DEVELOPMENT OF AUTONOMOUS FEATURES AND INDOOR LOCALIZATION TECHNIQUES FOR CAR-LIKE MOBILE ROBOTS

Dem Fachbereich Elektrotechnik und Informatik der
Universität Siegen

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

genehmigte Dissertation
von

M.Sc. Niramon Ruangpayoongsak

1. Gutachter: Prof. Dr.-Ing. Hubert Roth
2. Gutachter: Prof. Dr.-Ing. Rudolf Schwarte
Vorsitzender: Prof. Dr.-Ing. Robert Mayr

Tag der mündlichen Prüfung: 10.8.2006

urn:nbn:de:hbz:467-2447

Acknowledgements

I would gratefully like to thank my advisor Prof. Hubert Roth. I thank him for providing the long period funding from the DAAD and giving me the opportunity to work in a European commission project; Building Presence through Localization for Hybrid Telematic Systems (PeLoTe). Also, I thank him for giving me a chance to work on the new technology PMD camera and giving me the freedom to develop my own ideas. I am grateful for his support and for the discussions we had.

I would like to thank my second supervisor Prof. Rudolf Schwarte for providing us the 16x16 pixel and the 48x64 pixel PMD camera. I thank him for his discussion, contribution and comments for this thesis. Also, I thank him for providing me funding during the last period of my dissertation work. I would like to thank Prof. Robert Mayr for discussing the development of and contributing to the development of the nonlinear dynamic car model, and for valuable feedback on our results.

I would like to give special thanks to our electronics man, Mr. Werner Utsch, and to a PMD Technology GmbH member, Mr. Markus Grothof. I am grateful to both for their support and for the discussions we had. I am most grateful for friendship and help from Mr. Jörg Kuhle and all colleagues.

Lastly, I would like to thank my parents for always being there for me and supporting me. I gratefully thank them and appreciate their love, sacrifice, patience, advice, and help.

Contents

List of Figures

List of Tables

Abstract

Intelligent autonomous navigation in a large-scale and unknown indoor environment is an important problem in mobile robotics. For a car-like mobile robot with a racing car platform, the movement control concept is similar to that of a car.

The autonomous features enable robots to control own motion without human interference. Three autonomous features are addressed in this thesis; obstacle avoidance, doing 180° turns in a narrow corridor, and path following control. Since the obstacle positions are not known beforehand, the strategy requires not only the obstacle avoidance but also trajectory generation and robot localization.

The robot localization can be broken down into relative and absolute localization. This thesis addresses the development of the model-based relative localization technique and the landmark-based absolute localization technique.

The model-based relative localization is applied by the non-linear dynamic car model to the Kalman filter. The study of integrating sensor data from odometer, gyroscope and compass for the position and heading estimators provides a discussion of the performance of three localization methods; differential drive, gyroscope estimator, and compass estimator.

The landmark-based absolute localization is applied by using the 3D camera and the 3D artificial landmark and is called the position calibration. Three parts of the position calibration are developed: The design of landmarks, the landmark recognition, and the robot position prediction and update. Lastly, the improvement for the resolution of the position calibration by using 2D and 3D images is studied.

Kurzfassung

Intelligente autonome Navigation ist ein wesentliches Problem in der mobilen Robotik. Für einen modellbasierten Fahrzeug-ähnlichen mobilen Roboter ist das Steuerungskonzept ähnlich wie beim Auto.

Autonome Feature erlauben Robotern eigene Bewegungen zu kontrollieren ohne menschliche Interaktion. Drei autonome Feature werden in diese Arbeit behandelt: Hindernisvermeidung, 180° Drehung in einem schmalen Flur und Pfadverfolgung. Weil die Hindernispositionen unbekannt sind, erfordert die Strategie nicht nur Hindernisvermeidung, sondern auch Bahnplanung und Roboterlokalisierung.

Die Roboterlokalisierung kann in relative und absolute Lokalisierungen unterteilt werden. In dieser Arbeit soll die Entwicklung der modellbasierten relativen Lokalisierungstechnik und der absoluten Lokalisierung durch Landmarken untersucht werden.

Die Entwicklung der modellbasierten relativen Lokalisierung wird durch ein nichtlineares dynamisches Automodell mit nachfolgendem Kalman Filter erreicht. Die Integration der Sensordaten des Entfernungsmessers, Trägheitsgyroskop-Sensors und der Kompass-Sensoren durch den Kalman Filter ermöglicht die Analyse der Leistung der drei Positionierungsmethoden; durch differentiellen Antrieb, Gyroskops- und Kompass-Abschätzung.

Die absolute Lokalisierung wird durch den Einsatz einer 3D-Kamera und 3D-Landmarken erreicht und wird im Folgen der Positionskalibrierung genant. Drei Teile der Positionskalibrierung werden entwickelt: Das Design der Landmarken, die Erkennung der Landmarken sowie die Voraussage und die Aktualisierung der Roboterposition. Schließlich wird die Verbesserung der Auflösung der Positionskalibrierungstechnik durch 2D und 3D Bilder untersucht.

1 Introduction

Developed as an inexpensive self-design concept, the Mobile Experimental Robot for Locomotion and Intelligent Navigation (MERLIN) was constructed and exploited as a test bed for control algorithms and tele-operation for education [KLA 02] [KUH 04]. MERLIN robots are robust for indoor, outdoor, also in rough terrains environment. In this work, the autonomous features and robot localization techniques for MERLIN are developed.

1.1 Problem Specification

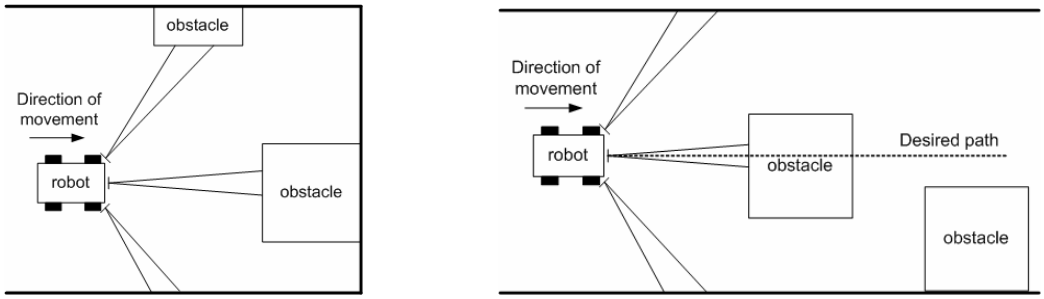
This thesis addresses the problems of both the autonomous features and the robot localization for a car-like mobile robot in a large scale and unknown indoor environment.

1.1.1 Autonomous features

Two important factors for designing the algorithms for autonomous features are robot motion and perception. Since robots have the same manoeuvring as a car and have obstacle detection sensors on board. The problems of autonomous features are set as follows:

- Avoiding obstacle collisions. Obstacle collision avoidance is an important feature for preventing damage during navigation. By using the range sensors for obstacle detection, the robot perception is shown in Figure 1.1. The algorithm must be designed such that the robot overcomes the obstacle without crashing and finally comes to a free area.
- Turning 180° in a narrow corridor. When the robot is driving along a narrow corridor with a dead end as shown in Figure 1.1a, the robot should try to turn 180° into the opposite direction. The designed algorithm must provide robustness in the case of existence of unknown obstacles.
- Path following. The autonomous path following problem in an unknown environment is that the robot should follow a user-specified path and also automatically avoid collision with obstacles. The situation is illustrated in Figure 1.1b below, where the obstacle is on the desired path. The task

of autonomous path following control is changing robot's orientation to avoid collision and to finally reach the destination.



(a) (b)

Figure 1.1: Different scenarios: (a) narrow corridor with a dead end; (b) obstacles on the desired path.

1.1.2 Localization techniques for mobile robots

The robot localization can be classified into two main categories [GOE 99]:

- Relative (local) localization: The technique is to localize the robot position and orientation by using various on board sensors such as odometer, gyroscope, etc. The robot position and heading relative to its start position. This is also called the self-localization.
- Absolute (global) localization: The technique is to obtain absolute position using positioning system such as beacons, landmarks, GPS, etc. The absolute robot position and heading is not relative to its start position.

This thesis classifies the problem of robot localization in the same way as described above.

1.1.2.1 The relative localization techniques. In a large area, when the structure of the building is not known beforehand and the absolute positioning system is not available, the robot needs to rely on its self-localization. Differential drive positioning is the most basic type of the relative localization that uses only the distances information from the odometers. However, slippage of the car's wheels cause accumulated errors. An example of the error that occurs in differential drive positioning is shown in Figure 1.2. The estimated robot's heading from the differential drive contains accumulated errors that result in a large error in the estimated final position. The improvement is to

use an exact relative localization technique, such as the model based localization. Regarding the car-like motion, the nonlinear dynamic model of the car is selected as a robot model and the discrete extended Kalman filter is applied as the robot position and heading estimator. Therefore, this thesis addresses the problem of relative localization for a car-like mobile robot by using the nonlinear dynamic model and discrete extended Kalman filter.

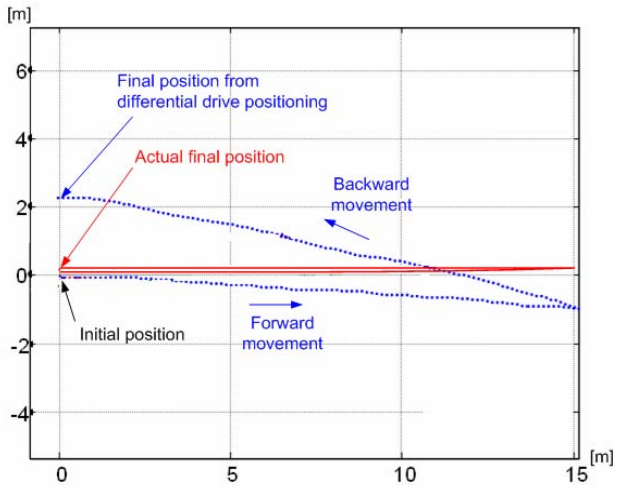


Figure 1.2: The robot position from differential drive positioning.

1.1.2.2 The absolute localization techniques. There are many techniques for absolute localization and one of them is landmark based localization [BOR 96]. Two types of landmarks are:

- Natural landmark: These landmarks are existing objects in the environment. Examples of natural landmarks in indoor environment are doors, tables and chairs.
- Artificial landmark: These landmarks are designed, built, and placed in the environment. The shape and color of landmark designs are usually depending on the perception of robot.

The cameras are often applied with the vision-based localization technique using artificial landmarks. The 2D camera captures color or gray images whereas the 3D camera provides the depth or the range images. The Photonic Mixer Device (PMD) camera is a 3D camera available in the market. This thesis addresses additionally the problem of applying the 3D camera to solve the artificial landmark based localization problem. It is challenging work to detect and recognize the landmarks. The questions of

how to use the 3D camera to detect the landmark, what the landmark should look like, and what information could be obtained from the landmark need to be answered. This landmark based localization technique is called the position calibration.

1.3 An overview of the thesis

This thesis is divided into five main chapters. The coming chapter provides a survey of the related work in mobile robot localization techniques. Next, Chapter 3 provides an overview of the mobile robot and its user interface. The different robot prototypes, sensor measurements, the wireless data communication, the graphic user interfaces, the steering control and the speed control are explained. Chapter 4 describes our solutions to the problems of autonomous features; obstacle avoidance, turning 180° in a narrow corridor, and path following control. Chapter 5 then explains the use of the nonlinear dynamic car model and the discrete extended Kalman filter for the relative localization of the robot. Chapter 6 describes in detail the innovative use of a 3D 16x16 pixel PMD camera and 3D artificial landmark for mobile robot localization and presents the results of the on-line experiment of the position calibration. Chapter 7 presents the improvement for the resolution of the position calibration by using images from the 3D 48x64 pixel PMD camera and a web camera. The final chapter, Chapter 8, provides a summary and perspective.

2 A Survey of Related Works

The survey of related work focuses on the development of robot localization techniques. The related works are divided into robot localization techniques in indoor environment, applications of model-based techniques for car-like mobile robots and absolute localization techniques based on artificial landmarks.

2.1 Robot localization techniques in indoor environment.

Several techniques using the simultaneous localization and mapping (SLAM) exist for localization in an indoor environment [FOX 99] [CHO 01] [ROE 03] [TRE 04] [SIM 05]. The SLAM techniques provide at the same time the robot position and the generating map. This technique requires that the user prepares a part of the known map before the robot starts navigation. During navigation, the robot finds its position on this map by matching its perception data of the environment with the partially known map. In some applications, the map is not known beforehand, e.g. in search and rescue mission. By using SLAM techniques, when the known map is not available, the accuracy of the robot position and the quality of the map are therefore depending on the quality of the robot localization technique.

