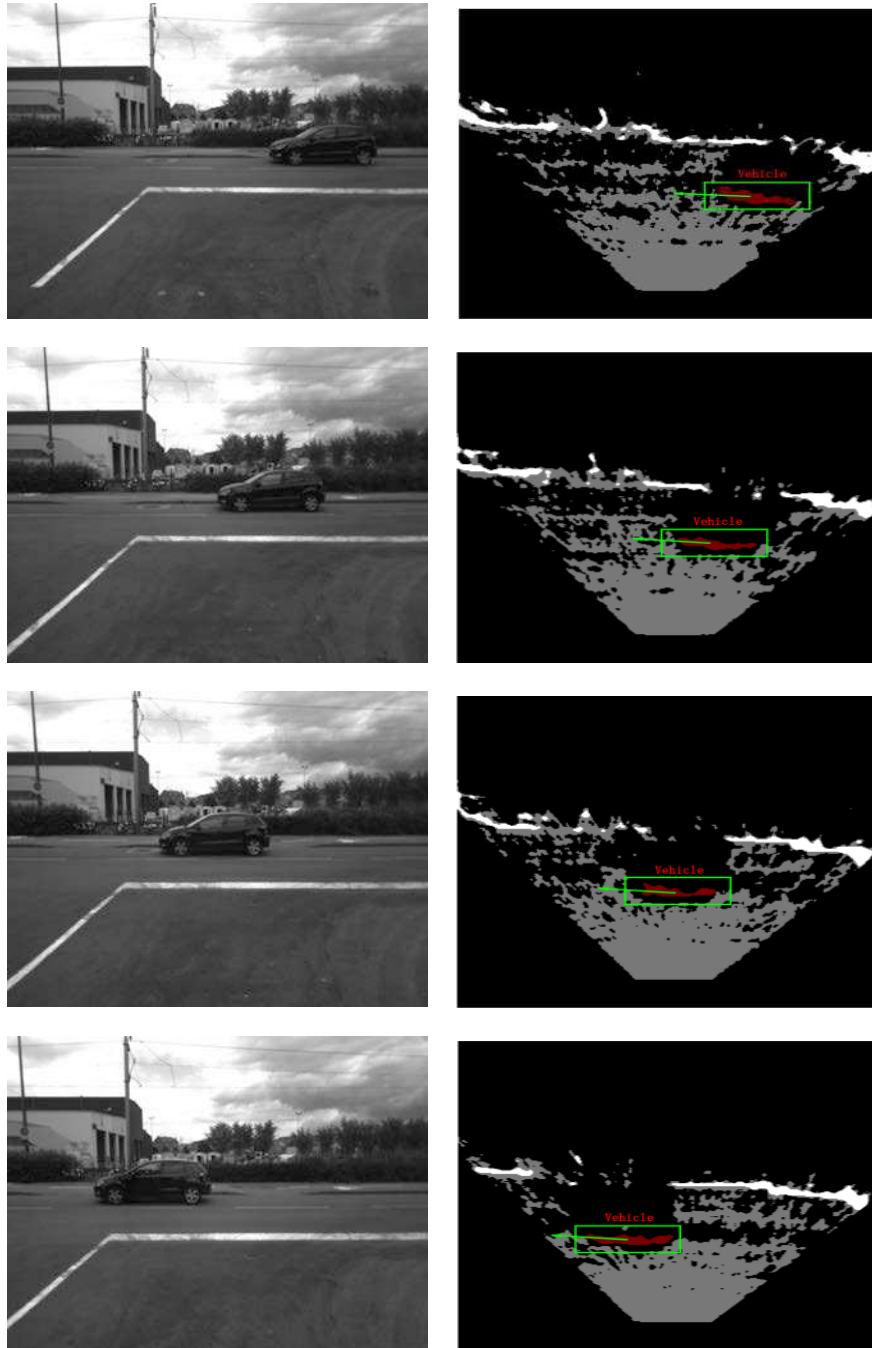


Figure 6.10: Experimental results of dynamic occupancy grid map: independent moving objects (red), static occluded areas (white), free areas (gray), undetected areas (black).

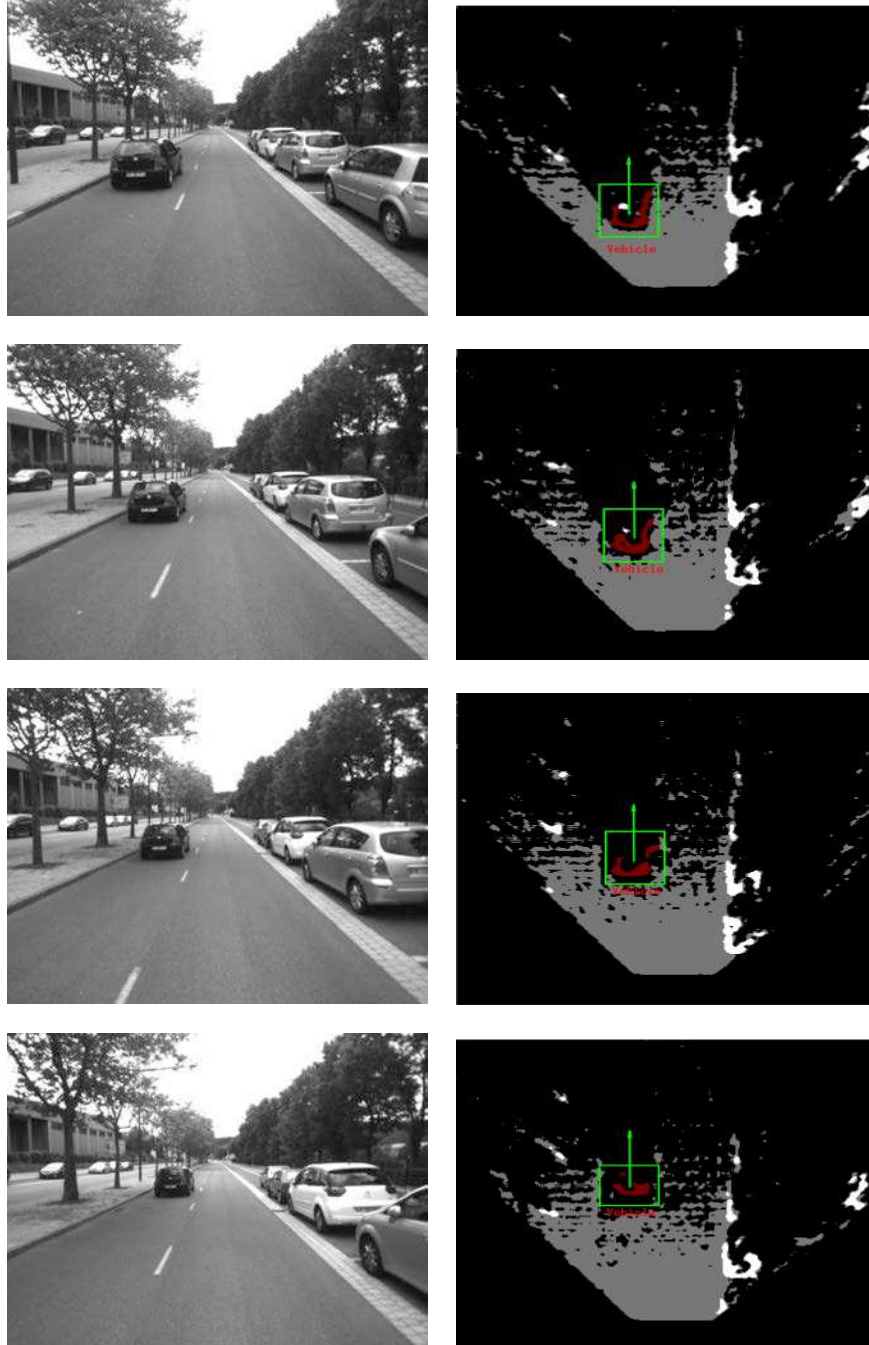


Figure 6.11: Experimental results of dynamic occupancy grid map: independent moving objects (red), static occluded areas (white), free areas (gray), undetected areas (black).

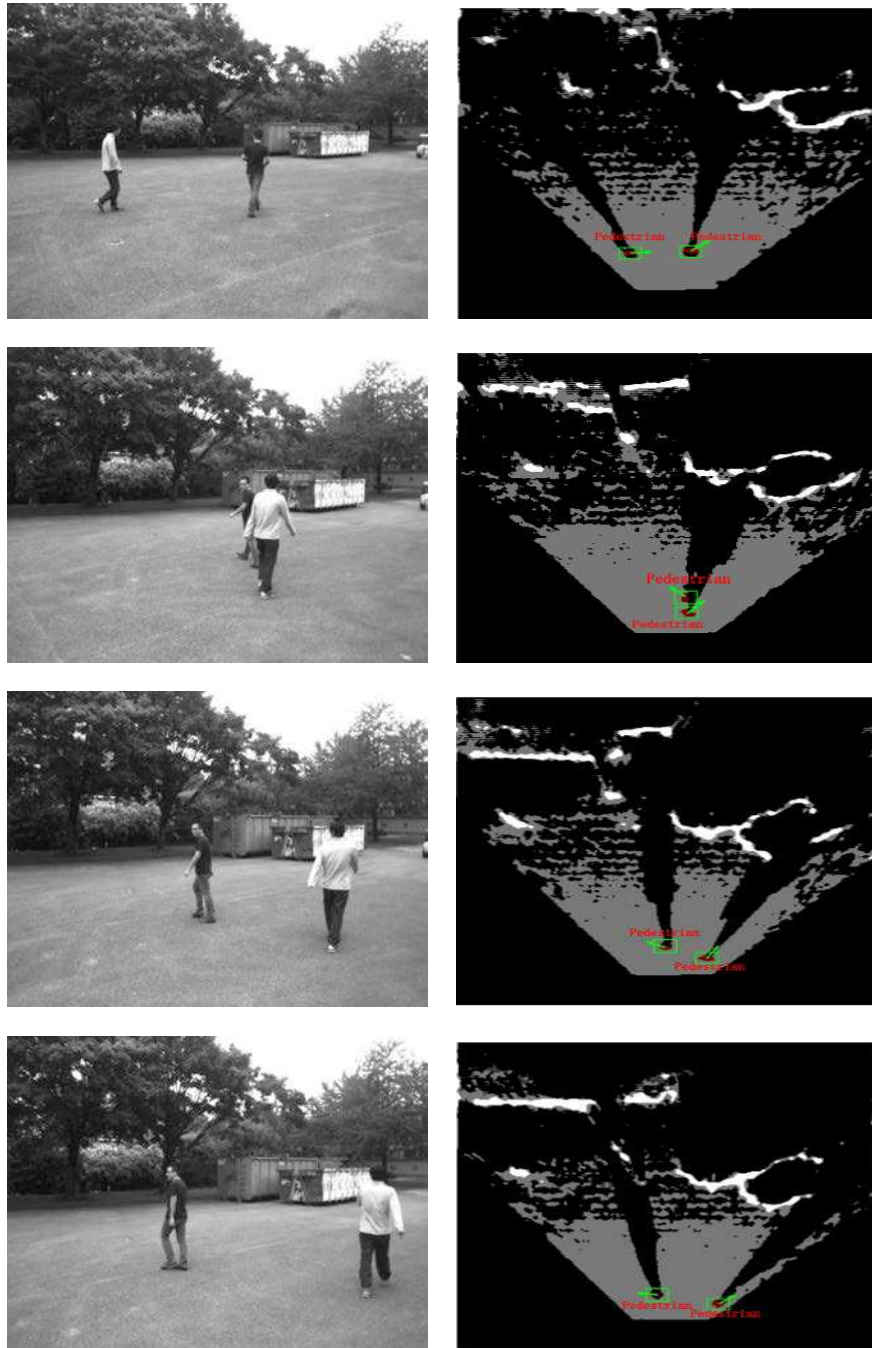


Figure 6.12: Experimental results of dynamic occupancy grid map: independent moving objects (red), static occluded areas (white), free areas (gray), undetected areas (black).