Besides, Goel [GOE 99] succeeded in implementing the integration of the relative and absolute localization techniques for both an indoor area and an outdoor area. In an indoor environment, the odometer and gyroscope provide inertial measurements for kinematic model based localization. Though this results in a much improved estimated position compared to using only the well calibrated odometer, the kinematic model in his work is specifically applicable only for two-wheel robots.

Though there are various localization techniques existing on the different robot platforms. These developed techniques are depending not only on onboard sensor but also a robot's movement model and environments. The improvements of these techniques are still under research.

2.2 Applications of model-based techniques for car-like mobile robots.

For car-like mobile robots, the kinematic model is often applied for solving motion control problems. These research topics include a finite-dimensional iterative learning controller for steering [FER 96], a path following lateral controller [MEL 02],

and a switching controller for the position control [USH 02]. Though the kinematic model provides exact representation of the robot's movement, the model does not provide a representation of the non-linear characteristic of the car driving motion like the dynamic model.

This thesis develops a non-linear dynamic car model for the car-like robot localization. The discrete extended Kalman filter is applied as the robot position estimators. Two estimations are obtained by using the yaw angle from the gyroscope and from compass sensor and are compared with the existing differential drive technique.

2.3 Absolute localization techniques based on artificial landmarks.

The PMD camera was built and developed at the University of Siegen [SCH 04]. The application of the emerging technology PMD camera to the MERLIN prototype results in new application for mobile robotics. The following survey of the literature is divided into two parts: the related work on vision-based localization and the artificial landmark, and comparisons of the 3D camera to other existing range sensors in mobile robotics.

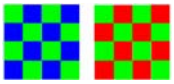
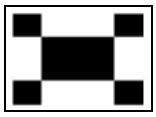
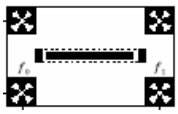
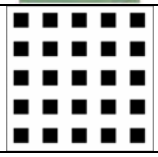
2.3.1 Vision-based localization and the artificial landmark.

In the field of vision-based localization, the existing recognition techniques are applied using a CCD camera and 2D artificial landmark. The different landmarks are designed by using color and shape patterns as distinguishing criteria. The recognition techniques to be exploited depend on the individual landmarks as summarized in Table 2.1. These recognition techniques rely on a clear definition of the landmark pattern using contrasts in color. These landmarks are designed by using high contrast colors, such as green and red or black and white. However, the contrast in color varies in different light conditions. In a low visibility area, the brightness of the light is weak and the contrast is insignificant. Therefore, different light sources such as neon light or spotlights change the extent of the contrast between colors.

The PMD camera provides better results than the CCD camera in this problem area, since the PMD camera is developed such that measurement is independent of light conditions indoors, and even in low visibility light conditions. Moreover, depth information can be obtained directly from the camera without any need for image processing. This thesis therefore proposes a step forward in mobile robotics research

using the 3D vision available through the PMD cameras for robot localization paired with and 3D artificial landmarks.

Table 2.1: A summary of artificial landmarks and their recognition techniques.

Landmark pictures	Description of the landmark	Recognition techniques
	Made of color patterns that are arranged in symmetrical and repeating color patches.	Tracked using the color histogram technique [YOO 02]
	A simple shape pattern that pairs colors	Tracked using the probability density in the condensation algorithm [JAN 02]
	Made of black and white colors in a rectangle pattern	Tracked by using the stereo depth information and the end point closed loop system [HAN 02]
	Made of black and white colors with a pattern of symmetrical shapes.	Tracked using the Laplacian sign of the Gaussian (SLOG) filters, fast Fourier transformation (FFT) and a vector quantization (VQ) neural net [FUC 04]
	Made of contrasting colors in a simple pattern	Tracked using the Retinex algorithm [MAY 02]
	Made of black and white colors with a rectangular pattern.	Tracks the geometric transformation [ZHA 04]

2.3.2 The PMD camera in comparison with other range sensors.

In mobile robotics, there are many types of range sensors that provide distance measurements. They are listed as follows:

- Sonar (ultrasonic) sensors
- Infrared sensors
- LED range finders
- Laser range finders
- Stereo vision using CCD cameras
- 3D vision using a Laser range finder.

Sonar and infrared sensors measure a single distance value, but provide no depth image output. The LED range finder and the laser range finder provide measurements for one-dimensional scanned distances at the height level of the sensor mounting position and the output is the distance value at each scan step within a vision range of 0-180 degrees. However, these sensors do not provide 3D vision. When compared to the

stereo vision that results from using CCD cameras, the PMD camera provides a more convenient way of obtaining a ready-to-use depth image. The current method of obtaining 3D vision is to use a laser range finder with an additional vertical scanning mechanism [SUR 03]. This method requires a high speed image processor and computational algorithms and additional mechanic for vertical scan of the laser range finder. However, the laser range finder is much heavier and bigger than the PMD camera.

Beyond its advantages when compared with these range sensors, the PMD camera is a promising device for the range image application in mobile robotics because it has compact size and light weight and it provides ready-to-use output.

3 The Mobile Experimental Robots for Locomotion and Intelligent Navigation (MERLIN)

The following sections present an overview of MERLIN. Section 3.1 introduces the series of MERLIN prototypes. Sensors and devices are added to the first prototype, which is later changed by introducing a new chassis. Section 3.2 explains MERLIN's wireless data communication structures for tele-operated control. Section 3.3 explains the characteristics and specification of onboard sensors and their data processing. The final section explains the robot steering and speed control.

3.1 Series of MERLIN prototypes

The Mobile Experimental Robots for Locomotion and Intelligent Navigation (MERLIN) series of micro-robots is designed for a broad spectrum of indoor and outdoor tasks using standardized functional modules like sensors, actuators and communication by radio link and wireless LAN. Several of the MERLIN prototypes are constructed and adapted for experimental environments. The first prototype is designed to test sensor data, wireless radio communication protocol, and control algorithms such as speed control and steering control. A later prototype is adapted to use a 3D camera and laser scanner, and the PC104 with wireless LAN. The later prototype requires a bigger chassis to carry the heavier load and allows for more power consumption. This section also discusses the problems encountered when changing the chassis and provides the solutions found.

3.1.1 The first MERLIN prototype

MERLIN was first adapted from a 1/10 scale Compagnucci racing model car as shown in Figure 3.1. The steering and propelling motors are already mounted by the manufacturer. The platform was modified for the installation of the Infineon 80C167CR microcontroller and sensors for navigation. Appendix A provides the specification and developed software structure in the microcontroller.

The robot measures 40 cm x 50 cm x 20 cm and weighs 5 kg including batteries. The sensors on board and their individual outputs are listed in Table 3.1. A magnetic 3-axis compass sensor provides the absolute angle for roll, pitch, and yaw. Four ultrasonic sensors are mounted to detect obstacle distance. They are located on the front left, on

the front right, on the front middle and at the rear of the robot's body. A gyroscope is mounted at the center of the robot's body for angular velocity measurement. Bumpers are mounted behind the crash protection plate under the ultrasonic sensor position for crash detection. Odometers on the front left and on the front right wheel are hall sensors with 8 magnets for measurement of the driving distances. A radio transceiver is also mounted on the top to enable wireless communication and control, and is set at the transmission rate of 9600 Baud. The maximum length of its data packet is 27 bytes. The radio transceiver covers an area of 30 meters indoors, and 120 meters on open ground. Two motors are equipped as steering and driving motors: The steering motor is a servomotor that steers the front wheels and the driving motor propels the rear wheels. There are two battery sets attached, 12V for the electronics and steering motor and 7V for the driving motor.

Table 3.1: Sensors on board and their outputs.

Sensors	Outputs
3-axis Compass Sensors	Yaw, Pitch and Roll angles
Ultrasonic sensors	Distance to obstacles
Gyroscope	Angular velocity
Bumper	Crash detection
Odometer(Hall Sensors)	Distance driven and speed

3.1.2 The development of MERLIN on the original chassis

MERLIN was further developed on its existing platform for hardware and software interface with a 3D PMD camera. The camera measures the distances from the target and provides distance value output on a 16x16 pixel image. Details of the PMD camera are provided in section 3.3.6.

The PMD camera can be equipped as an on-line camera by using the PC104. The PC104 is a single board embedded PC with the size of a floppy disk. The up to date PC104 on the market provides high speed communication via wireless LAN and fast computation. Our PC104 is the CPU-M1 from EEPD. Details of the CPU-M1 are provided in Appendix C. The exploitation of the PC104 together with the existing microcontroller on MERLIN has the following advantages:

- The microcontroller interfaces with the low level hardware layer such as pulse width modulation (PWM) signal generation, A/D conversion, or the interruption of routines. Therefore, the PC104 I/O is available for other extra devices, which may come.

- The microcontroller also interfaces with PC104 for high speed data transmission via RS232. As the results, the bigger data packets are available and we can therefore put more sensor data into the packets.
- PC104 passes the image and sensor data to a client PC via high speed wireless LAN in real time.
- Image and sensor data can be saved on the PC104 hard disk and later used for off-line data processing.

Figures 3.2a show MERLIN#2 after the installation of the devices on board. The PC104, PMD camera, and the Orinoco Client Gold USB WLAN are mounted on the first MERLIN prototype. The sensors and devices are shown in Figure 3.2b. All sensors are fitted in the same place except the compass sensor. The compass is now mounted on the top of the robot's body for reduction of the electromagnetic noise from the motor and batteries. This enables better measurement. After the PC104 has been added, the interfaces for keyboard, mouse, monitor, Ethernet are also present. The voltage regulator is also mounted to provide the 5 V and the maximum of 8 A needed for the operation of the PC104 and Harddisk. This new structure MERLIN measures 40 cm x 50 cm x 61 cm and weights 7 kg including batteries. The robot now weights much more than earlier and this causes a problem in steering and propelling. The bigger robot platform is therefore necessary.

3.1.3 MERLIN#2 with a new chassis

This new development of Merlin was constructed for semi-autonomous navigation in a search and rescue mission as an experiment for the EU project under the IST-Future and Emerging Technologies program, Building Presence through Localization for Hybrid Telematic Systems (PELOTE) [DRI 04] [RUA 05a]. The robot provides sensor data for mapping and video for a tele-operator. All devices are mounted on the bigger chassis as shown in Figure 3.3a. Further, it is also equipped with a laser scanner. The new chassis has a wheel diameter that is two times larger than the old chassis and its platform is made on a scale of 1/8. There are also additional devices onboard: infrared sensors, a web camera, laser scanner and small lamps. However, even though the platform is big enough and flexible enough to carry all of the devices, the prototype has difficulty in steering regarding the weight and friction on the wheels and there are also difficulties in the installation of the hall sensors and magnets since the free space on the wheel is not available.

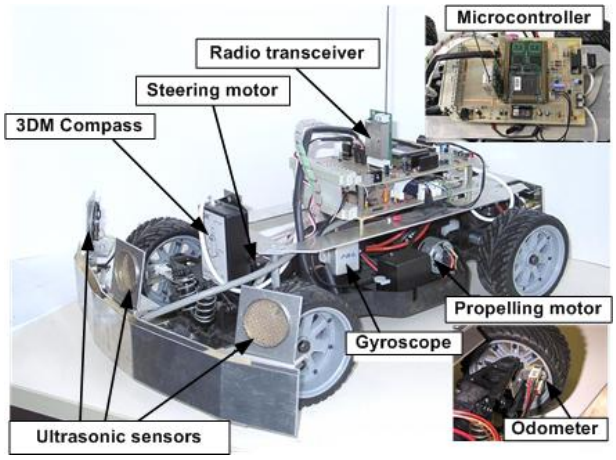
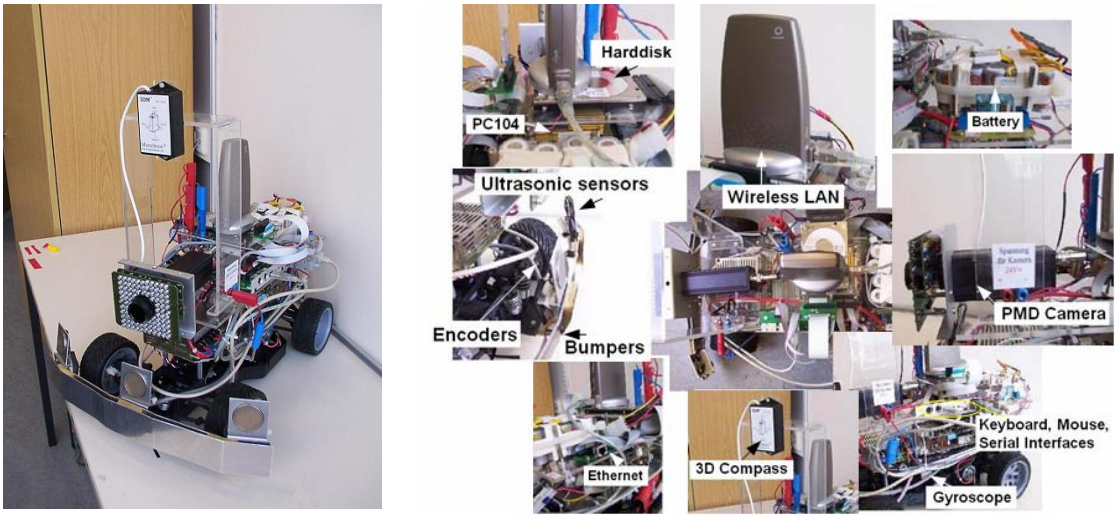
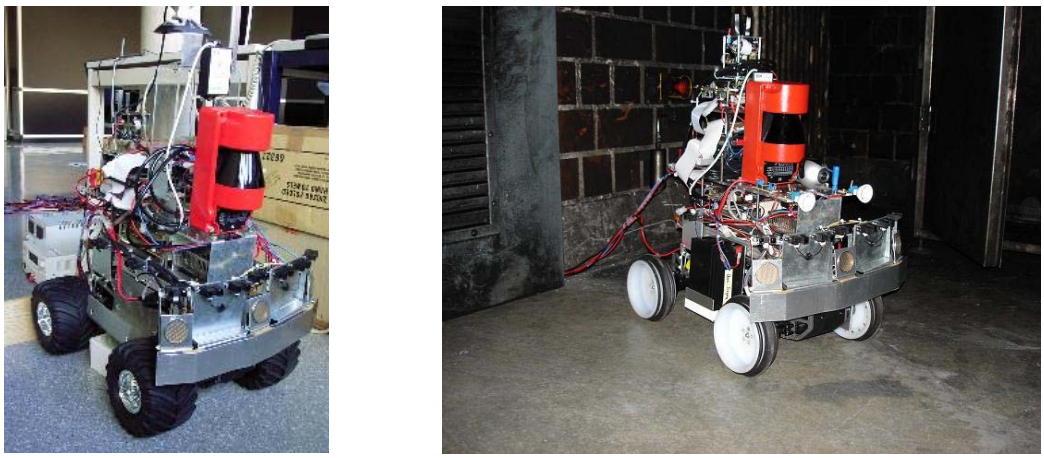