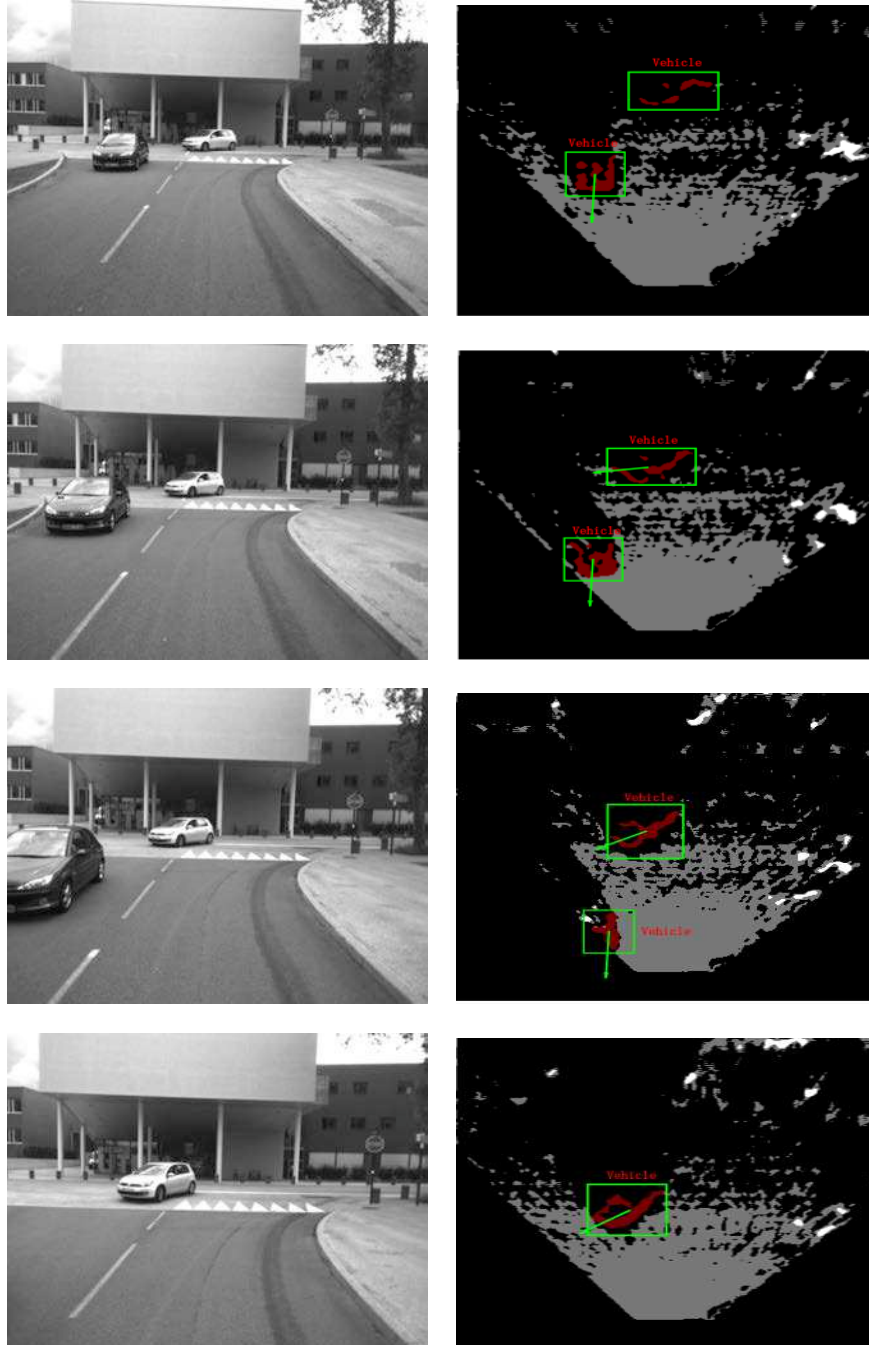


Figure 6.13: Experimental results of dynamic occupancy grid map: independent moving objects (red), static occluded areas (white), free areas (gray), undetected areas (black).



## 6.2 Occupancy Grid Mapping by Lidar

### 6.2.1 Introduction

Lidar provides an efficient approach to detect distances to the nearest obstacles. According to a certain measuring model of the sensor (e.g. inverse model [Moravec 1985], forward model [Thrun 2003], Gaussian Beam Process [Plagemann 2007]), the occupancy probability of each cell could be quickly calculated and updated. Three common sensor models for lidar are list as follows:

- Inverse sensor model: For lidar measurements  $Y_{lidar}$  in a cell  $C_{i,j}$ , the probability  $P_{i,j}(O|Y_{lidar})$  specifies the probability of occupancy of the grid cell  $C_{i,j}$  conditioned on the measurement  $Y_{lidar}$ . This probability constitutes an *inverse sensor model*, since it maps sensor measurements back to its causes. Occupancy grid maps usually rely on such inverse models. Notice that the inverse model does not take the occupancy of neighboring cells into account.
- Forward sensor model: Forward models proposed in [Thrun 2003] tries to overcome the assumption of inverse sensor model. It is able to calculate the likelihood of the sensor measurements for each map and set of poses, in a way that it considers all inter-cell dependencies. A forward model is of the form:  $P(Y_{lidar}|\mathcal{M})$ , which specifies a probability distribution over sensor measurements  $Y_{lidar}$  given a map  $\mathcal{M}$ . With forward model, mapping is transformed to an optimization problem, which aims to find the map that maximizes data likelihood.
- Gaussian beam process: To overcome the inherent sparsity of lidar measurements, [Plagemann 2007] presented a so-called Gaussian beam process, which treats the measurement modeling task as a nonparametric Bayesian regression problem and solves it using Gaussian processes. The major benefit of this approach is its ability to generalize over entire range scans directly. We can learn the distributions of range measurements for whole regions of the robot's configuration space from only few recorded or simulated range scans.

In our work, we apply the basic inverse model of lidar in occupancy grid mapping.

### 6.2.2 Ray Casting

As we described in Sec. 2.1.2, raw lidar measurements consist of detected ranges and scan angles in polar coordinate system. They can be quickly transformed into Cartesian coordinate system according to Eq. 2.1. It is reasonable to assume that every cell that lies along a line from the ego position to the detected point is free (within the maximum detected range). Before achieving this, cells traversed by a laser ray have to be marked out. We use Bresenham algorithm [Bresenham 1965] to cast rays into the grids of the map.

The Bresenham algorithm is an algorithm which determines in which order one should form a close approximation to a straight line between two given points. It

is commonly used to draw lines on a computer screen or raster grid. It only uses integer addition, subtraction and bit shifting, all of which are very cheap operations in standard computer architectures. Hence, its implementation is very simple and fast<sup>2</sup>. The start point for the Bresenham algorithm is, in our case, always the ego car position. The endpoint is the occupied cell on that line. By doing this, we can induce all free cells corresponding to occupied regions. Areas that are occluded by occupied cells are classified as unknown. An example is shown in Fig. 6.14. The gray cells between the start point and the occluded cells are all the traversed grids by the laser rays.

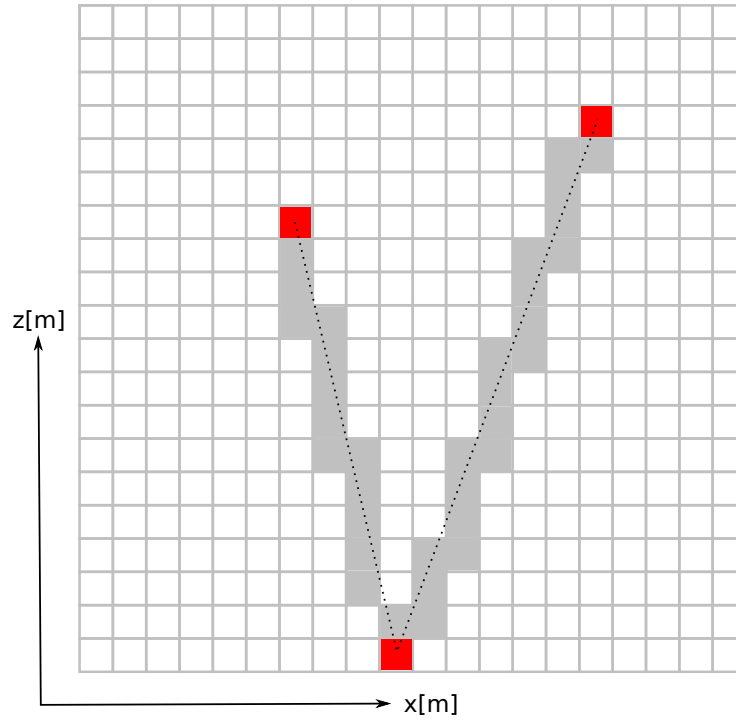


Figure 6.14: Ray casting using Bresenham algorithm

### 6.2.3 Lidar Inverse Model

As we described previously, the probability  $P_{i,j}(O|Y_{lidar})$  represents the inverse sensor model of a 2D lidar. The objective of the inverse sensor model is to describe the formation process by which measurements are generated in a physical world and how they affect the grid cells. A lidar actively emits a signal and records its echo. The returned echo depends on a variety of properties, such as the distance to the target, the surface material and the angle between the surface normal and the beam.

A complex lidar measuring model concerning both range and angular errors is introduced in Sec. 2.1.2, for an off-line calibration problem. However, in order to accelerate the mapping process, we simplify the measuring model by only considering

<sup>2</sup>[http://rosettacode.org/wiki/Bitmap/Bresenham's\\_line\\_algorithm](http://rosettacode.org/wiki/Bitmap/Bresenham's_line_algorithm)

the range error. Let  $Y_{lidar}^k = (r^k, \theta^k)$  represent a measurement of the  $k$ th laser beam. For a cell  $C_{i,j}$  in the grid map, its distance to  $Y_{lidar}^k$  is  $d(C_{i,j}, Y_{lidar}^k)$ . Then, the inverse sensor model is:

$$P_{i,j}(O|Y_{lidar}^k) = \lambda \exp(-\frac{1}{2}d(C_{i,j}, Y_{lidar}^k)^T \Sigma(r^k)^{-1}d(C_{i,j}, Y_{lidar}^k)) \quad (6.22)$$

where  $\lambda$  is a scale factor. The parameter  $\Sigma(r^k)$  is the covariance matrix decided by  $r^k$ :

$$\Sigma(r^k) = \begin{bmatrix} f(r^k) & 0 \\ 0 & f(r^k) \end{bmatrix} \quad (6.23)$$

$f(r^k)$  reflects the amount of affected regions with respect to the distance measured and is usually proportional to  $r^k$ . An illustrative example of the inverse model is shown in Fig. 6.15. The radius of a point reveals its influenced areas. Along with the increase of distance, the uncertainty of measurement grows. Hence, the influenced areas enlarge with the distance.

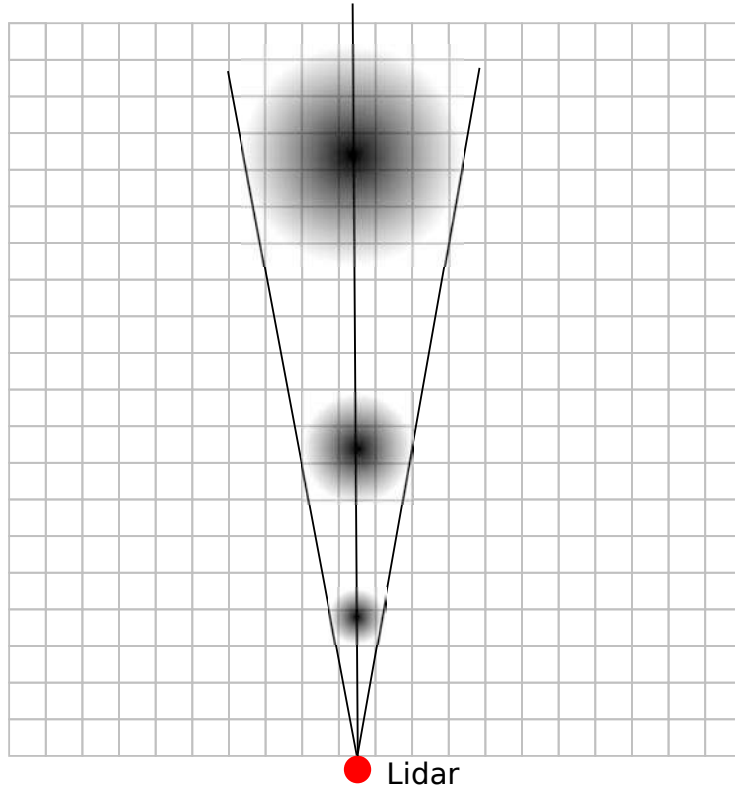


Figure 6.15: Inverse model of the lidar

As mentioned before, occupancy grids not only provide information about occupied space, but also about free and unknown space. Therefore, it is necessary to define the cells that are considered to be free or unknown given the measurement  $Y_{lidar}^k$ . Cells that are in front of the measurement should have a substantially lower

occupancy probability. Inversely, cells behinds the measurement should be unknown more likely. Therefore, the finally occupancy probability is as follows:

$$P_{i,j}(O|Y_{lidar}^k) = \begin{cases} P_{i,j}(O|Y_{lidar}^k) \text{ in Eq.6.15} & \text{if } C_{i,j} \text{ is in front of } Y_{lidar}^k \\ \max(0.5, P_{i,j}(O|Y_{lidar}^k)) & \text{if } C_{i,j} \text{ is behind } Y_{lidar}^k \end{cases} \quad (6.24)$$

An illustrative example of the inverse model is shown in:

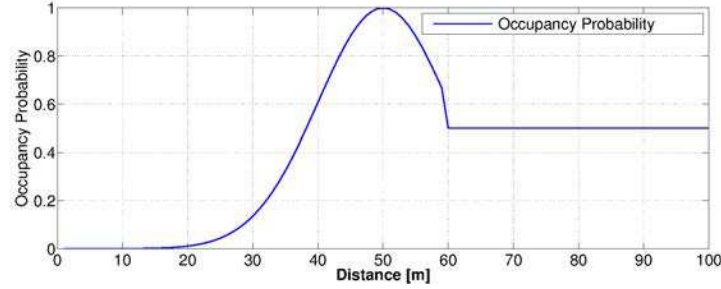


Figure 6.16: Profile of lidar inverse model for a beam hit on an obstacle at 50 meters

#### 6.2.4 Experimental Results

We implement aforementioned steps and show occupancy grid maps created by our equipped single layer lidar SICK LMS221. Angular resolution of the lidar is set to one degree, which allows to emit 181 laser beams. The maximum detected range is 80m. A grid map  $\mathcal{M}$  covers an area of  $30m \times 30m$ . The size of each cell  $C_{i,j}$  is  $20cm \times 20cm$ .  $f(r^k) = r^k/30$ ,  $\lambda = 1$ . We only show the occupancy grid map build by lidar measures within the view of the stereoscopic system. Several results are shown in Fig. 6.17 and 6.18. In the created maps, gray grids represent free areas, black cells are unknown space and white cells are occluded regions. Around the hit point of the lidar beam, inverse lidar model spread the uncertainty due to the measured distance. From the experimental results, we can find that, lidar measures and their corresponding occupancy grid maps are well in measuring distance. However, lidar is not able to continually observe the environment (it sparsely scans only in one layer). Some obstacles are not detected in lidar based occupancy grid map.

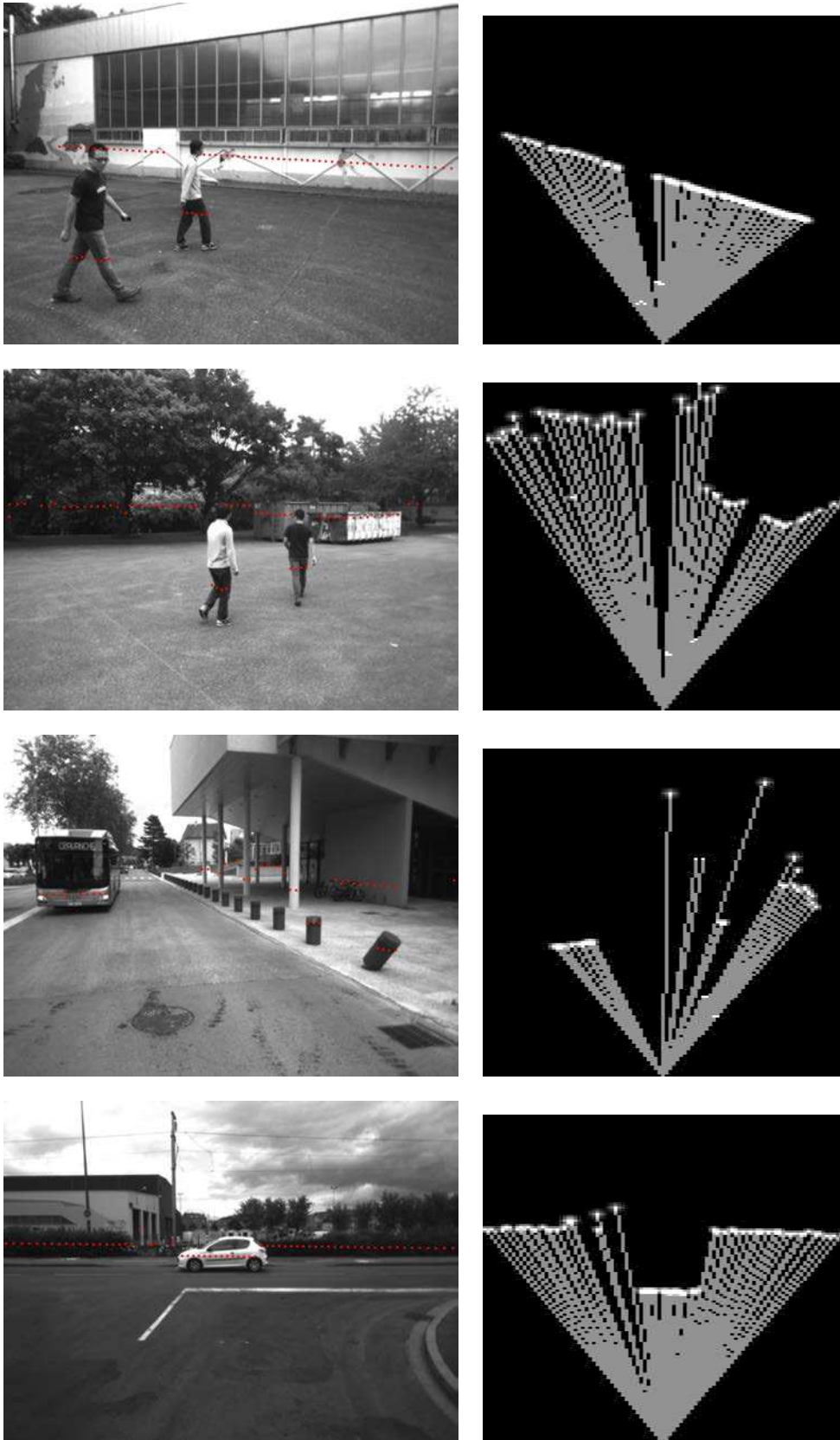


Figure 6.17: (left) road scenarios and projection of lidar measures; (right) corresponding occupancy grid maps

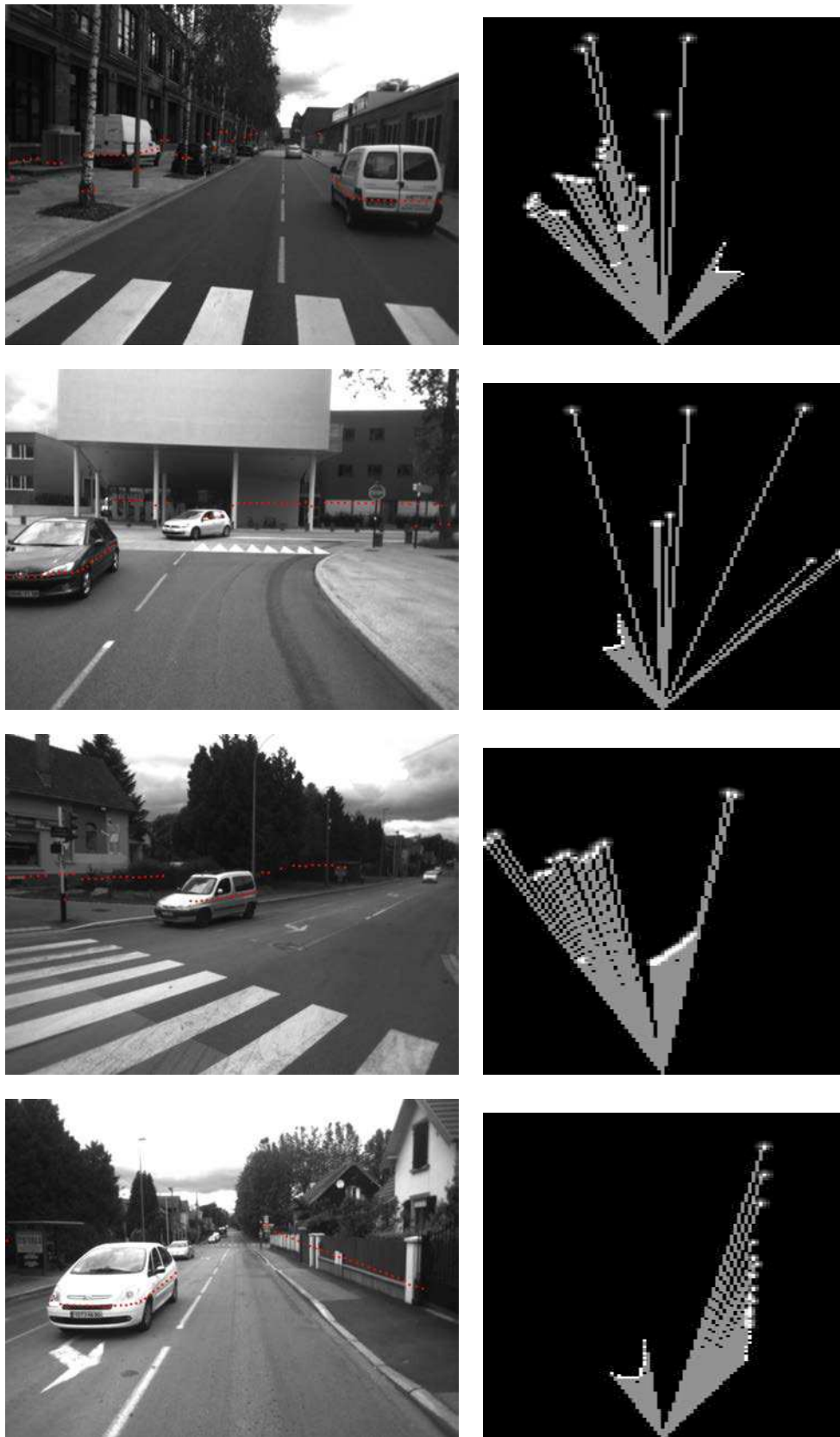


Figure 6.18: (left) road scenarios and projections of lidar measures; (right) corresponding occupancy grid maps

## 6.3 Occupancy Grid Mapping by Fusing Lidar and Stereo Vision

### 6.3.1 Introduction

Standard data fusion methods used for navigation tasks incorporate integration of data from sets of onboard sensors of the same type, or at least of those based on similar sensing principles. Typical classes for such sensors are range finders, data from other localization systems, etc. As the goal is to achieve maximum robustness of after fusion data, it is assumed that further improvements might be obtained by direct integration across different kinds of sensing systems. It is expected that this will introduce new possibilities in the fusion field. The expectation is to bring better performance through exploring capabilities to combine multiple sources in situations when some sensors fail. A straightforward candidate for this is direct fusion of range measuring devices with intensity images. As we stated before, the lidar is superior to stereoscopic system in distance measuring. In adverse, stereoscopic system provides more integral appearance descriptions of the environment. In literature, many methods are proposed to integrate data from lidar and stereoscopic system. These methods can be roughly divided into two categories:

- Object level fusion. In [Flórez 2011], moving objects are firstly detected by a multi-layer lidar. Then, the objects are verified by a stereoscopic system to improve integrity of a driving assistance system. Similar to this, the method proposed in [Aycard 2011] separately detects moving objects by a multi-layer lidar and a stereoscopic system. The independently detected objects are merged for further tracking and classification.
- Map level fusion. In [Stepan 2005], occupancy grid maps are built independently by a monocular camera and a lidar. With considering different accuracy of the two sensors, the combination of them leads to more accurate 2D maps of the environment. [Moras 2011] proposed a credibilist approach used to model the sensor information to create occupancy grid map. Conflicting information are handled by Dempster-Shafer theory. Another fusion model named linear opinion pools is proposed in [Adarve 2012]. It models the confidences of lidar measures and stereo data for integrating them in an occupancy grid map.

### 6.3.2 Fusing by Linear Opinion Pool

In our application, we choose the linear opinion pool proposed in [DeGroot 1974, Adarve 2012] to integrate the lidar based and stereo based occupancy grid maps together. The original idea is to perform fusion as a weighted sum of sensor observations. The weight is estimated as a confidence on every sensor's observation. By combining these confidences, the opinion of a non reliable sensor is lowered by a low confidence and hence, the result of the fusion process will depend on those reliable sensors which receive a higher weight.