Figure 3.1: The first MERLIN prototype (MERLIN#1)



(a) (b)

Figure 3.2: MERLIN#2: (a) prototype; (b) devices on board of MERLIN#2.



(a) (b)

Figure 3.3: 1/8 scale chassis: (a) with the original wheels; (b) with the new wheels made of plastic and rubber rings (MERLIN#3).

The solution for these problems is to build wheels from plastic and rubber rings and to leave free space for mounting the hall sensors and magnet, instead of using the original wheel from the manufacturer. The wheels' rubber rings have smaller contact surface to the ground and thus less friction. The required steering force is much smaller than that of the old wheels and the power consumption in propelling the robot is also decreased. The odometers are mounted to the new plastic wheels, and magnets are glued into the prepared holes on the self-built wheels. Figures 3.3b show the robot after changing the wheels.

3.2 MERLIN Data Communication Structure

Wireless communication plays an important role in mobile robotics since it provides flexibility for navigation in a large area. There are two types of data communication systems on MERLIN: Data communication is possible via radio link and via WLAN. The software is introduced below first, then data communication via radio link and via WLAN, respectively.

3.2.1 MERLINServer and MERLINClient

Two java socket programs were developed for the wireless communication interface. The programs are MERLINServer and MERLINClient. MERLINServer passes the data from the socket to the COM port and vice versa as shown in Figure 3.4. Whenever MERLINServer receives a packet from the socket, it immediately passes the data packet to the microcontroller (robot) via the COM port. Whenever the opposite occurs, when MERLINServer receives data from the microcontroller via COM port, it passes the data packet via socket to MERLINClient.

MERLINClient exchanges the data from the socket to the graphic user interface (GUI). Whenever MERLINClient receives a data packet from the internet socket, it passes the data to the GUI and vice versa. MERLINClient and MERLINServer must both be started and connected to each other and hold the connection during robot operation. MERLINServer and MERLINClient share data via internet sockets. Note that appendix B provides the pictures and functions of GUI in details.

3.2.2 Data Communication via radio transceiver.

Two radio transceivers are exploited for data communication via radio link. One transceiver is connected through the RS232 serial port to a server computer and another transceiver is mounted directly on the robot. As shown above in Figure 3.4, the robot communicates via a radiotransceiver with a PC that acts as a server. This PC runs the java server program called MERLINServer. The client PC runs the MERLINClient program and serves as an interface for the operator that is transmitted through GUI. MERLINClient receives a command from the operator, such as a joystick command or path command and sends these commands to the robot via MERLINServer.

On the other side, the microcontroller translates the received control commands and applies the PWM signal to the motors. When data is transmitted in the opposite direction, such as the numerical sensor data from the wheel encoders, the gyroscope, the 3D compass, the ultrasonic sensors, and bumpers, they are sent as a packet to GUI via MERLINServer and MERLINClient.

There are two modes of TCP/IP connections. One is the remote host and the other is the local host. In the remote host mode, two PCs are connected via internet socket. One is the client PC and another is the server PC. In the local host mode, one PC runs both MERLINServer and MERLINClient as is depicted with a broken line in Figure 3.4. This radio data communication is available on all MERLIN prototypes.

3.2.3 Data communication via wireless LAN (WLAN).

MERLIN permits fast wireless data transmission from the PC104 located on the robot to a client PC. Therefore, WLAN communication is available only in MERLIN#2 and MERLIN#3. The WLAN communication functions such that the client PC runs MERLINClient and connects to the PC104 that runs MERLINServer via WLAN and the internet socket. The data transmission rate between the client PC and the PC104 can be as high as 11 Mbps. Figure 3.5 depicts the process of data communication via wireless LAN. The MERLINServer functions, as explained above, as an interface between the MERLINClient and microcontroller. In the same way, the PMDServer functions as a software interface for the PMD camera and the PMDGUI via the PMDClient. The same IP address is used with different port numbers to enable parallel communication between the two socket connections.

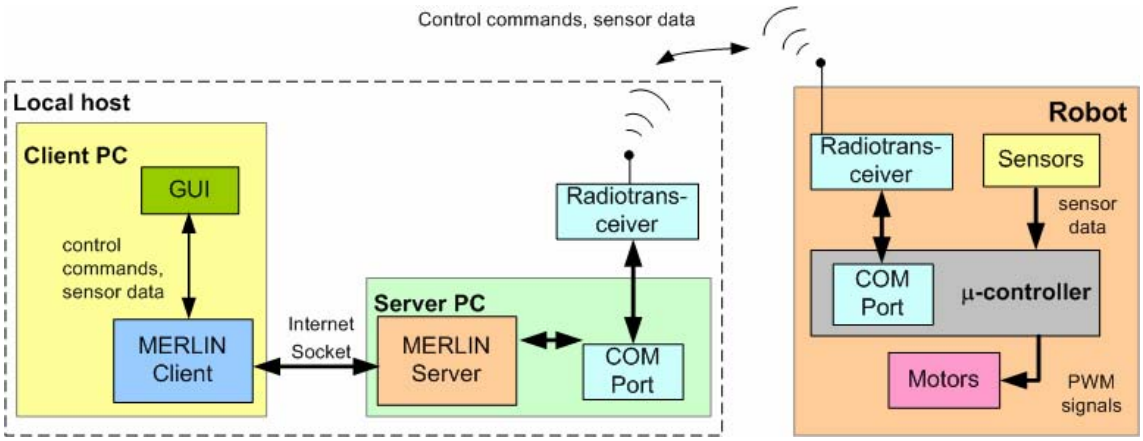


Figure 3.4: Data communication structure via radio transceiver.

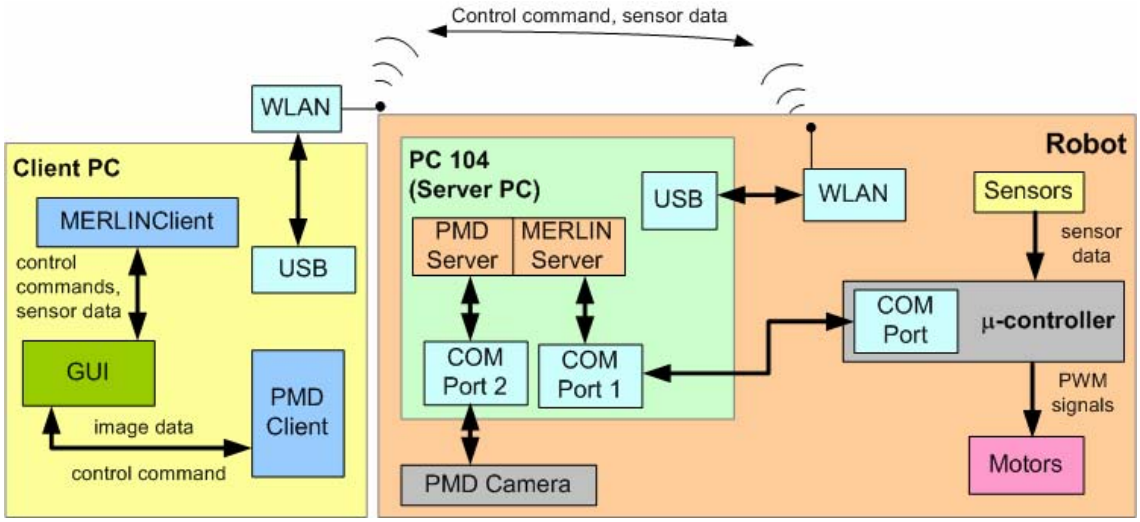


Figure 3.5: Data Communication Structure via wireless LAN.

3.3 Sensors for navigation

The processing of sensor data is very important and must be carefully performed. It involves converting measurement signal data into its numerical measured value. Any errors during this process affect the outcome of the control algorithms and their performance. Therefore, processing the data of each sensor is the first step for the development of all other algorithms. The discussion which follows presents the onboard sensor data acquisition.

3.3.1 The Odometer

The odometer provides information about the distance driven and speed. From our experiences, the front wheels of the robot's body are less affected by slippage than the rear wheels. Therefore, we mounted hall sensors and magnets on the front left and on the front right wheels, instead of on the rear wheels. Figure 3.6a shows the position of magnets and hall sensors that are mounted on the front wheel. When the robot is driving, the magnets pass through hall sensors and hall sensor generates a pulse at that moment.

3.3.1.1 Measurement of the driven distance. The output pulse signal of hall sensor is the input of the fast external interrupt on the microcontroller. That is, the interrupt routine is activated whenever the hall sensor generates the pulse. For the 1/8 and 1/10 scale chassis, we selected 8 and 16 magnets, respectively. The distance between two magnets should also be long enough for the microcontroller to clear the interrupt level signal. Every time that the interrupt signal is evoked, the distance is calculated. The distance driven is measured as the accumulated value after each reset. The calculations for the distances driven are as follows:

$$s_{odo} = \frac{2\pi R_{odo}}{n_{magnet}}, \tag{3.1}$$

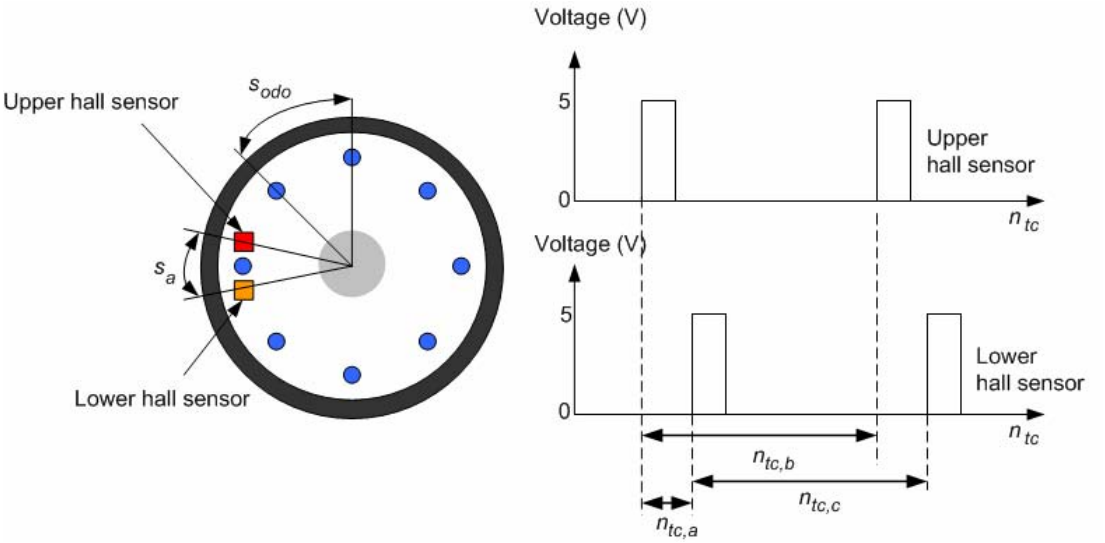
$$d_r = n_{odo,r} s_{odo}, \tag{3.2}$$

$$d_l = n_{odo,l} s_{odo}, \tag{3.3}$$

where s_{odo} is the distance between two magnets. R_{odo} is the radius of the wheel and n_{magnet} is the number of magnets on the wheel. The d_r and d_l are the driving distance on the right and on the left front wheels, respectively. The number of pulses for both the right and the left front wheels is $n_{odo,r}$ and $n_{odo,l}$, respectively. The number of pulses on each wheel is counted by the interrupt counter. The pulses on the counter are added up for the forward driving direction and deducted for the backward driving direction.

3.3.1.2 The detection of the driving direction. Two hall sensors are mounted on each wheel to determine the driving direction and the driving speed. The first hall sensor is mounted over another one as shown in Figure 3.6a. One is called the upper hall sensor and the other is called the lower hall sensor.

The generated pulse signals from the upper and lower hall sensors on each wheel are connected into a 54LS74 Flip-Flop logic IC and the output of the flip-flop is the logic state low or high that represent the directions forward and backward.



(a) (b)

Figure 3.6: Variable definition for speed calculation: (a) magnet and hall sensor positions on a wheel; (b) output pulse signals from hall sensor.

3.3.1.3 Measurement of the driving speed. The driving average speed v_{avg} of the robot is defined as the average speed of the front right and the front left wheels. The general equation for the speed calculation of a wheel is

$$z_v = \frac{s}{n_{tc} \cdot T_{odo}}, \tag{3.4}$$

where z_v is the wheel speed, s is the driven distance, n_{tc} is the number of pulses, and T_{odo} is the time constant of odometer's interrupt. T_{odo} is specified in the Timer interrupt and is set to 12.8 μs. The graphical definitions of these variables are shown in Figure 3.6b. The wheel speed can be measured in several ways as follows:

(a) by measuring the distance between the upper and the lower hall sensor (s_a) and the time interval that one magnet takes for passing from the upper to the lower hall sensor ($n_{tc,a}$),

(b) by measuring the distance between two magnets (s_{odo}) and the time interval that two magnets require to pass through the upper hall sensor ($n_{tc,b}$),

(c) by measuring the distance between two magnets (s_{odo}) and the time interval that two magnets require to pass through the lower hall sensor sensor ($n_{tc,c}$).

When exploiting one of these alternatives, the distance s in (3.4) is replaced by s_a or s_{odo} and n_{tc} is replaced by $n_{tc,a}$, $n_{tc,b}$ or $n_{tc,c}$.

3.3.2 The Gyroscope

The gyroscope provides analog output voltage that is proportional to the angular velocity, when the robot drives in a curvature. Two gyroscope sensors were tested on MERLIN, the Gyrostar ENV-05DB and ENV-05F-03. First, we exploited the ENV-05DB sensor which is capable of measuring an angular speed of up to ± 80 degrees/second. This sensor is no longer manufactured. Therefore, we switched to the ENV-05F-03, which is capable of measuring ± 60 degrees/second angular speed. The signal is amplified and is connected to the 10-Bits Analog to Digital (A/D) converter of the microcontroller.

Before making the first measurement of the angular velocity, the calibration of the offset value has to be performed. The offset value $O_{d,offset}$ is calculated from by averaging 20 measured values. This is also called the calibration of the gyroscope. This offset value is kept constant until recalibration is performed. The offset may change regarding the drift in the gyroscope when recalibration becomes necessary.

3.3.2.1 Measurement of the angular velocity. The angular velocity ω is calculated from the digitized value by using

$$\omega = c_{gyro}(O_{d,offset} - O_{d,average}), \tag{3.5}$$

where the $O_{d,average}$ is the average value from 10 measurements and the c_{gyro} is the scale factor of the graph of the angular velocity and the digital value of the gyroscope output.

3.3.2.2 The relative yaw angle measurement. At every time period of T_{gyro}, the current yaw angle is the accumulated yaw angle from the previous state. This relative yaw angle is calculated as follows:

$$z_{\psi,gyro} = z'_{\psi,gyro} + \omega T_{gyro}, \quad (3.6)$$

where $z_{\psi,gyro}$ and $z'_{\psi,gyro}$ are the relative yaw angle at the current and the previous state, respectively.

3.3.3 The 3-axis magnetic compass

The 3-axis compass is exploited for measurement of the absolute angle. The Microstrain 3DM compass sensor is a magnetic compass sensor. The measurement refers to the magnetic earth's pole and gravitation. A 3DM compass sensor is a 3-axis orientation sensor capable of measuring: ± 180 degrees of yaw heading, ± 180 degrees of pitch, and ± 70 degrees of roll. The sensor is calibrated and tested by using the test software from the manufacturer. Since the magnetic compass sensor is very sensitive to the magnetic field, the compass sensor is mounted at a height of 30 cm from the robot's base. The data from the 3DM compass sensor is sent to the microcontroller via the universal asynchronous receiver/transmitter (UART) at 9600 baud. The measurement is evoked by an interrupt routine. The sensor sends out the raw measurement data after a poll command is received. The raw data integer value is kept in the register of UART and is converted to a degree unit by multiplying with a constant value of 0.0055.

3.3.4 Ultrasonic sensors

The 6500 sonar ranging module is exploited on the robot in combination with the ultrasonic transducer for obstacle detection. This sensor provides accurate sonar ranging in the range of 6 inches to 35 feet from the target. The measurement is based on the principle of the time of flight; an ultrasonic impulse is sent out by a transducer, reflects on a target, and returns to the same transducer. The counter timer starts when the signal is sent out and stops when the reflected signal is received.

The range of the ultrasonic measurement is set to the maximum range of 7.14 meters to limit the cycle's measurement time. By using the counter timer with a time period of T_{ultr} = 25.6 μs, the operating time for each channel is 41.984 ms. The cycle time for four sonar ranging channels on board is 168 ms. The time of flight of the sound to the target equals half of the time for one trip. Therefore, the distance of the object is calculated by

$$z_{ultr} = \frac{c_{sound} n_{ultr} T_{ultr}}{2} \tag{3.7}$$

where c_{sound} is the speed of sound and is equal to 340 m/s, n_{ultr} is the counted number and T_{ultr} is the time period and z_{utlr} is the measured distance.

3.3.5 Infrared sensors

The SHARP GP2D12 and GP2Y0A02YK infrared sensors are capable of obstacle detection within a range of 10-80 cm and 20-150 cm, respectively. The sensors provide analog output voltage that is proportional to the distance to the obstacle. The output from each sensor feeds to the A/D converter of the microcontroller and the measurement update is performed periodically. The infrared sensor is used for auxiliary obstacle detection in combination with the ultrasonic sensor since the cycle time of the infrared sensor measurement is shorter than that of the ultrasonic sensors.

3.3.6 The Photonic Mixer Device (PMD) camera

The PMD camera has emerged from the new PMD technology. At the University of Siegen, the PMD camera has been researched and developed [HES 02] [SCH 04]. The camera provides ready-to-use depth information. In an image frame, each pixel of the image gives the distance value and also the 2D gray scale value. Each of 16x16 pixel on the PMD image is the measured distance value to an obstacle and its detection area not only covers altitude like a laser scanner does, but it covers a vertical and horizontal detection area with its cone volume of vision.

3.3.6.1 The PMD camera system. The principle of a 3D TOF imaging system is shown schematically in Figure 3.7. An object is illuminated by a high frequency modulated light source. The reflected light signal is compared with an electric reference signal. All systems presented are equal in the fundamental functions of photo-detection and signal processing. The 3D TOF imaging system starts with the detection of light, wideband amplifying, signal conversion and quantization. Each step has its own error source e.g. noise, that is transferred through the whole system. However, the light backscattered from the object is directly sensed and demodulated in the same area by the PMD array. The depth information of the scene is therefore acquired in pixels using

the correlation results of the received optical signal and the demodulation signal in parallel for the complete matrix.

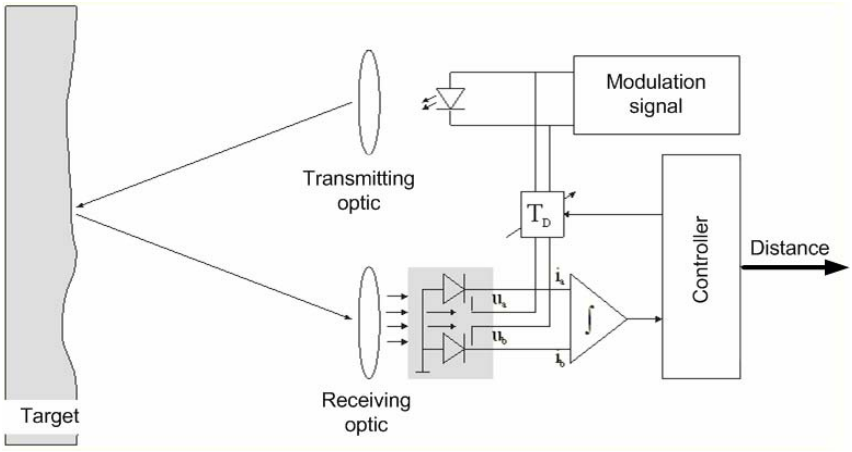


Figure 3.7: Schematic PMD TOF operation.

The principles of the camera system's structure are described here. Besides the PMD-matrix, the camera consists of the light source, a micro-controller, programmable logic and an analog/digital converter. The programmable logic generates two modulation signals and controls the phase difference. The velocity of light is denoted here by *c*. The following formula shows the relation between the frequency (*f*) and the range of unambiguousness (l_{max}) in a continuous wave time-of-flight measurement system:

$$l_{\max} = \frac{c}{2f} \tag{3.8}$$

Because the light signal has to run the distance between camera and object twice, the range of unambiguousness is reduced by half. The first modulation signal is used to control the light source. The second signal builds the push-pull voltage for the photo gates of each of the 16x16 pixel PMD-arrays. The two supplied read out values A and B of each pixel are multiplexed to the analog/digital converter. The digital data is then processed in a micro-controller. This controller is the camera's main control and evaluation tool. It shifts the phase between the modulation signals stepwise and collects the delivered correlation values. Using a least-square algorithm, the controller evaluates the phase delay of every single pixel. This phase difference φ_{Pixel} is proportional to the distance value d_{Pixel} that can be calculated by the following term:

$$d_{Pixel} = \frac{\varphi_{Pixel} c}{2\pi 2 f} \tag{3.9}$$

3.3.6.2 The 16x16 pixel PMD camera. Figure 3.8 shows the photograph of the assembled camera. The camera's output is the distance information in each of the 16x16 pixels. The camera operates at the frame rate of 3 Hz and the half angle of the detection area is 12 degrees. The maximum detection distance is 15 meters limited by the modulated frequency.

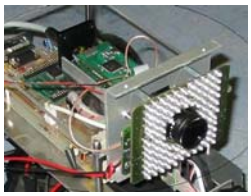


Figure 3.8: A photograph of the 16x16 pixel PMD camera.

3.3.6.3 The 48x64 pixel PMD camera. This camera is manufactured by PMDTec and the name of product is PMD[vision]® 3k-S. Figure 3.9 shows the photograph of the camera. The camera's output is the distance information in each of the 3072 pixels. The camera operates at the frame rate of up to 25 fps and the half angle of the detection area is 40 degrees with the lens focus of 12mm. The maximum detection distance is 7.5 meters for the modulation frequency of 20 MHz. The modulation frequency is adjustable and is inversely proportional to the maximum detection distance.

Figure 3.9: A photograph of the 48 x 64 pixel PMD camera.

3.4 Steering and propelling the car-like mobile robots.

The steering and propelling methodology for the car-like mobile robot is similar to that of a car. MERLIN turns its front wheels into the desired steering position and the driving speed after receiving the command from the operator. The robot has two

motors; one for steering and another for propelling. These motors are controlled individually by using the pulse width modulation (PWM) that is generated by the microcontroller onboard.

3.4.1 The steering control

The purpose of steering control is to control the steering angle of the front wheels so that the robot drives in the desired orientation. Since the servo Futaba S3003 has internal position controller, assigning the proper PWM value is setting the steering angle. The frequency of the steering PWM signal is set to 50 Hz. The control of the steering angle is divided into two modes, the joystick control mode for manual driving and the path control mode for autonomous driving.