In [Adarve 2012], under the linear opinion pool, the posterior distribution over the occupancy of a cell  $C_{i,j}$  gives the opinion of  $m$  sensors  $Y_1 \dots Y_m$ . Each sensor gives two quantities: its estimation for the occupancy of the cell  $P_{i,j}(O|Y_k)$  and  $w_k(C_{i,j})$ , a measure of the confidence for such estimations. Under the assumption of mutually independent cells, the fusion of all sensory information is as follows:

$$P_{i,j}(O|Y^1 \dots Y^m) = \alpha \sum_{k=1}^m w_k(C_{i,j}) P_{i,j}(O|Y^k) \quad (6.25)$$

where  $\alpha = [\sum_k w_k(C_{i,j})]^{-1}$  is a normalization factor for the weights. In our cases, the sensor measures either come from the lidar or stereoscopic system. Since we have already got occupancy grid maps from stereo vision system and lidar. Different to [Adarve 2012], where every measurement from stereoscopic system is assigned with a weight, we perform the fusion between the lidar measures and occupancy grid map made by stereoscopic system respectively. Hence, the fusion process becomes:

$$P_{i,j}(O|Y^1 \dots Y^m) = \alpha \sum_{k=1}^m w_k(C_{i,j}) P_{i,j}(O|Y_{lidar}^1 \dots Y_{lidar}^{m-1}, Y_{stereo}^m) \quad (6.26)$$

Based on equation 6.26, for each sensor measure  $Y^k$  (either from lidar or from stereoscopic system), we have to define  $P(O|Y^k)$ , the probability of a cell being occupied given the sensor information; and  $w_k(C_{i,j})$ , the confidence on the opinion.

### 6.3.2.1 Align Lidar with Stereoscopic System

The lidar measures and stereoscopic data are located in two different coordinate systems,  $\mathcal{R}_{stereo}^3$  and  $\mathcal{R}_{lidar}^3$ . Before applying a fusion of the data, the measurements from the two coordinate systems have to be correctly aligned.

The alignment is based on the extrinsic calibration results presented in chapter 5. After the extrinsic calibration, the 3D rigid transformation  $\phi, \Delta$  from lidar to stereoscopic coordinate system is acquired. Hence, before the fusion, all the lidar measures are transformed into stereoscopic coordinate system according to Eq. 5.3. Next, we will discuss the models of confidences.

### 6.3.2.2 Confidences of Lidar and Stereoscopic System

The occupancy probability of a cell  $C_{i,j}$  being occupied given the sensor measurement is given by Eq. 6.22 (lidar) or Eq. 6.17 (stereo). The confidences of the two kinds of measurements are described as follows.

For the confidence of a lidar measurement, consider the probability that an unexpected object is detected before the real hit of the laser beam. This is modeled by:

$$w_{lidar}(z, z^*) = \begin{cases} \beta & \text{for } z \in [0, z^*] \\ \beta e^{\frac{-(z-z^*)^2}{2\sigma^2}} & \text{for } z \in (z^*, z_{max}] \end{cases} \quad (6.27)$$

where  $z^*$  is the detected range,  $\beta$  is the value of occupancy probability at the hit



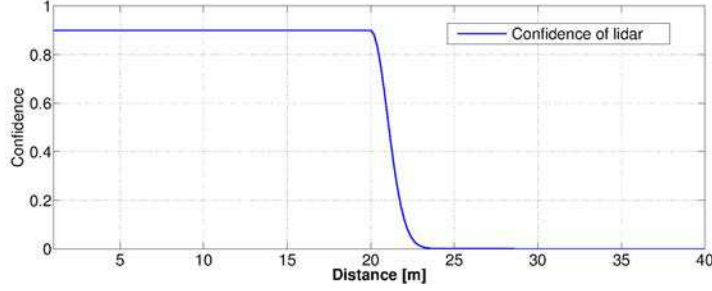


Figure 6.19: The confidence of lidar measurement ( $z^* = 20, \beta = 0.9, \sigma = 1$ )

position. After the hit, the confidence decreases following a Gaussian function. An example is shown in Fig. 6.19.

For the stereoscopic system, we model its measuring confidence according to the distance. In fact, the confidence models the fact that stereo vision works better at short distances, decreasing its accuracy when obstacles are far away. let  $d_{max}$  be the maximum possible detected range of the stereoscopic system ( $d_{max} = focallength * baselinelength$ ), the measuring confidence is modeled by:

$$1 - \frac{d^2}{d_{max}^2} \quad (6.28)$$

Therefore, based on the occupancy probability models in Eq. 6.22, 6.17 and confidence models in Eq. 6.27 and Eq. 6.28, we apply the linear opinion pool (Eq. 6.26) to integrate data from lidar and stereoscopic system in an occupancy grid map.

### 6.3.3 Experimental Results

The fusion experiments are performed in the dataset acquired by our platform. The image data and lidar data are firstly synchronized according to their storing time in our platform. Then, linear opinion pool based fusing is performed. In our experiments, the parameters for confidence models are set as:  $\beta = 0.95, \sigma = 0.5, d_{max} = 80m$ . Fig. 6.20, 6.21 and 6.22 demonstrate the fusion results. Because our lidar sparsely scans the environment in a single plane, sometimes, it would miss the objects due to sudden bump (Fig. 6.20 (a)), or road geometry (Fig. 6.22 (b)), or the black objects which would absorb the laser beams (Fig. 6.21 (a)). From the results, we can see that fusing the lidar and stereoscopic measurements makes the occupancy grid map more integral in observing the environment than the lidar based occupancy grid map. However, the fusion does not help the occupancy grid mapping results based on stereo vision system. The main reason is that, stereo vision system observes the environment much more densely than the single layer 2D lidar we used. Hence, most of the lidar detections within the view of stereoscopic system are already observed by vision system.

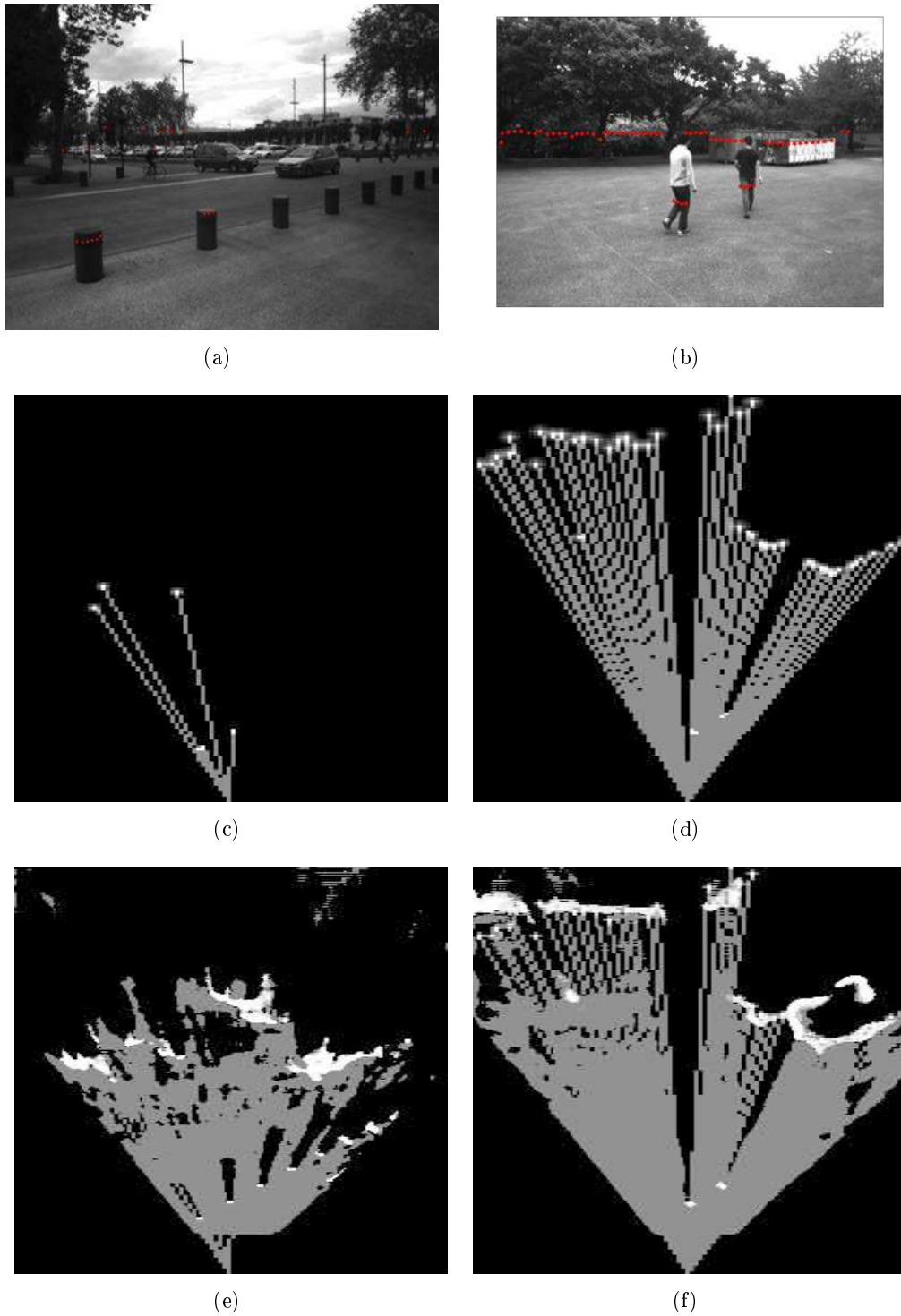


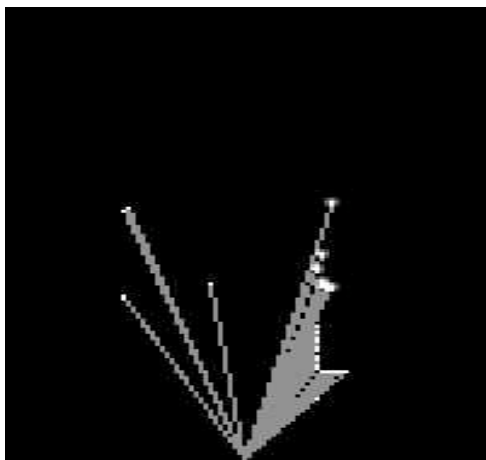
Figure 6.20: Experimental results: (top) scenarios in urban environment; (middle) occupancy grid map based on lidar; (bottom) occupancy grid map by fusing lidar and stereoscopic measurements.



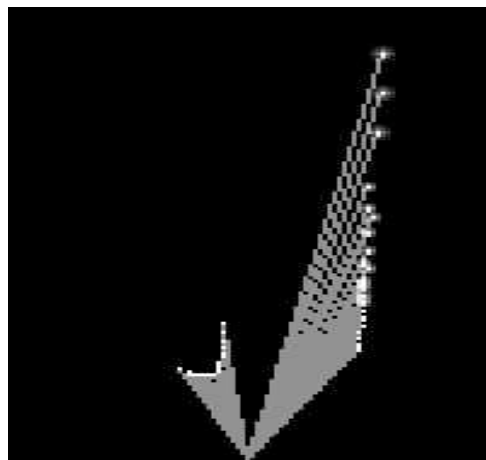
(a)



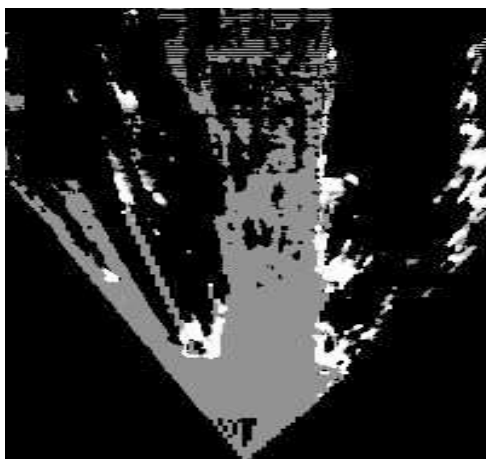
(b)



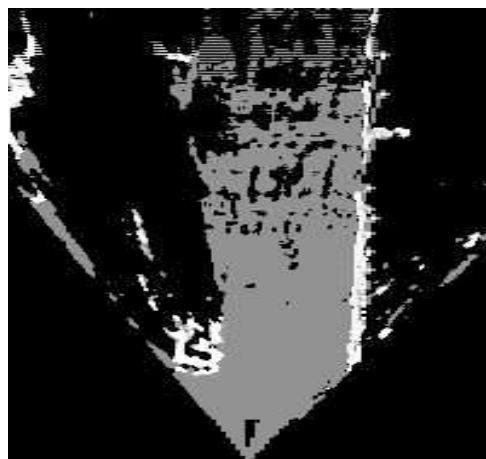
(c)



(d)



(e)



(f)

Figure 6.21: Experimental results: (top) scenarios in urban environment; (middle) occupancy grid map based on lidar; (bottom) occupancy grid map by fusing lidar and stereoscopic measurements.