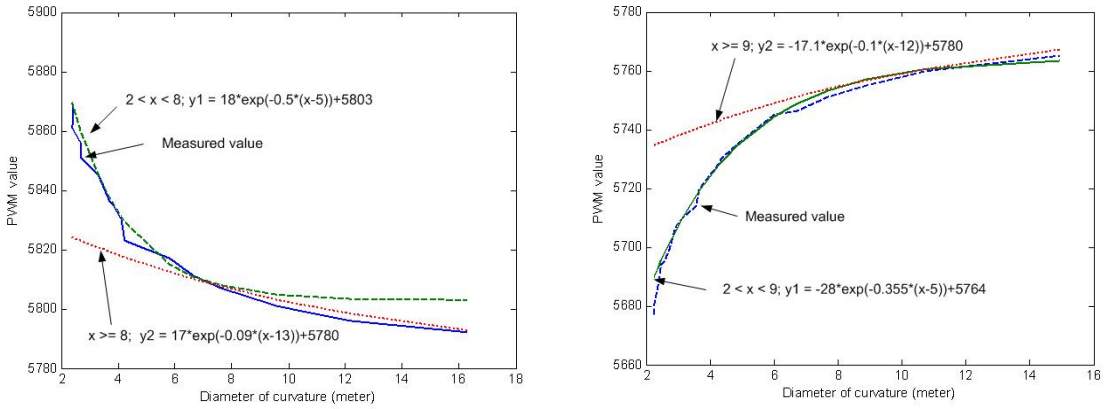
3.4.1.1 Steering control by using Joystick. In its joystick control mode, the robot calculates the PWM value from the joystick commands that are the integer scale numbers in a range of 0 to 100. The PWM value is calculated by using

$$pwm_{steering} = 2(n_{js} - 50) + pwm_{steering,0} \tag{3.10}$$

where $pwm_{steering}$ is PWM value assigned related to the desired scale number, n_{js} is the scale number in a range of 0 to 100 and $pwm_{steering,0}$ is PWM value assigned related to the zero degree (straight) steering position.

3.4.1.2 Steering control for autonomous path following. In the path control mode, the orientation control is assigning the PWM to achieve the desired steering position. The two main types of path commands used are the line path and arc path. For a line path, the robot always sets its steering position to zero. For a curved path or arc path, the robot sets its steering angle according to the path command, which contains of the radius of curvature. The calculation of PWM value is explained here. Let x be the diameter of the curvature.

When the robot drives in a clockwise direction, there are two mathematical equations, which are formulated based on the robot's performance in experiments as shown in Figure 3.10a:



(a) (b)

Figure 3.10: The relationship between the PWM value and the diameter of the curvature when driving: (a) in a clockwise direction; (b) in a counter-clockwise direction.

$$pwm_{steering} = 18e^{(-0.5(x-5))} + 5803; \qquad \text{if } (2 < x < 8), \tag{3.11}$$

$$pwm_{steering} = 17e^{(-0.09(x-13))} + 5780; \qquad \text{if } (x \geq 8), \tag{3.12}$$

The experiments showed that, for each value of PWM, the robot drives with a constant steering angle in a circle path, so the diameter of the circle is measured. Similarly, when the robot drives in a counter-clockwise direction, the relation between PWM value and diameter of curvature as shown in Figure 3.10b are

$$pwm_{steering} = -28e^{(-0.355(x-5))} + 5764; \qquad \text{if } (2 < x < 9), \tag{3.13}$$

$$pwm_{steering} = -17.1e^{(-0.1(x-12))} + 5780; \qquad \text{if } (x \geq 9). \tag{3.14}$$

3.4.2 Speed control

We apply the well known proportional and integral (PI) control for motor closed loop speed control. The continuous PI controller is derived into the digital controller as in [FRA 98]. The control algorithm of the digital PI controller is

$$u_{spc}(k) = u_{spc}(k-1) + K_p\left[e(k) - e(k-1)\right] + K_i e(k), \tag{3.15}$$

$$u_{spc,\min} \leq u_{spc}(k) \leq u_{spc,\max}, \tag{3.16}$$

where $u_{spc}(k)$ is the control input at the time step k, K_p and K_i are the proportional and integral gain, $e(k)$ and $e(k-1)$ is the speed error at time step k and at the previous time step k-1, respectively. The saturation of the control input is applied as in (3.16). The minimum and maximum values of the control variables are $u_{spc,min}$ and $u_{spc,max}$. The speed error is the difference between the average velocity v_{avg} and the reference velocity v_{ref}.

$$e(k) = v_{ref} - v_{avg}(k). \tag{3.17}$$

$$pwm_{propelling}(k) = pwm_{propelling}(k-1) + u_{spc}(k), \tag{3.18}$$

$$pwm_{propelling,mf} \leq pwm_{propelling}(k) \leq pwm_{propelling,mb}, \tag{3.19}$$

where $pwm_{propelling}(k)$ and $pwm_{propelling}(k-1)$ are PWM values assigned to the propelling motor at the time steps k and $k-1$,respectively. $pwm_{propelling}(k)$ is constrained by the maximum forward value $pwm_{propelling,mf}$ and the maximum backward value $pwm_{propelling,mb}$ as in (3.19). These values have been set to limit the robot maximum speed for safety. The frequency of the propelling PWM signal is set to 100 Hz.

The controller gains K_p and K_i are tuned by using the trial and error method. The resolution of the hall sensors is 8 pulses /rev. Table 3.2 shows the reference table that contains the tuned parameters K_p and K_i corresponding to various reference speeds. These parameters are tuned according to trial and error on the real system.

Table 3.2: Parameters for the PI controller at the specified reference speed.

Desired speed (m/s)	K_p	K_i
0.5	0.080	0.009
0.4	0.070	0.009
0.3	0.050	0.008
0.2	0.030	0.006
-0.2	0.025	0.005
-0.3	0.035	0.006
-0.4	0.045	0.007
-0.5	0.060	0.008

Further from measurement of these on board sensors is how MERLIN navigates. The next chapter explains autonomous features for navigation. Those are obstacle avoidance, 180 degree turn, and path following control.

4 Autonomous Features

Chapter 3 gave an overview of MERLIN, introduced the data communication, sensors, sensor data processing, and robot steering and speed control. These sensor data and motion controls are further exploited in the several autonomous features described below in this chapter. Several obstacle avoidance techniques were designed and are presented in Section 4.1. Section 4.2 describes the robot's capability to execute an autonomous 180 degree turn in a narrow corridor. The last section explains the development of an autonomous path following in an unknown environment.

4.1 Obstacle detection and collision avoidance.

In an unknown environment, the information about obstacles, such as walls or objects, are not known beforehand. Various techniques for obstacle collision avoidance can be implemented based on the perception capabilities of on-board sensors and strategy. Two categories of obstacle collision avoidance are discussed here; obstacle collision avoidance and wall following.

The goal of obstacle collision avoidance is to avoid collision without the necessity of identifying what type of obstacles the robot encounters. When the robot detects an obstacle, it tries to turn into another direction where the obstacle is not detected. For the wall following, the robot avoids collision by following the obstacle’s boundaries and keeps at a constant distance away from the obstacle.

The design of these obstacle avoidance algorithms is based on the perception capabilities of the robot. Selected because of their light weight and compact size, four ultrasonic and six infrared sensors were mounted onto the robot as shown in Figure 4.1. The infrared sensor provides short distance obstacle detection up to 0.8 meters, whereas the ultrasonic sensors provide long distance detection up to 7.0 meters. Among the sensors located on the front and on the back of the robot, infrared sensors have higher priority (compared to the ultrasonic sensors) regarding the fast measurement updates. The ultrasonic sensors take a longer cycle time waiting for their reflected signal. Therefore, the data from the infrared sensors only replace the ultrasonic measured data when the obstacle lies within 0.8 meter from the robot.

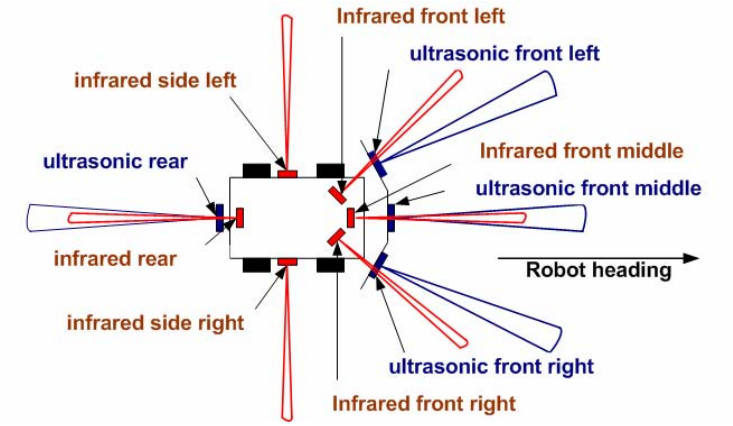


Figure 4.1: Positions of the ultrasonic and infrared sensors.

4.1.1 Obstacle collision avoidance

Without prior knowledge of the obstacles met, the robot relies on its ultrasonic and infrared sensors to detect obstacles. When the obstacles detected lie within 1.5 meters, the obstacle collision avoidance controller is applied automatically. A car-like mobile robot avoids collision by turning into other direction and driving forward when the obstacle is far away, or by driving backwards when the obstacle is close by. The steering control works using a fuzzy logic controller. However, in some situations, the fuzzy logic controller has lower performance than the if-then rules. Therefore, the designed controller consists of two options; the fuzzy logic controller and the if-then rule controller.

4.1.1.1 Fuzzy logic controller. MERLIN was designed to avoid collision with obstacles using a fuzzy logic controller [DRI 93]. The fuzzy logic controller is selected because of its small memory requirement for computation. The structure of the fuzzy logic controller is illustrated in Figure 4.2. The inputs of the controller are the distances measured by the three ultrasonic ranging sensors and three infrared sensors, and the output is a PWM signal used to control the steering motor.

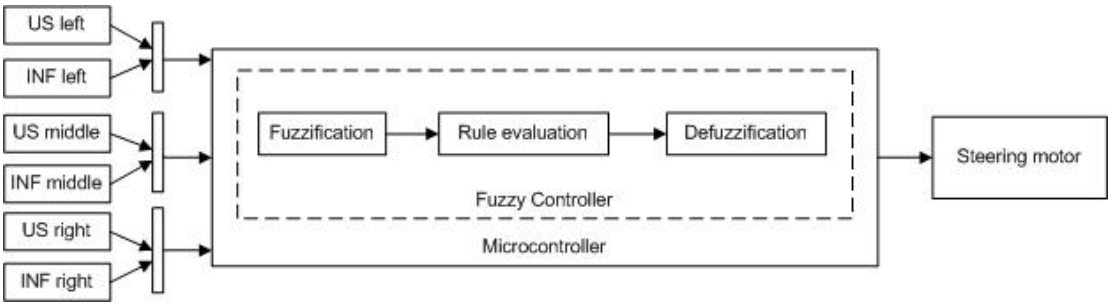


Figure 4.2: The structure of the fuzzy controller.

The fuzzy logic controller is mainly composed of three components, fuzzification, rule evaluation and defuzzification. The Sugano-type inference system known as the singleton output membership function was also selected for use, due to its characteristic of enhancing the efficiency of the defuzzification process by weighting the average of crisp outputs of inference rules.

4.1.1.2 Controller design. The fuzzy logic controller starts by converting the obstacle measured distance into the fuzzy language according to gathered experiences. As it does this, the crisp input is translated into a fuzzy value. The distinct (non-fuzzified) variables that are the distances measured by those sensors are fuzzified based on corresponding affiliation functions (with a value between 0 and 1). The specified fuzzy subsets are shown in Figure 4.3. The input variables are "Left", "Middle", and "Right". The distance measurement is divided into three fuzzy subsets, *Very near* (*VN*), *Near* (*N*) and *Far* (*F*). The steering angle is divided into five subsets that are *Left*, *Slightly left*, *Straight*, *Slightly right*, and *Right*. After they have been fuzzified, the fuzzy representations of the input are used to compute the fuzzy output truth values. These values are based on a Max-Min inference.

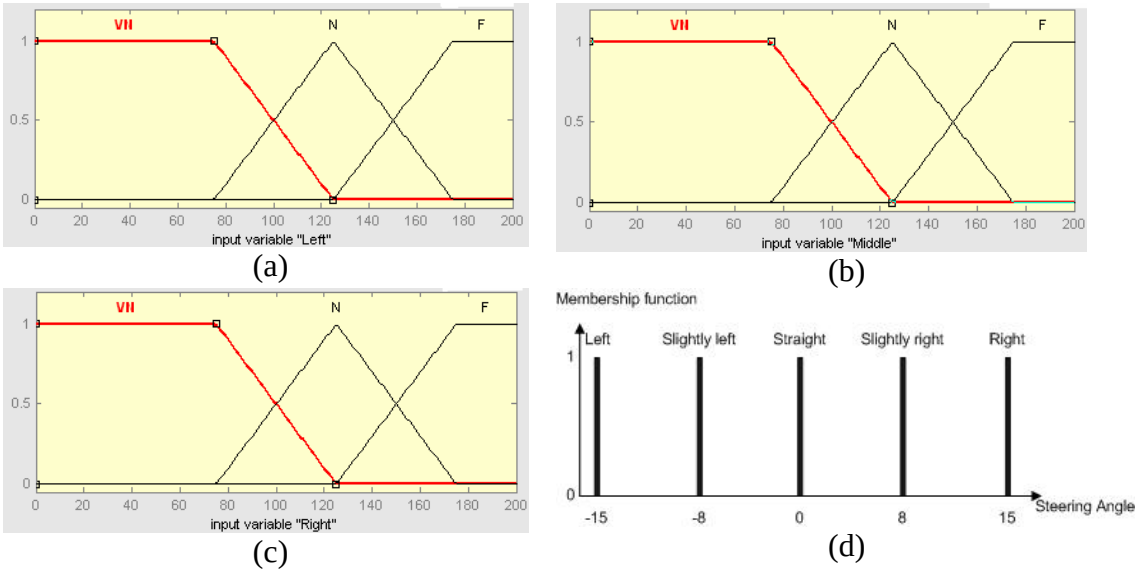


Figure 4.3: Membership functions for obstacle avoidance: (a) input variable "Left"; (b) input variable "Middle"; (c) input variable "Right"; (d) output variable "Steering".