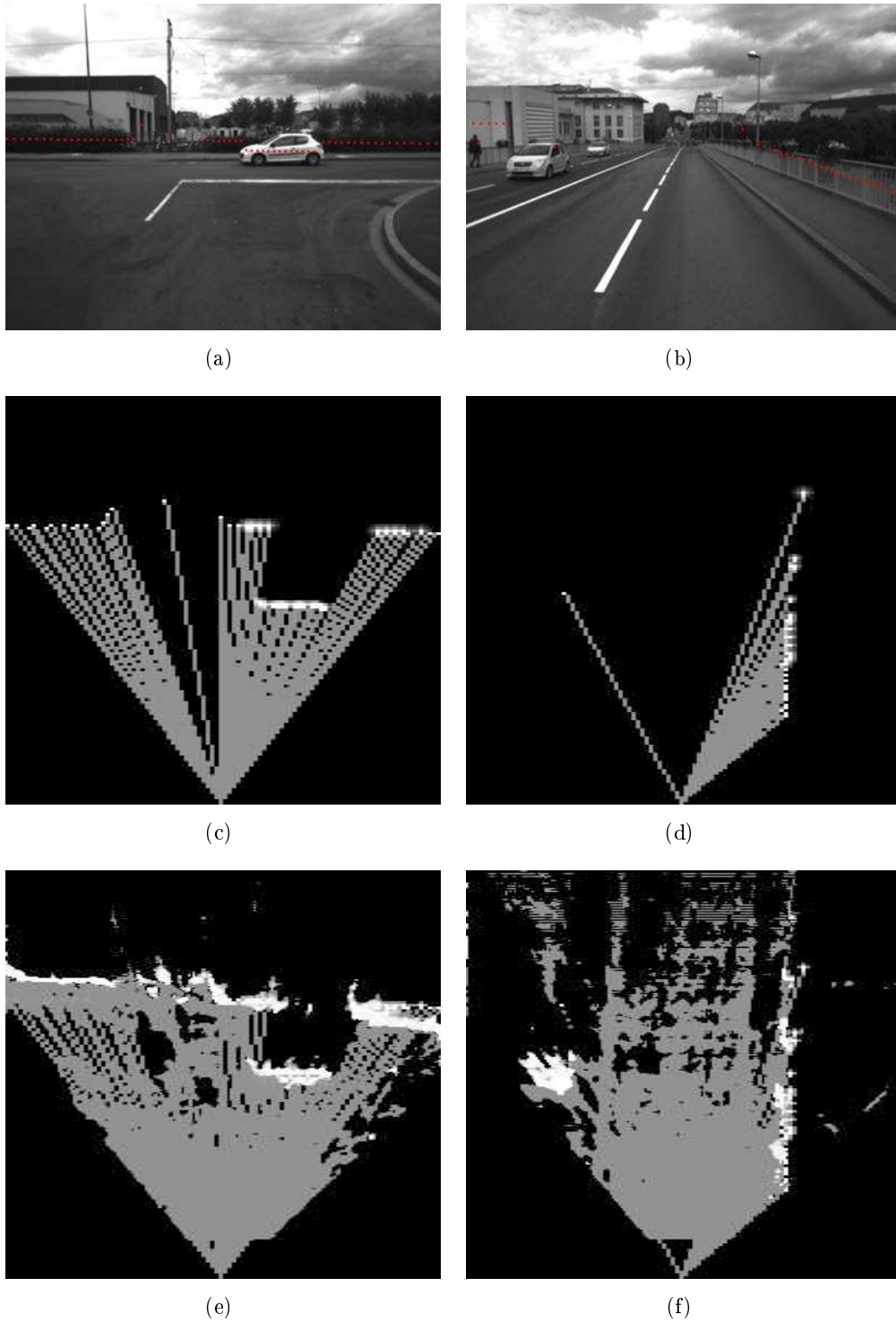


Figure 6.22: Experimental results: (top) scenarios in urban environment; (middle) occupancy grid map based on lidar; (bottom) occupancy grid map by fusing lidar and stereoscopic measurements.

## 6.4 Conclusion and Future Works

In this chapter we described a framework of generating occupancy grid map from stereoscopic system, lidar and combination of both. Stereoscopic based occupancy grid mapping is achieved by firstly analyzing ground plane geometric characteristic (pitch angle) and pixel-wisely subtract ground plane in the image. Next, reconstructed 3D points are corrected by estimated pitch angles to improve the performance of occupancy grid map. Occupancy probability of each grid is calculated based on associated 3D points in each grid. In addition, independent moving objects and their categories are also labelled in the occupancy grid map by the processing results of chapter 4.

Then, we also create occupancy grid map based on a classic inverse sensor model of a single layer lidar. Since single layer lidar is not stable in observing the environment, we prove that fusing stereo information could improve the quality of occupancy grid map. However, for the stereo vision based occupancy grid map, fusing with 2D lidar data does not bring sufficient improvements. The main reason of this problem is that, in our platform, a single layer 2D lidar is not robust enough to observe the environment.

For the future works, the current works could be improved by a Bayesian occupancy filter that is capable of accurately estimating and tracking the state of dynamic environments. Also, we intend to further develop tracking method in the occupancy grid map. To improve the fusion of stereo vision based and lidar based occupancy grid map, we hope the multi-layer lidar or 3D lidar could be helpful.

# Conclusions and Future Works

---

## 7.1 Conclusions

The problems addressed in this thesis concern several basic topics of perception systems in intelligent vehicles. Perception systems are the eyes of intelligent vehicles, which are in charge of environment analyzing and understanding. After a review of basic knowledges used in the thesis, we proposed a series of methods to analyze the environment.

In the first, a stereo vision based ego-motion estimation method is presented in chapter 3. The stereo vision based visual odometry has been researched for decades. Hence, we just follow a classic framework shown in Fig. 3.1. Although this framework is relatively stable, in literature, it lacks engineering comparisons of different image feature detectors, as well as different kinds of feature association methods. We compare several representative feature detectors and proposed several standards to choose the suitable feature detectors. In addition, we also compare tracker based with matching based feature association approaches. From the comparison, we find that CenSURE (or STAR) feature detector with KLT feature tracker achieve the best performance. The selected approach is tested both in an open dataset and our dataset. The real experimental results show that, compared with GPS, visual odometry still lacks accuracy, especially in long distance.

In the second, an independent moving object detection, segmentation and recognition method is proposed in chapter 4. The moving object detection is based on the results of visual odometry. The outliers discarded by ego-motion estimation are used for detecting moving objects in U-disparity image. The segmentation is also performed in U-disparity image by a flood-fill algorithm. Then, spatial features of segmented moving objects are extracted and sent to a pre-trained classifier for recognizing three common categories of objects: pedestrian, vehicle and motor/cyclist. In our proposed method, kernel PCA algorithm is adopted to boost original simple spatial features by mapping the low dimensional data into high dimensional space. Several commonly used classifiers, such as random forest, gradient boost trees are compared to get the best classifier. The real experiments show that gradient boost trees achieve best recognition rate.

In the third, we developed a method to achieve optimal extrinsic calibration between the stereoscopic system and a 2D lidar using 3D reconstruction from stereo vision. We improve the accuracy of extrinsic calibration by considering the measuring models of the two sensors. The introduction of sensor models make the estimation (stereo estimation of 3D planes and final estimation of the extrinsic pa-

rameters) more precise. The improvement of accuracy is demonstrated by comparing our method to a popular method.

At last, occupancy grid map, a classic tool of interpreting environment, is built by stereoscopic system, lidar and fusing them together. For better mapping the environment by stereoscopic system, we estimate the pitch angle of the ground plane with regards to the stereoscopic system. The pitch angle compensation greatly improve the quality of occupancy grid map. Furthermore, the moving object detection and recognition results acquired previously are integrated to enhance the occupancy grid map. As for a complementary sensor to stereoscopic system, we also create occupancy grid map from 2D lidar measurements. Then, a linear opinion pool based fusing method is proposed to integrate the two kinds of measurements into one occupancy grid map. Real experimental results show that, the fusion improve the quality of lidar based occupancy grid map. However, the fusion does not improve the quality of stereo based occupancy grid map, due to 2D lidar's sparse data.

## 7.2 Future Works

In the author's point of view, many perspectives are envisaged to improve the current works in the future.

To improve the accuracy of stereo vision based visual odometry, several approaches could be tried:

- Develop more accurate feature detectors and feature tracking method.
- Improve the accuracy of current feature detector by estimating the sub-pixel positions.
- Consider fusing the visual odometry with other sensors, such as IMU.

For the independent moving object detection/segmentation/recognition, the possible improvements maybe:

- Combine the moving object detection with object recognition, which maybe solve the problem of failure in detecting moving pedestrians.
- Better segmentation method directly performed in raw image should be tried. The current segmentation in disparity image is not quite satisfied.
- The extracted spatial features could boost the visual features (HoG, LBP, Haar). We hope to furthermore improve the recognition results by combining these two kinds of features.

For the works of calibration, in the future, we plan to introduce more different sensors, e.g. multi-layer lidar, 3D lidar. Hence, we plan to calibrate a multiple sensor system.

At last, for the occupancy grid mapping of the environment, the potential improvements are:

- 
- Bayesian filtering framework could be implemented to guide the update of the map in sequence.
  - Implement object tracking in the occupancy grid map.
  - Semantic environmental interpretation could be added into the occupancy grid map. For example, we hope to know where is the road, buildings, pedestrians and vehicles in the map. The semantic information could be either from the object detectors or semantic image segmentation methods.
  - To improve the fusing results of lidar and stereo vision system, more stable and powerful lidar, such as multi-layer lidar, 3D lidar could replace the original single layer lidar.

After these possible improvements, an autonomous navigation system is able to achieved based on the current works. We hope the works presented in the thesis would be helpful for the following Ph.D students who research in this area.





# Publications

---

**Journal:**

1. Y. Li, Y. Ruichek and C. Cappelle, "*Optimal Extrinsic Calibration between a Stereoscopic System and a Lidar*", in IEEE Transactions on Instrumentation & Measurements 2013 (IF: 1.3).

**Conference:**

1. Y. Li, Y. Ruichek, C. Cappelle "*3D Triangulation based Extrinsic Calibration between a Stereo Vision System and a LIDAR*", in 14th IEEE International Conference on Intelligent Transportation Systems (ITSC), Washington D.C., USA, 2011.
2. Y. Li, Y. Ruichek "*Moving Objects Detection and Recognition Using Sparse Spacial Information in Urban Environments*", in Proc. of IEEE Intelligent Vehicles Symposium (IVS), Alcala de Henares, Spain, 2012.
3. Y. Li, Y. Ruichek, C. Cappelle "*Calibrage Extrinsèque Entre Un Système Stéréoscopique Et Un Télémètre Laser*", In Proc. of Conférence Internationale Francophone d'Automatique (CIFA 2012), Grenoble, France, 2012.
4. Y. Li, Y. Ruichek and C. Cappelle "*Extrinsic Calibration between a Stereoscopic System and a Lidar with Sensor Noise Models*", in Proc. of IEEE Conference on Multisensor Fusion and Information Integration (MFI), Hamburg, Germany, 2012.
5. Y. Li, Y. Ruichek "*Building Variable Resolution Occupancy Grid Map from Stereoscopic System - a Quadtree based Approach*", in Proc. of IEEE Intelligent Vehicles Symposium (IVS), Gold Coast, Australia, 2013.
6. Y. Li, Y. Ruichek "*Observing Dynamic Urban Environment through Stereovision based Dynamic Occupancy Grid Mapping*", IAPR 17th International Conference on Image Analysis and Processing (ICIAP), Napoli, Italy, 2013.
7. Y. Li, Y. Ruichek "*Perception of dynamic urban environments using stereovision based dynamic occupancy grid map*", in Transport Research Arena, Paris, 2014.



# Bibliography

- [Adarve 2012] Juan Adarve, Mathias Perrollaz, Alexandros Makris and Christian Laugier. *Computing Occupancy Grids from Multiple Sensors using Linear Opinion Pools*. In In Proc. IEEE International Conference on Robotics and Automation (ICRA), 2012. (Cited on pages [142](#) and [143](#).)
- [Agrawal 2008] Motilal Agrawal, Kurt Konolige and Morten Rufus Blas. *CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching*. In In Proc. of the European Conference on Computer Vision, 2008. (Cited on pages [38](#) and [57](#).)
- [Aliakbarpour 2009] H. Aliakbarpour, P. Núñez, J. Prado, K. Khoshhal and J. Dias. *An Efficient Algorithm for Extrinsic Calibration between a 3D Laser Range Finder and a Stereo Camera for Surveillance*. In 14th International Conference on Advanced Robotics, 2009. (Cited on page [92](#).)
- [Aycard 2011] Olivier Aycard, Qadeer Baig, Siviú Bota, Fawzi Nashashibi and Sergiu Nedevschi. *Intersection safety using lidar and stereo vision sensors*. In in Proc. IEEE Intelligent Vehicles Symposium (IV), 2011. (Cited on page [142](#).)
- [Badino 2007] Hernán Badino, Uwe Franke and Rudolf Mester. *Free Space Computation Using Stochastic Occupancy Grids and Dynamic*. In Programming, Proc. Intl Conf. Computer Vision, Workshop Dynamical Vision, 2007. (Cited on pages [120](#), [123](#) and [130](#).)
- [Bay 2008] Herbert Bay, Andreas Ess, Tinne Tuytelaars and Luc Van Gool. *Speeded-Up Robust Features (SURF)*. Journal of Computer Vision and Image Understanding, vol. 110, pages 346–359, 2008. (Cited on pages [37](#), [38](#), [40](#) and [57](#).)
- [Beardsley 1994] P.A. Beardsley, A. Zisserman and D. Murray. *Navigation using affine structure from motion*. In in European Conference on Computer Vision (ECCV), pages 85–96. Springer-Verlag, 1994. (Cited on page [96](#).)
- [Bill 1999] Triggs Bill, McLauchlan Philip and Richard Hartley. *Bundle Adjustment - A Modern Synthesis*. In Proceedings of the International Workshop on Vision Algorithms: Theory and Practice, ICCV’1999, pages 298–372. Springer-Verlag, 1999. (Cited on page [48](#).)
- [Bishop 2005] Richard Bishop. Intelligent vehicle technology and trends. Artech House, 2005. (Cited on page [1](#).)
- [Bjorck 1996] Ake Bjorck. Numerical methods for least squares problems. Society for Industrial and Applied Mathematics (SIAM), 1996. (Cited on page [26](#).)

- [Bo 2011] Liefeng Bo and Dieter Fox. *Depth Kernel Descriptors for Object Recognition*. In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pages 821–826, Seattle, USA, 25-30 Sept 2011. (Cited on page 82.)
- [Bouguet 1999] Jean-Yves Bouguet. *Visual methods for three-dimensional modeling*. PhD thesis, California Institute of Technology, USA, 1999. (Cited on page 14.)
- [Bouguet 2000] Jean-Yves Bouguet. *Pyramidal implementation of the Lucas Kanade feature tracker*. Inter Corporation, Microprocessor Research Labs, 2000. (Cited on page 43.)
- [Bouguet 2010] Jean-Yves Bouguet. *Camera Calibration Toolbox for Matlab*. [http://www.vision.caltech.edu/bouguetj/calib\\_doc](http://www.vision.caltech.edu/bouguetj/calib_doc), 2010. (Cited on page 96.)
- [Brailon 2006] Christophe Brailon, Cédric Pradalier, Kane Usher, Jim Crowley and Christian Laugier. *Occupancy grids from stereo and optical flow data*. In Proc. of the Int. Symp. on Experimental Robotics, 2006. (Cited on page 120.)
- [Breiman 1984] L. Breiman, J. Friedman and J. Stone. Classification and regression trees. Chapman and Hall(CRC), 1984. (Cited on page 83.)
- [Breiman 2001] Leo Breiman. *Random Forests*. Machine Learning, vol. 45, pages 5–32, 2001. (Cited on pages 80 and 83.)
- [Bresenham 1965] J. Bresenham. *Algorithm for computer control of a digital plotter*. IBM Systems Journal, vol. 4, pages 25–30, 1965. (Cited on page 136.)
- [Bretscher 1995] Otto. Bretscher. Linear algebra with applications. Prentice Hall, 1995. (Cited on page 20.)
- [Bretzner 1998] L. Bretzner and T. Lindeberg. *Feature Tracking with Automatic Selection of Spatial Scales*. Journal of Computer Vision and Image Understanding, vol. 71, pages 385–392, 1998. (Cited on page 36.)
- [Brostow 2008] Gabriel Brostow and Jamie Shotton. *Segmentation and Recognition Using Structure from Motion Point Clouds*. In Proceedings of the 10th European Conference on Computer Vision: Part I, pages 44–57, 2008. (Cited on pages 80 and 81.)
- [Brown 2003] M. Brown and D. Lowe. *Recognising panoramas*. In International Conference on Computer Vision, 2003. (Cited on page 30.)
- [Burges 1998] Christopher Burges. *A Tutorial on Support Vector Machines for Pattern Recognition*. Data Mining and Knowledge Discovery, vol. 2, pages 121–167, June 1998. (Cited on page 80.)

- [Calonder 2010] Michael Calonder, Vincent Lepetit, Christoph Strecha and Pascal Fua. *BRIEF: Binary Robust Independent Elementary Features*. In In Proc. of European Conference on Computer Vision, 2010. (Cited on page 41.)
- [Chartrand 2008] R. Chartrand and W. Yin. *Iteratively reweighted algorithms for compressive sensing*. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2008. (Cited on page 22.)
- [Cheng 2006] Y. Cheng, M. Maimone and L. Matthies. *Visual odometry on the mars exploration rovers*. IEEE Robotics & Automation Magazine, vol. 13, pages 54–62, 2006. (Cited on page 49.)
- [Cheng 2007] M. Maimone AND Y. Cheng and L. Matthies. *Two years of visual odometry on the mars exploration rovers: Field reports*. Journal of Field Robotics, vol. 24, pages 169–186, 2007. (Cited on page 49.)
- [Chili 2008] M. Chili and A. Davison. *Active Matching*. In In Proceedings of European Conference on Computer Vision, 2008. (Cited on page 46.)
- [Cole 2006] D.M. Cole and P.M. Newman. *Using laser range data for 3D SLAM in outdoor environments*. In IEEE International Conference on Robotics and Automation, pages 1556–1563, may 2006. (Cited on page 4.)
- [Comport 2007] A. Comport, E. Malis and P. Rives. *Accurate quadrifocal tracking for robust 3D visual odometry*. In IEEE International Conference on Robotics and Automation, pages 40–45, 2007. (Cited on page 49.)
- [Cristiano 2009] Ludwig. Cristiano Premebida. Oswaldo and Nunes. Urbano. *LI-DAR and vision-based pedestrian detection system*. Journal of Field Robotics, vol. 26, Issue 9, pages 696–711, 2009. (Cited on page 92.)
- [Dalal 2005] N. Dalal and B. Triggs. *Histograms of oriented gradients for human detection*. In Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, volume 1, pages 886 –893, june 2005. (Cited on page 80.)
- [Danescu 2009] Radu Danescu, Florin Oniga, Sergiu Nedevschi and Marc-Michael Meinecke. *Tracking Multiple Objects Using Particle Filters and Digital Elevation Maps*. In Intelligent Vehicles Symposium, IEEE, 2009. (Cited on pages 120 and 130.)
- [Davision 2007] A. Davision, I. Reid, N. Molton and O. Stasse. *MonoSLAM: Real-Time Single Camera SLAM*. IEEE Trans. Pattern Analysis and Machine Intelligence,, vol. 29, no. 6, pages 1052–1067, June 2007. (Cited on page 3.)
- [DeGroot 1974] M. H. DeGroot. *Reaching a consensus*. Journal of the American Statistical Association, vol. 69, pages 118–121, 1974. (Cited on page 142.)

- [Dellaert 2000] F. Dellaert, S. Seitz and S. Thrun. *Structure from Motion with Correspondence*. In IEEE Conference on Computer Vision and Pattern Recognition, 2000. (Cited on page 48.)
- [Dey 2012] Soumyabrata Dey and Vladimir Reilly. *Detection of independently moving objects in non-planar scenes via multi-frame monocular epipolar constraint*. In Proceedings of the 12th European conference on Computer Vision (ECCV), 2012. (Cited on page 72.)
- [Dominguez 2007] Eugenio Dominguez. *Computational Photography*. <http://www.cs.cmu.edu/afs/andrew/scs/cs/15-463/f07/proj4/www/edomingu/>, 2007. (Cited on page 33.)
- [Elfes 1989] A. Elfes. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. PhD thesis, Carnegie Mellon University, 1989. (Cited on page 118.)
- [Fischler 1981] Martin A. Fischler and Robert C. Bolles. *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*. Communications of the ACM, vol. 24, pages 381–395, 1981. (Cited on page 28.)
- [Fletcher 2008] Luke Fletcher, Seth Teller and et al. *The MIT-Cornell Collision and Why it Happened*. Journal of Field Robotics, vol. 25, pages 775–807, 2008. (Cited on page 3.)
- [Flórez 2011] Rodríguez Flórez, Frémont Vincent and Philippe Bonnifait. *Multi-modal object detection and localization for high integrity driving assistance*. Machine Vision and Applications, pages 1–16, 2011. (Cited on page 142.)
- [Folkesson 2004] J. Folkesson. *Graphical SLAM - a self-correcting Map*. In Proceedings of International Conference on Robotics and Automation (ICRA), 2004. (Cited on page 118.)
- [Fox 2007] D. Fox and D. Ramos. *A spatio-temporal probabilistic model for multi-sensor object recognition*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2402–2408, San Diego, CA, 2007. (Cited on page 92.)
- [Freund 1995] Yoav Freund and Robert E. Schapire. *A decision-theoretic generalization of on-line learning and an application to boosting*. In Proceedings of the Second European Conference on Computational Learning Theory, pages 23–37, London, UK, 1995. (Cited on page 80.)
- [Friedman 1977] J. Friedman, J. Bentley and R. Finkel. *An algorithm for finding best matches in logarithmic expected time*. ACM Transactions on Mathematical Software, vol. 3, pages 209–226, 1977. (Cited on page 45.)

- [Friedman 1999] J. Friedman. *Stochastic Gradient Boosting*. Computational Statistics and Data Analysis, vol. 38, pages 367–378, 1999. (Cited on page 83.)
- [Friedman 2000] Jerome H. Friedman. *Greedy Function Approximation: A Gradient Boosting Machine*. Annals of Statistics, vol. 29, pages 1189–1232, 2000. (Cited on pages 80 and 83.)
- [Gauglitz 2011] Steffen Gauglitz, Tobias Hollerer and Matthew Turk. *Evaluation of Interest Point Detectors and Feature Descriptors for Visual Tracking*. International Journal of Computer Vision, vol. 94, pages 335–360, 2011. (Cited on page 40.)
- [Gavrila 2004] D.M. Gavrila, J. Giebel and S. Munder. *Vision-based pedestrian detection: the PROTECTOR system*. In IEEE Intelligent Vehicles Symposium, 2004, pages 13–18, june 2004. (Cited on page 3.)
- [Geiger 2011] Andreas Geiger, Julius Ziegler and Christoph Stiller. *StereoScan: Dense 3d Reconstruction in Real-time*. In IEEE Intelligent Vehicles Symposium, Baden-Baden, Germany, June 2011. (Cited on pages 49 and 50.)
- [Geiger 2012] Andreas Geiger, Philip Lenz and Raquel Urtasun. *Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite*. In in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012. (Cited on page 62.)
- [Gleser 1981] Leon Jay Gleser. *Estmethods in a Multivariate "Errors in Variables" Regression Model: Large Sample Results*. The Annals of Statistics, vol. 9, pages 24–44, 1981. (Cited on page 23.)
- [Godambe 1991] V.P. Godambe. Estimating functions. Oxford University Press, 1991. (Cited on page 28.)
- [Gonzalez 1992] R. Gonzalez. Digital image processing. Addison-Wesley Pub, 1992. (Cited on page 75.)
- [Guizilini 2012] V. Guizilini. *Semi-parametric models for visual odometry*. In in Proceedings of IEEE International Conference on Robotics and Automation, 2012. (Cited on page 48.)
- [Handa 2010] A. Handa, M. Chili. and A. Davison. *Scalable active matching*. In In Proceedings of Computer Vision and Pattern Recognition, 2010. (Cited on page 46.)
- [Harris 1988] Chris Harris and Mike Stephens. *A combined corner and edge detector*. In In Proc. of Fourth Alvey Vision Conference, 1988. (Cited on page 32.)
- [Hartley 1997] Richard I. Hartley and Strum Peter. *Triangulation*. Computer Vision and Image Understanding, vol. 68, no. 2, pages 146–157, November 1997. (Cited on page 96.)