To simplify the representation of the vehicle's activity, 18 fuzzy rules are applied as shown in Table 4.1. The main idea behind the fuzzy control rules is described as

follows: The process of control according to fuzzy rules is to translate the fuzzy output to a crisp value. To do this, the singleton defuzzification method is used. The crisp output of the fuzzy controller u^* is calculated as

$$u^* = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i}, \tag{4.1}$$

where w_i is the firing strength of the rule or the maximum membership function of rule i and z_i is the output level of rule i.

Table 4.1: The fuzzy inference rules for obstacle avoidance

Rule no.	Left	Middle	Right	Steering
1	F	F		Straight
2	N	F		Straight
3	VN	F		Straight
4	VN	N		Right
5	F	N		Slightly left
6	F	VN		Left
7	N	N		Left
8	N	VN		Left
9	VN	VN		Right
10		F	F	Straight
11		F	N	Straight
12		F	VN	Straight
13		N	VN	Left
14		N	F	Slightly right
15		VN	F	Right
16		N	N	Right
17		VN	N	Right
18		VN	VN	Left

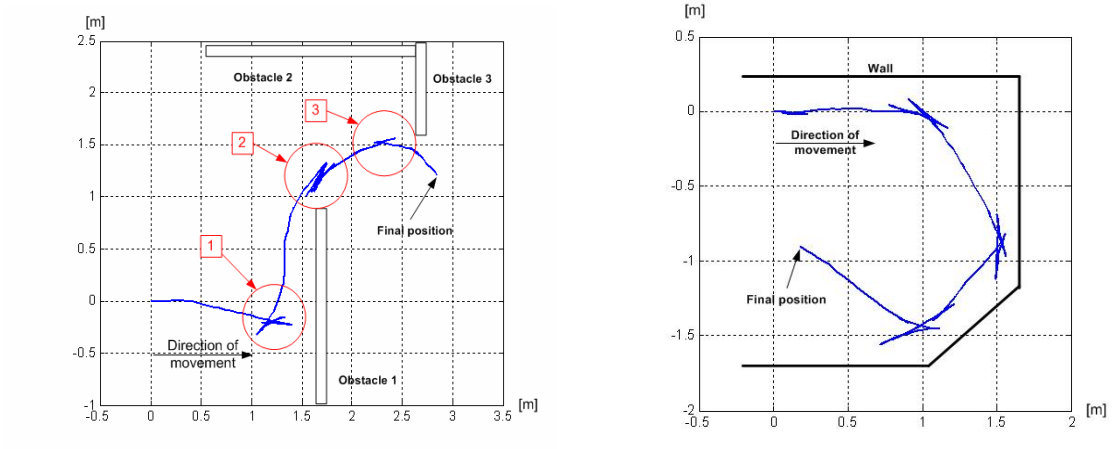
4.1.1.3 If-then rule controller. When the robot is very near to the obstacle or at the wall corner, the robot sometimes needs to drive backwards. This is when the if-then rules come into effect. These if-then rules are necessary when the robot is near to a wall corner and the front distances to the left and to the right are nearly equal. The if-then rules are applied as the two concatenated steps of movement, driving backward followed by driving forward, in which the robot steers the front wheel into the same orientation as its previous move. In order to this, the priori orientation is recorded. For the next move, the robot's heading remains in the same orientation but opposite direction as the previous move. The if-then rules are shown in Table 4.2.

4.1.1.4 Test results in different scenarios. Experiments were performed to test the autonomous steering control using two types of scenarios: driving along an open-

ended wall and into a dead-end. Figure 4.4a shows three obstacles that represent the open-end wall and the driven path. The solid line shows the robot's driving path. The robot started from position (0, 0). In front of the 1st obstacle, in the 1st marked circle, the robot drove forward and backward and turned in another orientation many times before getting free of the obstacle. Finally free of the obstacle, the robot's head was pointing into the 2nd obstacle. It drove forward and stopped again in front of the 2nd obstacle. The robot tried to change its orientation in the 2nd marked circle. At this position, the robot moved straight on forward and backward several times since the measured distance from the front left sensor and the front right sensor were almost identical. Also, the distance form the side left and the side right sensors were identical. As shown in the figure, the repetitions of movement are shown in the marked circles. After that robot met the obstacle free direction, it moved forward. From this position, the robot found its way out from in front of the obstacles and came out to the final position.

Table 4.2: If-then rules for obstacle avoidance.

Rule no.	Priori (if)		Next (then)	
	Orientation	Direction	Steering angle position	Propelling direction
1	Clockwise	Forward	Maximum left	Backward
2	Clockwise	Backward	Maximum left	Forward
3	Counter clockwise	Forward	Maximum right	Backward
4	Counter clockwise	Backward	Maximum right	Forward



(a) (b)

Figure 4.4: The driven path of obstacle avoidance; (a) in the open-end wall scenario; (b) in the close-end wall scenario.

Figure 4.4b shows the robot's driving path in the close-end wall scenario. The robot started at position (0, 0) facing the wall and tried to move into an obstacle-free area. Finally, the robot turned around away from the dead end. The robot changed its orientation at each wall corner and only turned clockwise regarding the if-then rule control as mentioned above.

4.1.2 The PMD camera for obstacle detection.

An application of the camera to mobile robots is to use the camera for obstacle detection and collision avoidance in an indoor environment, where the obstacles can be tables, chairs, people, wall, etc. The advantages of using a PMD camera over ultrasonic sensors are that specula reflection phenomenon does not exist and the blind area is covered. Figure 4.5 shows an obstacle lying in the blind area between the front middle and the front right ultrasonic sensors and detection areas of each ultrasonic sensor and of the PMD camera.

The first step in obstacle collision avoidance is to acquire the PMD data. In order to do this, the 16x16 array is arranged into three groups, left, middle and right. That means that the matrix is divided into 3 bands, 16x6, 16x4, and 16x6. Among the left, middle and right bands of detection on the PMD camera, the shortest distance is chosen as a representation of all pixels in the band. We take the representative shortest distance in each band by using the following method. Since there is a lot interfering noise when the 16x16 pixel takes measurements while the robot is driving, the robot stops before the camera captures image and takes the average value from four images.

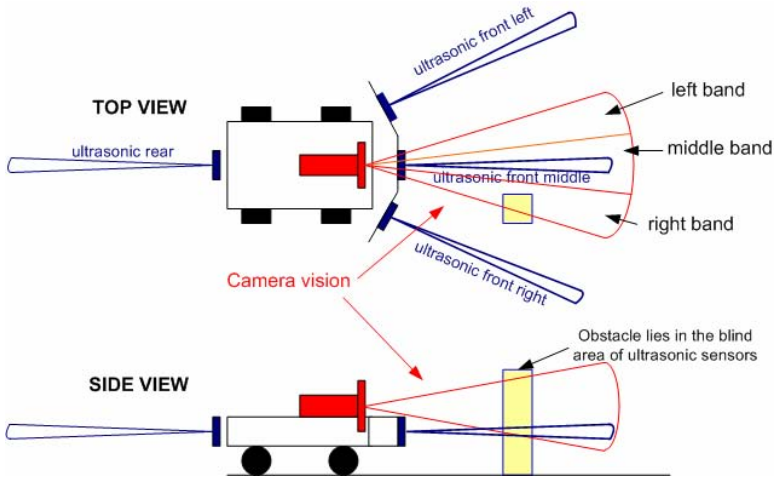


Figure 4.5: The area of detection of PMD camera and ultrasonic sensors divided into left, middle and right sections.

There are two ways to implement the PMD camera on MERLIN. The first way is to use the PMD camera to replace the three ultrasonic sensors located at the front of the robot. The advantage of using the PMD camera instead of the front ultrasonic sensors is that the robot is able to detect small objects, such as the leg of a table, that normally lie within the ultrasonic sensors' blind spot. However, only the camera has narrow vision of detection. The second way is to use the camera along with the ultrasonic sensors. This is more effective than the first method. The combination of the ultrasonic sensors and the PMD camera vision as in Figure 4.5 (top view) provide the superposition of obstacle detection. The ultrasonic sensors detect objects located far away from the robot and located within detection areas of the most left and the most right ultrasonic sensors, whereas the PMD camera detects objects close to the robot with its fine pixel detection in the front area of the robot.

4.1.3 Wall following

Another technique to avoid obstacle collision is to let the robot follow a wall. This is particularly appropriate for avoiding collision with large obstacles with a long boundary. The robot tries to stay at a constant distance away from and parallel to the wall, instead of only trying to avoid collision as described in section 4.1.1.4. The infrared sensors mounted at the front of the robot's body are exploited in this technique and the fuzzy logic algorithm and if-then rules are applied. The wall following algorithm can be divided into two modes; wall following on the left hand side and wall following on the right hand side. The fuzzy logic controller used here is identical to the fuzzy logic controller described in section 4.2.1.1.

The input of the controller is the error between the desired reference d_{ref} and the measured distance d from the front left (left hand side mode) or front right sensor (right hand side mode). The input variable "*error*" is calculated by

$$error = d_{ref} - d \tag{4.2}$$

The fuzzy rules are evaluated as follows:

1. If *error* is *neg*, then *steering* is *right,*
2. If *error* is *pos*, then *steering* is *left,*

where *neg* and *pos* have an input membership function as shown in Figure 4.6a. The output variable *steering* has two membership functions that are *right* and *left* as shown

in Figure 4.6b.

As an auxiliary controller, the if-then control is also applied for the wall corner. Figure 4.7a shows collision avoidance at the inner wall corner. The robot's position numbers, 1 - 4, represent the movement steps. The robot takes the previous movement's orientation in to consideration. As in section 4.1.1.3, the if-then rule allows the robot to move in the same orientation as the previous move. Moreover, at wall edges, when the measured distance *d* is suddenly jumps to a large value as shown in Figure 4.7b, an extra rule is to let the robot turn around the edge. The wall following controller is exploited in testing for robot localization in Chapter 5. The experimental result of the wall following autonomous function is presented below in Section 5.4.2, in Figure 5.6.

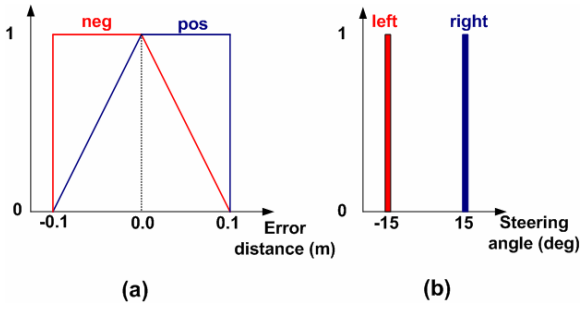


Figure 4.6: Membership functions for wall following: (a) input; (b) output.

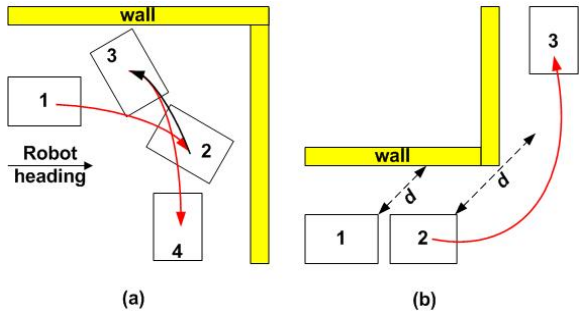


Figure 4.7: Wall following; (a) at a corner; (b) along an edge.

4.2 An autonomous 180 degree turn in a narrow corridor.

The autonomous turning function is used at the dead end of a narrow corridor where the robot has to turn its head into the direction from which it came. Regarding the car-like characteristics, the robot manoeuvres itself around in this situation in a manner similar to the path shown in Figure 4.8.

At early stage, the autonomous function is basically developed with the assumption that there would be no other obstacle in the dead-end corridor. Trial experiments resulted in a best combination of paths which are stored in the robot's memory. Whenever this manoeuvre is necessary, the movements are read and performed step by step until the end, which is when the robot comes to the same heading angle of +180° or -180°. When the obstacle avoidance is integrated, the process becomes as shown in Figure 4.9.