- [Hartley 2004] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*, second edition. Cambridge University Press, 2004. (Cited on pages 14, 15, 16 and 96.)
- [Hayashi 2011] Fumio Hayashi. *Econometrics*. Princeton University Press, 2011. (Cited on page 21.)
- [Hirschmueller 2008] Heiko Hirschmueller. *Stereo processing by semiglobal matching and mutual information*. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 30, pages 328–341, Feb 2008. (Cited on page 73.)
- [Hoiem 2008] Derek Hoiem and Alexei Efros. *Putting Objects in Perspective*. International Journal of Computer Vision, vol. 80, pages 3–15, October 2008. (Cited on page 80.)
- [Huang 2009a] Lili Huang and M. Barth. *A novel multi-planar LIDAR and computer vision calibration procedure using 2D patterns for automated navigation*. IEEE Intelligent Vehicles Symposium, pages 117–122, june 2009. (Cited on page 93.)
- [Huang 2009b] Lili Huang and M. Barth. *Tightly-coupled LIDAR and computer vision integration for vehicle detection*. In IEEE Intelligent Vehicles Symposium, volume 1, pages 604 – 609, 2009. (Cited on pages 92, 94 and 100.)
- [Huffel 1991] S. Van Huffel and Vanderwalle. The total least squares problem: Co-computation aspects and analysis. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1991. (Cited on pages 22 and 23.)
- [Hwang 2007] Jae Pil Hwang, Seung Eun Cho, Kyung Jin Ryu, Seungkeun Park and Euntai Kim. *Multi-Classifer Based LIDAR and Camera Fusion*. In IEEE International Conference on Intelligent Transportation Systems, pages 467–472, Oct 2007. (Cited on page 92.)
- [Kalman 1960] R. Kalman. *A New Approach to Linear Filtering and Prediction Problems*. Transactions of the ASME—Journal of Basic Engineering, vol. 82, pages 35–45, 1960. (Cited on page 126.)
- [Kanatani 2008] K. Kanatani, Y. Sugaya and H. Niitsuma. *Triangulation from two views revisited: Hartley-Sturm vs. optimal correction*. In 19th British Machine Vision Conference, pages 173–182, Leeds, 2008. (Cited on page 96.)
- [Kang 2005] Jinman Kang, Issac Cohen and Chang Yuan. *Detection and Tracking of Moving Objects from a Moving Platform in Presence of Strong Parallax*. In Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV), 2005. (Cited on page 72.)
- [Kaucic 2005] R. Kaucic. *A Unified Framework for Tracking through Occlusions and across Sensor Gaps*. In Proceedings of the IEEE Computer Society

- Conference on Computer Vision and Pattern Recognition (CVPR), 2005. (Cited on page 72.)
- [Kolmogorov 2002] V. Kolmogorov and R. Zabih. *Computing visual correspondence with occlusions using graph cuts*. In in Proceedings of International Conference on Computer Vision, 2002. (Cited on page 73.)
- [Konolige 1998] K. Konolige. *Small Vision Systems: Hardware and Implementation*. In Robotics Research, pages 203–212. Springer London, 1998. (Cited on page 73.)
- [Labayarde 2002] R. Labayarde, Didier Aubert and Jean-Philippe Tarel. *Real time obstacle detection in stereovision on non flat road geometry through "v-disparity" representation*. In IEEE Intelligent Vehicle Symposium, volume 2, pages 646–651, 2002. (Cited on page 73.)
- [Lacroix 1999] S. Lacroix, A. Mallet and R. Chatila. *Rover self localization in planplane-like environments*. In International Symposium of Artificial Intelligence, Robotics and Automation for Space (i-SAIRAS), 1999. (Cited on page 49.)
- [Lamdan 1988] Y. Lamdan and H. Wolfson. *Geometric hashing: a general and efficient model-based recognition scheme*. In in Proceedings of International Conference of Computer Vision, 1988. (Cited on page 45.)
- [Leibe 2006] Bastian Leibe, Krystian Mikolajczyk and Bernt Schiele. *Segmentation Based Multi-Cue Integration for Object Detection*. In Proceedings of the British Machine Vision Conference, Edinburgh, UK, pages 1169–1178, 2006. (Cited on page 80.)
- [Leibe 2008a] B. Leibe, K. Schindler, N. Cornelis and L. Van Gool. *Coupled Object Detection and Tracking from Static Cameras and Moving Vehicles*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 30, no. 10, pages 1683 –1698, oct. 2008. (Cited on page 80.)
- [Leibe 2008b] Bastian Leibe, Ales Leonardis and Bernt Schiele. *Robust Object Detection with Interleaved Categorization and Segmentation*. International Journal of Computer Vision, vol. 77, pages 259–289, 2008. (Cited on page 80.)
- [Lenz 2011] 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), IEEE, pages 926–932, Baden-Baden, Germany, June 2011. (Cited on page 72.)
- [Leonard 1991] J. Leonard. *Mobile robot localization by tracking geometric beacons*. IEEE Transactions on Robotics and Automation, vol. 7, pages 376–382, 1991. (Cited on page 117.)

- [Leonard 2008] John Leonard. *A perception-driven autonomous urban vehicle*. Journal of Field Robotics, vol. 25, pages 727–774, October 2008. (Cited on page 117.)
- [Leutenegger 2011] S Leutenegger. *BRISK: Binary Robust invariant scalable keypoints*. In In Proc. of IEEE International Conference on Computer Vision, 2011. (Cited on pages 41, 42, 43 and 57.)
- [Levenberg 1944] K. Levenberg. *A method for the solution of certain problems in least-squares*. Quarterly Applied Math, pages 164–168, 1944. (Cited on page 26.)
- [Li 2007] Ganhua Li, Yunhui Liu, Li Dong, Xuanping Cai and Dongxiang Zhou. *An algorithm for extrinsic parameters calibration of a camera and a laser range finder using line features*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3854–3859, 2007. (Cited on page 92.)
- [Li 2010] Y. Li, L. Song and J. Wang. *Automatic Weak Calibration of Master-Slave Surveillance System Based on Mosaic Image*. In International Conference on Pattern Recognition, 2010. (Cited on page 30.)
- [Lindeberg 1994] T. Lindeberg. *Scale-space theory: A basic tool for analysing sstructure at different scale*. Journal of Applied Statistics, vol. 21, pages 224–270, 1994. (Cited on page 36.)
- [Lowe 1999] David Lowe. *Object Recognition from Local Scale-Invariant Features*. In Proceedings of the International Conference on Computer Vision-Volume 2, ICCV '99, pages 1150–, Washington. DC, USA, 1999. (Cited on page 80.)
- [Lowe 2004] David Lowe. *Distinctive Image Features from Scale-Invariant Keypoints*. International Journal of Computer Vision, vol. 60, pages 91–110, 2004. (Cited on pages 36, 38, 39, 40 and 57.)
- [Lucas 1981] B. Lucas and T. Kanade. *An iterative image registration technique with an application to stereo vision*. In in Proceedings of the 7th International Conference on Artificial Intelligence, pages 674–679, 1981. (Cited on page 43.)
- [Mahlisch 2006] M. Mahlisch, R. Schweiger, W. Ritter and K Dietmayer. *Sensor-fusion Using Spatio-Temporal Aligned Video and Lidar for Improved Vehicle Detection*. In IEEE Intelligent Vehicles Symposium, pages 424–429, 2006. (Cited on page 92.)
- [Markovsky 2007] Ivan Markovsky and Sabine Van Huffel. *Overview of total least-squares methods*. Journal of Signal Processing, Elsevier, vol. 87, pages 2283–2302, 2007. (Cited on page 23.)

- [Marquardt 1963] D. W. Marquardt. *An algorithm for least-squares estimation of nonlinear parameters*. Journal of the Society for Industrial and Applied Mathematics, vol. 11, pages 431–441, 1963. (Cited on pages 26 and 27.)
- [Matas 2002] J. Matas, O. Chum and M. Urban. *Robust wide basebase stereo from maximally stable Extremal regions*. In British Machine Vision Conference, 2002. (Cited on page 30.)
- [Matthies 1987] L. Matthies and S. Shafer. *Error modeling in stereo navigation*. IEEE journal of Robotics and Automation, vol. 3, pages 239–248, 1987. (Cited on page 48.)
- [Matthies 1989] L. Matthies. *Dynamic Stereo Vision*. PhD thesis, Carnegie Mellon University, 1989. (Cited on page 48.)
- [McCarthy 2004] C. McCarthy and N. Barnes. *Performance of optical flow technique for indoor Navigation with a mobile robot*. In in Proceedings of IEEE International Conference on Robotics and Automation, 2004. (Cited on page 48.)
- [Mikolajczyk 2005] Krystian Mikolajczyk and Cordelia Schmid. *A Performance Evaluation of Local Descriptors*. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 27, pages 1615–1630, 2005. (Cited on page 57.)
- [Miroslav 1999] Kulich Miroslav and Preucil Libor. *Knowledge Acquisition for Mobile Robot Environment Mapping*. In Database and Expert Systems Applications, volume 1677 of *Lecture Notes in Computer Science*, pages 123–134. Springer Berlin Heidelberg, 1999. (Cited on page 118.)
- [Moras 2011] Julien Moras, Véronique Cherfaoui and Philippe Bonnifait. *Credibilist occupancy grids for vehicle perception in dynamic environments*. In In Proc. IEEE International Conference on Robotics and Automation, 2011. (Cited on page 142.)
- [Moravec 1979] H. Moravec. *Visual mapping by a robot rover*. In in Proceedings of the 6th international joint conference on Artificial Intelligence, 1979. (Cited on page 31.)
- [Moravec 1980] H. Moravec. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot*. PhD thesis, Stanford University, 1980. (Cited on pages 48 and 49.)
- [Moravec 1985] H. Moravec and A. Elfes. *High resolution maps from wide angle sonar*. In In Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)., volume 2, pages 116–121, Mar. 1985. (Cited on pages 118, 119 and 136.)
- [Murray 2000] Don Murray and James J. Little. *Using Real-Time Stereo Vision for Mobile Robot Navigation*. Autonomous Robots, vol. 8, pages 161–171, April 2000. (Cited on page 120.)

- [Nedevschi 2007] S. Nedevschi and R. Danescu. *A Sensor for Urban Driving Assistance Systems Based on Dense Stereovision*. In Intelligent Vehicles Symposium, 2007 IEEE, pages 276–283, june 2007. (Cited on page 80.)
- [Nestares 2000] Oscar Nestares, David Fleet and David Heeger. *Likelihood Function and Confidence Bounds for Total-Least-Squares Problems*. In IEEE Conference on Computer Vision and Pattern Recognition, 2000. (Cited on page 24.)
- [Nguyen 2012] Thien-Nghia Nguyen, Bernd Michaelis and Al-Hamadi. *Stereo-Camera-Based Urban Environment Perception Using Occupancy Grid and Object Tracking*. IEEE Transactions on Intelligent Transportation Systems, vol. 13, pages 154–165, March 2012. (Cited on pages 117, 120, 123 and 130.)
- [Nister 2004] D. Nister. *Visual odometry*. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. (Cited on page 49.)
- [Nunez 2009] P. Nunez, P. Drews Jr, R. Rocha and J Dias. *Data Fusion Calibration for a 3D Laser Range Finder and a Camera using Inertial Data*. In Proc. of 4th European Conf. on Mobile Robots, pages 31–36, Sep. 2009. (Cited on page 92.)
- [Olson 2000] C. Olson, L. Matthies and M. Schoppers. *Robust stereo ego-motion for long distance navigation*. In IEEE Conference on Computer Vision and Pattern Recognition, 2000. (Cited on page 48.)
- [Perrollaz 2006] M. Perrollaz, R. Labayarde, C. Royere, N. Hautiere and D. Aubert. *Long range obstacle detection using laser scanner and stereo vision*. In IEEE Intelligent Vehicles Symposium, volume 1, pages 182–187, 2006. (Cited on page 92.)
- [Perrollaz 2010] Mathias Perrollaz, Yoder John-David, Spalanzani Anne and Christian Laugier. *Using the disparity space to compute occupancy grids from stereo-vision*. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2010. (Cited on pages 120, 123 and 130.)
- [Petrovskaya 2009] Anna Petrovskaya and Sebastian Thrun. *Model Based Vehicle Detection and Tracking for Autonomous Urban Driving*. Journal of Autonomous Robots, Springer, vol. 26, pages 123–139, 2009. (Cited on page 3.)
- [Plagemann 2007] Christian Plagemann, Kristian Kersting, Patrick Pfaff and Wolfram Burgard. *Gaussian beam processes: A nonparametric bayesian measurement model for range finders*. In In Proc. of Robotics: Science and Systems (RSS), 2007. (Cited on page 136.)
- [Rabe 2007a] C Rabe. *Fast detection of moving objects in complex scenarios*. In IEEE Intelligent Vehicles Symposium, 2007. (Cited on page 72.)