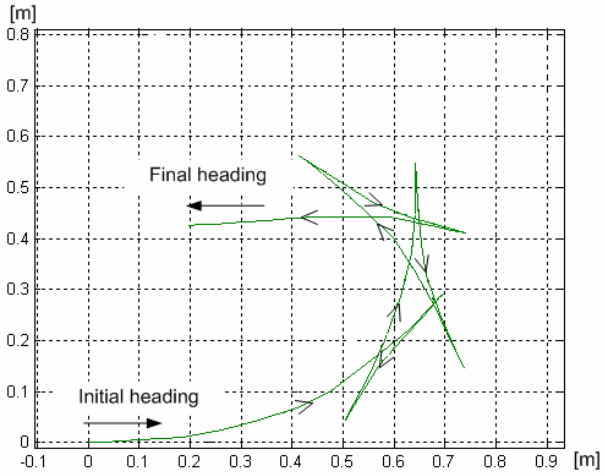


Figure 4.8: The experimental result for an autonomous 180 degree turn.

On left hand side of the flow chart is the process explained above for a turn without meeting any obstacle. When the robot does meet obstacles in its current driving direction, it stops and does process A, on the right hand side of the flow chart. The robot checks its record of previous move to determine whether it was moving forward or backward, clockwise (cw) or counter clockwise (ccw), in order to assign the movement command. The test result of for autonomous 180 degree turn is shown previously in Figure 4.8. The final heading position of the robot is close to 180 degrees from its original position. It is also important to make sure that the angle measurement during the turning process is exact. The method for reaching an angle measurement was explained above in Section 3.3.2.2.

4.3 Path following in an unknown environment.

In an unknown environment, the intelligent navigation requires the path following control. The path following control for mobile robots is the automatic control

of a robot along a specified path without human interference. The path following strategy is the integration of the basic path following control, wall following, and trajectory generation. The wall following control is explained above in section 4.1.3. Here, the trajectory generation, the basic path following control and the path following strategy are explained.

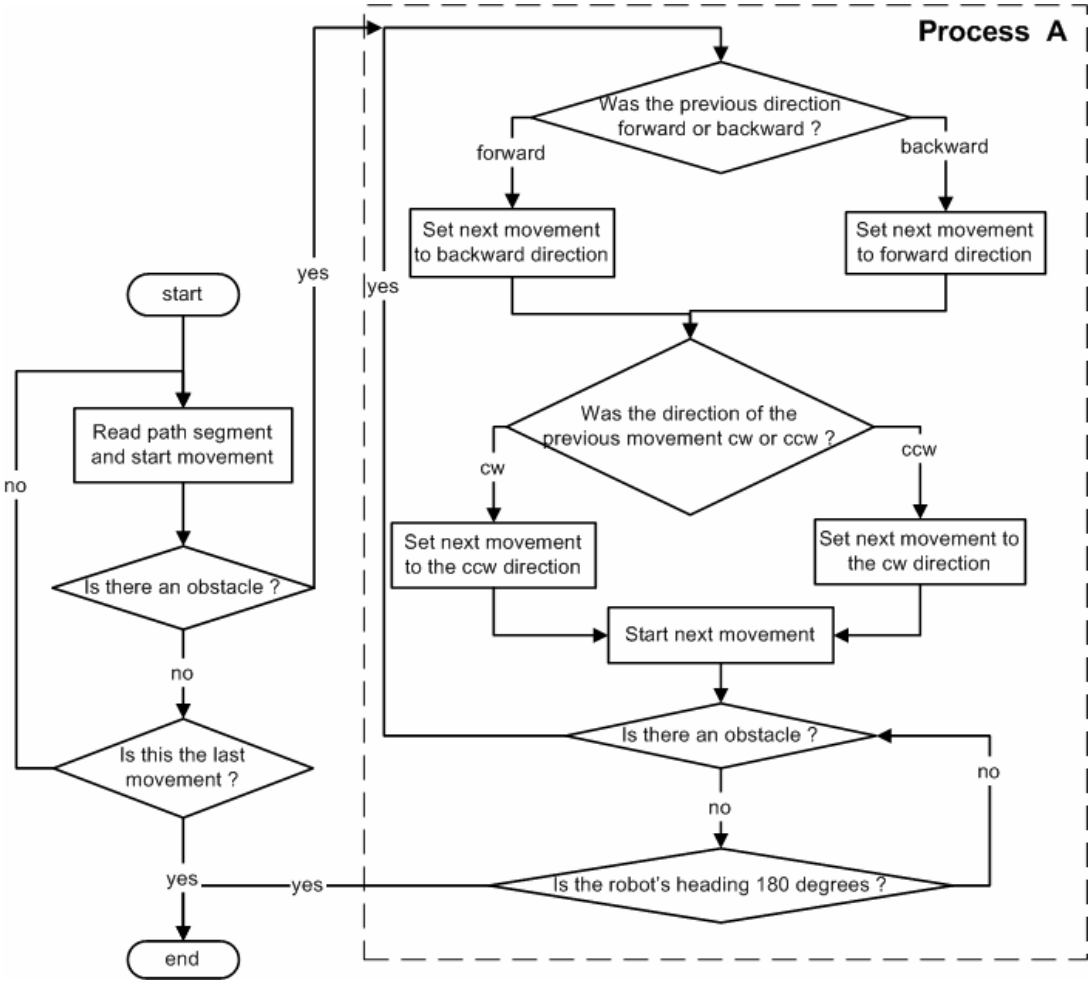


Figure 4.9: Autonomous 180 degree turning process.

4.4.1 Trajectory generation

When that robot has become free of any obstacles and is no longer on its original path, the robot has to reach its desired final position. The trajectory generation strategy provides the fittest trajectory between the robot's current position and the desired final position based on the car body's manoeuvring characteristics. The sample trajectories are shown in Figure 4.10. The trajectory consists of two non-symmetrical sub-paths with a different curvature radius for each sub-path $r_1 \neq r_2$. Also, the final heading angle of each sub-path is unequal $\theta_1 \neq \theta_2$.

The distances to destinations *dx* and *dy* are the distances between the robot's current position and the destinations in *x* and *y* directions. Note that the robot's heading position is referred to as x-axis direction. The distances to the destinations are calculated by

$$dx = r_1 \sin\theta_1 + r_2 \sin\theta_{2,} \tag{4.3}$$

$$dy = r_1 \cos\theta_1 + r_2 \cos\theta_{2,} \tag{4.4}$$

where $\theta_1 = \theta_2$ and $r_1 = r_2$. As a result, the sub-paths are symmetrical and the distances to destination of the symmetrical sub-paths are

$$dx_{sym} = 2r_1 \sin\theta_1, \tag{4.5}$$

$$dy_{sym} = 2r_1 \cos\theta_1. \tag{4.6}$$

The symmetrical trajectories are shown in Figure 4.9b. The radius is fixed as r_1 and the angles vary as θ_1, α_1, and β_1 result in three different trajectories. Note that for all trajectories, the current heading is also the final heading. In the iterative loop of the trajectory generation, the angle θ_1 and radius r_1 are varied. The iterative loop searches for the fittest trajectory using (4.5), (4.6), and

$$x_f = x_{final} - x_{current}, \tag{4.17}$$

$$y_f = y_{final} - y_{current}, \tag{4.18}$$

$$e_x = x_f - dx_{sym}, \tag{4.19}$$

$$e_y = |y_f| - dy_{sym}, \tag{4.20}$$

$$e_{sum} = |e_x| + |e_y|, \tag{4.21}$$

where x_{final}, y_{final}, $x_{current}$, and $y_{current}$ are the current and the final position coordinates, repectively. The fittest path is the trajectory with the minimum value of e_{sum}. Note that the minimum 1 meter radius is the shortest curvature radius and a 90 degree angle is the maximum angle for each sub-path.

4.3.2 Basic path following control

The basic path following control is the path following control that functions under the assumption that there is no obstacle along the path during the operation. The robot moves along the specified path and stops at the destination without avoiding

collision. The steering control as explained in section 3.4.1.2 is exploited here. Two available path types are curve path and line path.

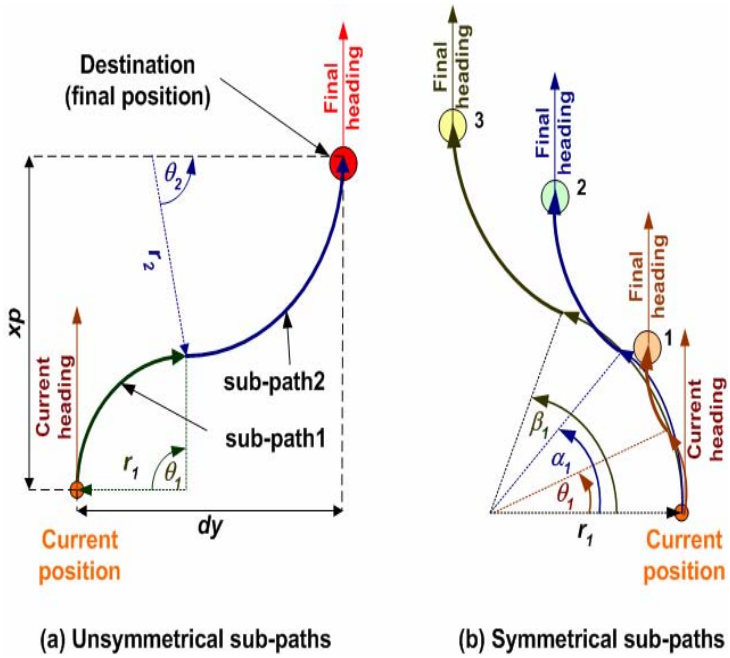


Figure 4.10: Trajectory generation

In a curve path command, the data packet consists of the radius of curvature and the desired heading. Whenever the robot receives a curve path command, the radius is converted into the PWM value using (3.9 - 3.12). Meanwhile, the desired heading is kept as a reference value. Periodically, the desired heading has to be compared to the current heading. The robot heading is also called the robot's orientation. Therefore, the measurement of robot heading is the relative yaw angle measurement in section 3.3.2.2. When the current heading reaches the desired heading, the curve path following is over. The first sub-path in Figure 4.10a is an example of two concatenated curve paths. In this figure, the first curve path consists of the radius of curvature is r_1 and the desired heading is θ_1. The second curve path consists of r_2 and θ_2, respectively.

For a line path rather than a curve path, the command packet consists of the path length and the movement direction. The robot sets the front wheels to drive straight on and the desired final heading is zero radian. The path length is kept as a reference value and is compared to the distance already driven, measured by the odometers on the front wheels as explained in section 3.3.1.1. When the driven distance reaches the path length, the line path following is over.

4.3.3 Path following strategy

The path following strategy integrates the wall following and trajectory generation with the basic path following control. Figure 4.11 shows the original path command as a broken line through the obstacle. Initially, the robot receives a path command from its user and begins by recording its current position and heading. When the robot detects the obstacle, it stops in front of the obstacle at position 1. The wall following algorithm determines the orientation of the robot for the next move by considering not only the obstacle's position, but also the robot's current position and its desired final position. The robot performs wall following until the robot meets no other obstacles in front of it. After the robot is free from obstacle, at position 2, it stops and calculates the difference between its current heading and its desired final heading. Then the robot adjusts its heading by turning into the direction of the final heading and stops when the robot heading reaches the final heading as shown at position 3.

At this moment, the robot checks the distance to the final position y_f. The robot sends a request for the trajectory generation to the client PC. At this stage, since the robot heading is pointing to the desired final heading, the trajectory generation is performed by using (4.7) - (4.11) on the client PC. The robot waits for the generated trajectory from the client PC. After the robot receives the trajectory, the robot moves along the trajectory path and stops at the destination, where the robot heading points in the direction of the desired final heading and the robot position is at the desired final position. If the robot finds obstacles before it reaches the destination, it repeats the process again from position 1. Please note that we call the operation from position 3 to the destination as trajectory following.