- [Rabe 2007b] C. Rabe, U. Franke and S. Gehrig. *Fast detection of moving objects in complex scenarios*. In Intelligent Vehicles Symposium, IEEE, pages 398–403, june 2007. (Cited on page 80.)
- [Rao 2008] C. Rao, H. Toutenburg, H. Shalabh and C. Heumann. *Least squares and alternatives*. Springer, 2008. (Cited on pages 19 and 20.)
- [Rodriguez F 2008] S.A. Rodriguez F, V. Fremont and P. Bonnifait. *Extrinsic calibration between a multi-layer lidar and a camera*. In IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, pages 214–219, aug. 2008. (Cited on page 92.)
- [Rodriguez F 2009] S.A Rodriguez F, V Fremont and P Bonnifait. *An experiment of a 3D real-time robust visual odometry for intelligent vehicles*. In Intelligent Transportation Systems, 12th International IEEE Conference on, pages 1–6, 4-7 Oct. 2009. (Cited on page 80.)
- [Rosin 1999] P. L. Rosin. *Measuring corner properties*. Computer Vision and Image Understanding, vol. 73, pages 291–307, 1999. (Cited on page 41.)
- [Rosten 2006] Edward Rosten and Tom Drummond. *Machine learning for high-speed corner detection*. In In Proc. of European Conference on Computer Vision, 2006. (Cited on pages 34 and 57.)
- [Rostern 2005] E. Rostern and T. Drummond. *Fusing ppoint and lline for high performance tracking*. In in Proceedings of the IEEE international Conference on Computer VSION, pages 1508–1511, 2005. (Cited on pages 34 and 57.)
- [Ruble 2011] E Rublee. *ORB: An efficient alternative to SIFT or SURF*. In In Proc. of IEEE International Conference on Computer Vision, 2011. (Cited on pages 41 and 57.)
- [Sawhney 2000] H. Sawhney. *Independent motion detection in 3D scenes*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, pages 1191–1199, 2000. (Cited on page 72.)
- [Scaramuzza 2011] D. Scaramuzza. *Visual Odoemtry [Tutorial]*. IEEE Robotics & Automation Magazine, vol. 18, pages 80–92, 2011. (Cited on page 54.)
- [Schmid 1997] C. Schmid and R. Mohr. *Local Greyvalue Invariant for Image Retrieval*. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 19, pages 530–535, 1997. (Cited on page 30.)
- [Schölkopf 1998] Bernhard Schölkopf, Smola Alexander and Klaus-Robert Müller. *Nonlinear component analysis as a kernel eigenvalue problem*. Neural Computation, vol. 10, pages 1299–1319, July 1998. (Cited on page 81.)

- [Se 2002] S. Se, D. Lowe and J. Little. *Mobile robot localization and mapping with uncertainty using Scale-Invariant visual landmarks*. International Journal of Robotics Research, vol. 21, pages 735–738, 2002. (Cited on page 30.)
- [Shi 1994] J. Shi and C. Tomasi. *Good feature to track*. In in Proc. of the Conference on Computer Vision and Pattern Recognition, 1994. (Cited on pages 30, 32, 43 and 57.)
- [Sivic 2005] J. Sivic, B. Russel, A. Efros and A. Zisserman. *Discovering objects and their location in images*. In International Conference on Computer Vision, 2005. (Cited on page 30.)
- [Soquet 2007] Nicolas Soquet, Didier Aubert and Nicolas Hautiere. *Road Segmentation Supervised by an Extended V-Disparity Algorithm for Autonomous Navigation*. In IEEE Intelligent Vehicle Symposium, pages 160–165, June 2007. (Cited on page 73.)
- [Srinivasa 2002] N. Srinivasa. *Vision-based vehicle detection and tracking method for forward collision warning in automobiles*. In IEEE Intelligent Vehicle Symposium, 2002., volume 2, pages 626–631, 2002. (Cited on page 3.)
- [Stauffer 2000] C. Stauffer and W. Grimson. *Learning patterns of activity using real time tracking*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, pages 747–757, 2000. (Cited on page 72.)
- [Stepan 2005] Petr Stepan, Miroslav Kulich and Libor Preucil. *Robust Data Fusion With Occupancy Grid*. IEEE Transactions on Systems Man and Cybernetics, Part C: Applications and Reviews, vol. 35, pages 106–115, 2005. (Cited on page 142.)
- [Streller 2002] D. Streller, K. Furstenberg and K. Dietmayer. *Vehicle and object models for robust tracking in traffic scenes using laser range images*. In The IEEE 5th International Conference on Intelligent Transportation Systems, pages 118–123, 2002. (Cited on page 4.)
- [Strutz 2010] T. Strutz. *Data fitting and uncertainty: A practical introduction to weighted least squares and beyond*. Vieweg+Teubner, 2010. (Cited on page 21.)
- [Sun 2002] J. Sun. *Stereo Matching Using Belief Propagation*. In in Proceedings of European Conference of Computer Vision, 2002. (Cited on page 73.)
- [Thompson 1990] W. Thompson and T. Pong. *Detecting Moving Objects*. International Journal of Computer Vision, vol. 4, pages 39–57, 1990. (Cited on page 72.)
- [Thrun 2003] Sebastian Thrun. *Learning Occupancy Grid Maps with Forward Sensor Models*. Autonomous Robots, vol. 15, pages 111–127, Sep. 2003. (Cited on pages 118, 119, 120 and 136.)

- [Thrun. 2004] S. Thrun., Y. Liu and Y. Andrew. *Simultaneous Localization and Mapping with Sparse Extended Information Filters*. International Journal of Robotics Research, vol. 23, pages 693–716, 2004. (Cited on page 118.)
- [Thrun 2005] S. Thrun, W. Burgard. and D. Fox. Probabilistic robotics. MIT Press, 2005. (Cited on page 118.)
- [Tomasi 1991] Carlo Tomasi and Takeo Kanade. *Detection and Tracking of Point Features*. Rapport technique, Carnegie Mellon University Technical Report CMU-CS-91-132, 1991. (Cited on page 43.)
- [Toyama 1999] K. Toyama, J. Krumm and B. Brumitt. *Wallflower: principles and practice of background maintenance*. In In Proceedings of the 7th IEEE International Conference on Computer Vision, 1999. (Cited on page 72.)
- [Tuytelaars. 2000] T. Tuytelaars. and L. Van Gool. *Wide basebase stereo matching based on local affine invariant regions*. In British Machine Vision Conference, 2000. (Cited on page 30.)
- [Tuytelaars 2008] T Tuytelaars and K Mikolajczyk. *Local Invariant Feature Detectors: A Survey*. Foundations and Trends in Computer Graphics and Vision, pages 177–280, 2008. (Cited on pages 30, 57 and 58.)
- [Vasconcelos 2012] Francisco Vasconcelos and Urbano Nunes. *A Minimal Solution for the Extrinsic Calibration of a Camera and a Laser-Rangefinder*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 99, 2012. (Cited on page 93.)
- [Viola 2001] P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. In in Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2001. (Cited on page 38.)
- [Viola 2004] Paul Viola and Michael Jones. *Robust real-time face detection*. International Journal of Computer Vision, vol. 57, pages 137–154, 2004. (Cited on page 80.)
- [Wall 2012] E. Wall, A. Rechtsteiner and L. M. Rocha. *Singular Value Decomposition and Principal Component Analysis*. <http://public.lanl.gov/mewall/kluwer2002.html>, 2012. (Cited on page 99.)
- [Wang 2006] Jia Wang and Zhencheng Hu. *Motion Detection in Driving Environment Using U-V-Disparity*. In 7th Asian Conference on Computer Vision, pages 307–316. Springer, 2006. (Cited on page 73.)
- [Wedel 2009] Andreas Wedel, Clemens Rabe and Uwe Franke. *Detection and Segmentation of Independently Moving Objects from Dense Scene Flow*. In Proceedings of the 7th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition, 2009. (Cited on page 72.)



- [Welch 1995] G. Welch and G. Bishop. *An Introduction to the Kalman Filter*, 1995. (Cited on page 126.)
- [wik 2012] *Rotations in three dimesions*. [http://en.wikipedia.org/wiki/Rotation\\_matrix](http://en.wikipedia.org/wiki/Rotation_matrix), 2012. (Cited on page 109.)
- [Wren 1997] C. Wren, A. Azarbayejani and D. Darrel. *Pnder: Real-time tracking of the human body*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, pages 780–785, 1997. (Cited on page 72.)
- [Ye 2002] Cang Ye. *Characterization of a 2-D Laser Scanner for Mobile Robot Obstacle Negotiation*. In In IEEE International Conference on Robotics and Automation, pages 2512–2518, 2002. (Cited on page 100.)
- [Zhang 2000] Z. Zhang. *A flexible new technique for camera calibration*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, pages 1330–1334, 2000. (Cited on pages 14 and 96.)
- [Zhang 2004] Qilong Zhang and R. Pless. *Extrinsic calibration of a camera and laser range finder*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, volume 3, pages 2301–2306, sept.-2 oct. 2004. (Cited on pages 92, 93, 94, 100, 104, 105, 107, 108, 110 and 111.)
- [Zhu 2010] Xiaolong Zhu, Huijing Zhao and Yiming Liu. *Segmentation and clas-sification of Range Image from an Intelligent Vehicle in Urban Environ-ment*. In IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS), volume 6, pages 1457 – 1462, 18-22 Oct 2010. (Cited on pages 80, 81, 82, 85, 86 and 87.)



## Résumé :

Les systèmes de perception, qui sont à la base du concept du véhicule intelligent, doivent répondre à des critères de performance à plusieurs niveaux afin d'assurer des fonctions d'aide à la conduite et/ou de conduite autonome. Les travaux de cette thèse concernent le développement d'un système de perception à base d'un capteur de vision stéréoscopique et d'un capteur lidar pour l'analyse de scènes dynamiques en environnement urbain. Les travaux présentés sont divisés en quatre parties. La première partie présente une méthode d'odométrie visuelle basée sur la stéréovision, avec une comparaison de différents détecteurs de primitives et différentes méthodes d'association de ces primitives. Un couple de détecteur et de méthode d'association de primitives a été sélectionné sur la base d'évaluation de performances à base de plusieurs critères. Dans la deuxième partie, les objets en mouvement sont détectés et segmentés en utilisant les résultats d'odométrie visuelle et l'image U-disparité. Ensuite, des primitives spatiales sont extraites avec une méthode basée sur la technique KPCA et des classifieurs sont enfin entraînés pour reconnaître les objets en mouvement (piétons, cyclistes, véhicules). La troisième partie est consacrée au calibrage extrinsèque d'un capteur stéréoscopique et d'un Lidar. La méthode de calibrage proposée, qui utilise une mire plane, est basée sur l'exploitation d'une relation géométrique entre les caméras du capteur stéréoscopique. Pour une meilleure robustesse, cette méthode intègre un modèle de bruit capteur et un processus d'optimisation basé sur la distance de Mahalanobis. La dernière partie de cette thèse présente une méthode de construction d'une grille d'occupation dynamique en utilisant la reconstruction 3D de l'environnement, obtenue des données de stéréovision et Lidar de manière séparée puis conjointement. Pour une meilleure précision, l'angle entre le plan de la chaussée et le capteur stéréoscopique est estimé. Les résultats de détection et de reconnaissance (issus des première et deuxième parties) sont incorporés dans la grille d'occupation pour lui associer des connaissances sémantiques. Toutes les méthodes présentées dans cette thèse sont testées et évaluées avec la simulation et avec des données réelles acquises avec la plateforme expérimentale véhicule intelligent SetCar<sup>®</sup> du laboratoire IRTES-SET.

**Mots-clés :** Véhicule intelligent, Odométrie visuelle, Détection et reconnaissance d'objets en mouvement, Calibrage extrinsèque entre caméras et Lidar, Grille d'occupation dynamique.

## Abstract:

Intelligent vehicles require perception systems with high performances. The works presented in this Ph.D thesis concern several topics on cameras and lidar based perception for understanding dynamic scenes in urban environments. The works are composed of four parts. In the first part, a stereo vision based visual odometry is proposed by comparing several different approaches of image feature detection and feature points association. After a comprehensive comparison, a suitable feature detector and a feature points association approach is selected to achieve better performance of stereo visual odometry. In the second part, independent moving objects are detected and segmented by the results of visual odometry and U-disparity image. Then, spatial features are extracted by a kernel-PCA method and classifiers are trained based on these spatial features to recognize different types of common moving objects e.g. pedestrians, vehicles and cyclists. In the third part, an extrinsic calibration method between a 2D lidar and a stereoscopic system is proposed. This method solves the problem of extrinsic calibration by placing a common calibration chessboard in front of the stereoscopic system and 2D lidar, and by considering the geometric relationship between the cameras of the stereoscopic system. This calibration method integrates also sensor noise models and Mahalanobis distance optimization for more robustness. At last, dynamic occupancy grid mapping is proposed by 3D reconstruction of the environment, obtained from stereovision and Lidar data separately and then conjointly. An improved occupancy grid map is obtained by estimating the pitch angle between ground plane and the stereoscopic system. The moving object detection and recognition results (from the first and second parts) are incorporated into the occupancy grid map to augment the semantic meanings. All the proposed and developed methods are tested and evaluated with simulation and real data acquired by the experimental platform "intelligent vehicle SetCar<sup>®</sup>" of IRTES-SET laboratory.

**Keywords:** Intelligent vehicle, Visual odometry, Moving objects detection and recognition, Cameras and lidar Extrinsic calibration, Dynamic occupancy grid mapping.