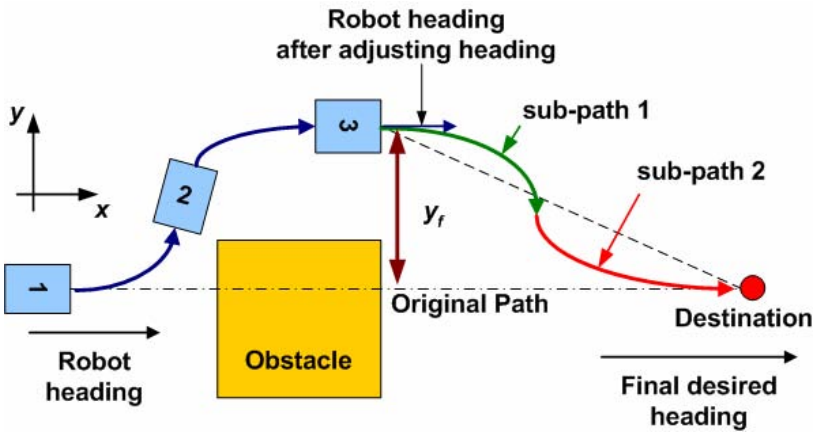


Figure 4.11: Path following strategy

4.3.4 The data communication for the path following strategy.

Figure 4.11 shows the data exchange between the robot and client PC during the real-time path following operation. The basic path following controller and wall following controller are located on the microcontroller. The client PC runs the robot localization, trajectory generation, and GUI. The localization technique is explained in detail below in Chapter 5 and details of GUI are provided in Appendix B. As explained in section 3.2, the robot transmits sensor data to and receives control commands from the client PC continuously. Whenever the robot desires a trajectory generation, the robot sends a request command to the client PC. After the client PC generates the trajectories, it sends the generated trajectory commands to the robot. On the client PC, the estimated current robot positions $x_{current}$ and $y_{current}$ are obtained from robot localization and the distances to the destinations x_f and y_f are updated by using (4.7) and (4.8) and are sent from the client PC to the robot periodically at every 0.2 seconds.

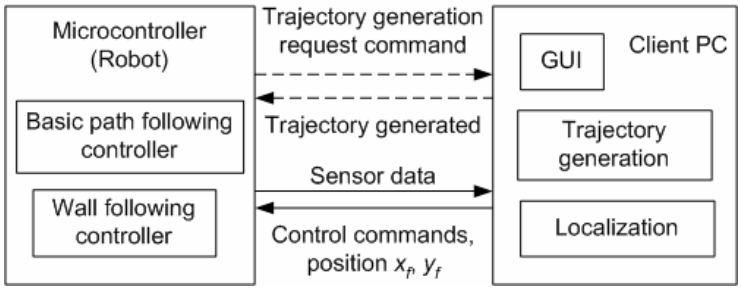


Figure 4.12: Architecture of data communication.

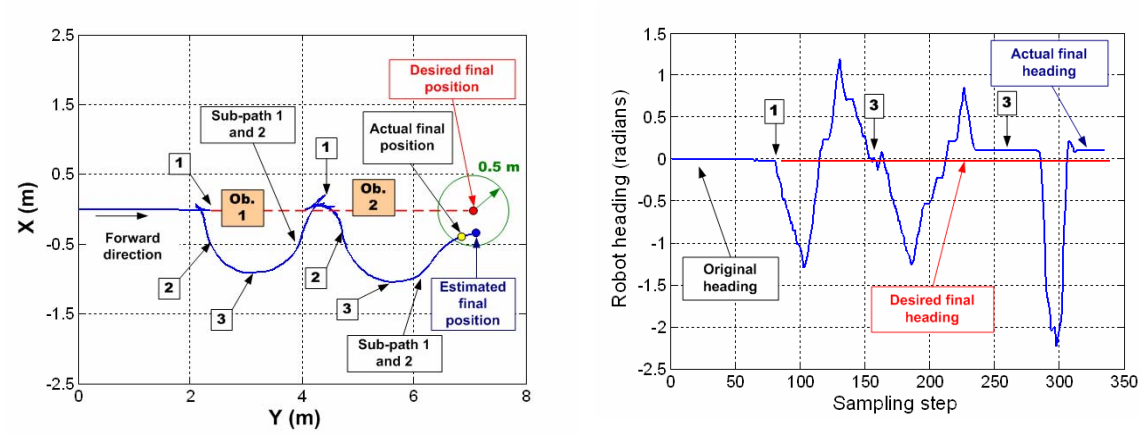
4.3.5 Experimental results

The experiment was performed on-line. The operator assigned a 7 meters line path to the robot. Since the path command is a line path, the desired heading is zero radian and the robot's desired final position is at 7 meters facing forward, the original path as shown as a dashed line in Figure 4.13a. The solid line represents the robot driven path and the position numbers are indicated as defined in Figure 4.11.

The robot started at (0, 0), meets the 1st obstacle at position 1 (0, 2.4), did wall following, stopped at position 2 (-0.5, 2.4), and started the trajectory following at position 3 (-0.9, 3.1) but it met the 2nd obstacle at position 1 (0.2, 4.4). There, the robot restarted the process from position 1 and it finally reaches the destination.

As shown in the figure, the final positions are classified into three types: the desired position, the actual position, and the estimated position. The desired final position is belonging to the original path. The actual position is the real position measured on the ground but the estimated position is obtained from the robot localization. According to the robot localization, the robot stopped at the estimated final position (-0.3, 7.0). In fact, the actual final position was at (-0.4, 6.8). The difference between the actual and estimated position occurred from the accumulated errors in robot localization. However, both the estimated and actual positions lied within the radius of 0.5 meters around the desired final position. During the process, the robot was two times at the position 3. Therefore, the robot heading was adjusted two times into the final desired heading as shown in Figure 4.13b. At the destination, the error in final heading was 0.13 radians only.

In the next chapter, the relative robot localization technique exploited in this section is explained and also the experimental result of the autonomous wall following from section 4.1.3 is presented.



(a) (b)

Figure 4.13: Result of the path following control: robot position; (b) robot heading.

5 Relative Localization using a Nonlinear Dynamic Model

As mentioned in section 4.3, the path following strategy exploits robot localization. Here, the chapter explains the robot's localization which uses a nonlinear dynamic model and the discrete extended Kalman filter. The robot's nonlinear dynamic model and the discrete extended Kalman filter are described in Sections 5.1 and 5.2, respectively. Section 5.3 explains the calculation of the robot's real position and heading errors by using its odometer, its gyroscope, and compass sensors. The final section presents and discusses the experimental results.

5.1 Robot modelling

The first step in model based localization is to develop a suitable mathematical model for the robot. This robot is a four wheel vehicle with the driving principle of a car, where the front wheels do the steering and the rear wheels propel the car forward. The nonlinear dynamic model represents the robot's dynamic movement, including its nonlinear characteristics which come from the side force at the wheels. This property, the side force at the wheels, comes most into effect when the car moves along a curved path. The following section introduces the nonlinear dynamic model and the application of the model for the mobile robot subsequently.

5.1.1 The nonlinear dynamic model

The nonlinear dynamic mathematical model for a four wheel vehicle was conceived of as a single track model describing transverse and longitudinal dynamics, neglecting roll and pitch angles and comprising front and rear wheels to one fictitious wheel [RIE 40] [MAY 91].

Figure 5.1 shows the dynamic variables of such a vehicle as follows:

- the yaw angle ψ, vehicle orientation,
- the yaw velocity ψ', the first derivative of the yaw angle,
- the longitudinal velocity v,
- the sideslip angle β,
- the actual position X and Y of the center of gravity in Cartesian coordinates,
- the front side force S_v and the rear side force S_h,
- the rear longitudinal forces H resulting from the driving motor,

- the steering angle δ_v.

In addition, the model has the following constants: vehicle mass m, moment of inertia θ and the distances l_v (l_h) between the front (rear) wheels and the longitudinal axes of the car.

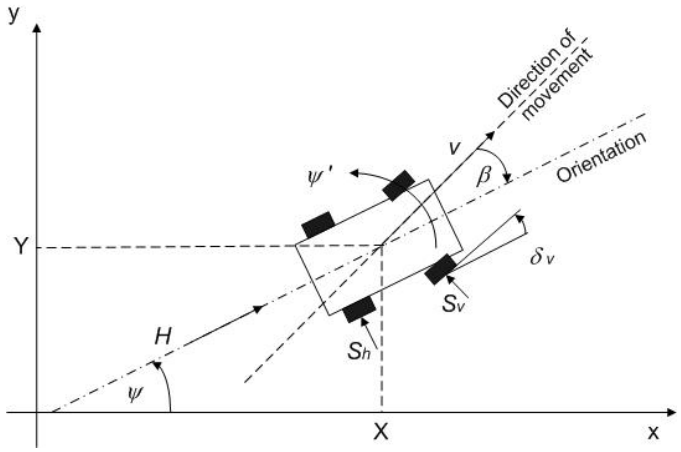


Figure 5.1: Dynamical variables of the vehicle.

By using on the balance of the forces acting on the vehicle in the longitudinal and lateral directions, the torques and the kinematic conditions, the nonlinear dynamic vehicle model is presented as follows:

$$\dot{\beta} = \dot{\psi} - \frac{1}{mv}\{(H - T)\sin\beta + S_h \cos\beta\} - \frac{1}{mv}\{S_v \cos(\beta + \delta)\} \tag{5.1}$$

$$\dot{\psi} = \psi' \tag{5.2}$$

$$\ddot{\psi} = \frac{1}{\theta}(S_v l_v \cos\delta - S_h l_h) \tag{5.3}$$

$$\dot{v} = \frac{1}{m}\{(H - T)\cos\beta - S_h \sin\beta\} - \frac{1}{m}\{S_v \sin(\beta + \delta)\} \tag{5.4}$$

The front and rear side forces S_v and S_h of the vehicle depend on the slip angles α_v and α_h, while α_v depends itself also on the steering angle δ_v. The nonlinear functions Γ_v and Γ_h determine the dynamics in the tires as follows:

$$S_v = \Gamma_v(\alpha_v) \tag{5.5}$$

$$S_h = \Gamma_h(\alpha_h) \tag{5.6}$$

The behavior of the wheels and the tires has been taken into consideration to represent a characteristic line, by applying these functions. This characteristic line includes limitations and descending behavior for high values in the argument. As shown in Figure 5.2, the functions are approximated by three straight lines describing the dependence of side force values S on their argument α. For low arguments, a nearly proportional ascending of the side force can be recognized, while beyond the value α_{max} the side force is descending. At α_{max} the value for the corresponding side force reaches its maximum. The area below α_{max} is called the ascending part, while the area where the side force descends is called the descending part of the characteristic line. As a consequence of the gradual inverse dynamics, the car begins to skid when driving along the descending part. During a normal maneuver without skidding, every wheel of the vehicle is working in the ascending area of the characteristic line. This nonlinear characteristic coming from automotive technology is also applicable to a car-like mobile robot in an indoor environment over dry, flat floors.

As here, high absolute values for α_v and α_h will never come up, and the functions Γ_v and Γ_h can be simplified to the amplification factors c_v and c_h. When this is true, then (5.5) and (5.6) result in

$$S_v = c_v \alpha_v = c_v \left(\beta - \frac{l_v}{v} \dot{\psi} + \delta \right) \tag{5.7}$$

$$S_h = c_h \alpha_h = c_h \left(\beta + \frac{l_h}{v} \dot{\psi} \right) \tag{5.8}$$

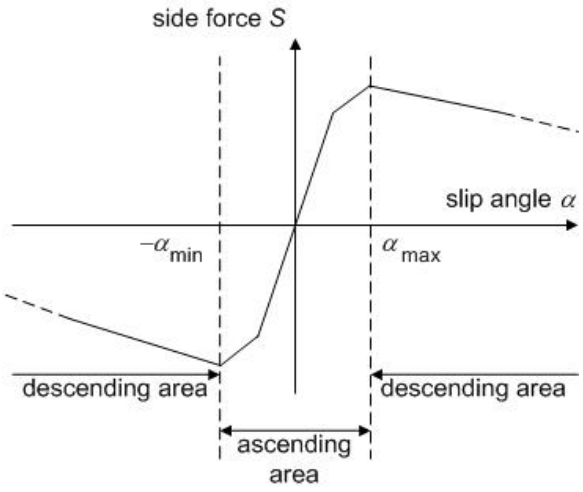


Figure 5.2: Characteristic line Γ of the wheels and tires.

5.1.2 Model realization

According to the calculations, the moment of inertia is equal to 0.169167 kg·m^2. The side force constants c_v and c_h are set from experiments. When the steering angle δ_v is constant and the driving motor force H is also constant, the robot drives on a circular path with a constant speed and the robot's position could be externally recorded by a V-scope^{TM} positioning sensor. The data were exploited with the model equations and MATLAB Simulink to obtain the side force constant. The results are c_v = 168450 and c_h = 152290. The length from the center of gravity to the front wheels is l_v = 0.15 meter and to the rear wheels l_h = 0.15 meter. The mass of the robot is m = 7 kg, including the batteries. Because the surface of the robot is small, and as the robot drives with low speed, it was not necessary to calculate the air resistance, which is negligible. The process of model identification is explained in detail in [TIM 04].

5.2 The Discrete Extended Kalman Filter (EKF)

The Kalman filter (KF) is a model-based sensor fusion technique used in many applications. The KF is also exploited for the measured data of PMD camera in the next chapter. Here, we explain how EKF is implemented. The discrete EKF [BRO 83] [WEL 02] is derived from the KF for application of discrete system. The details are explained in the following subsections.

5.2.1 A general discrete EKF

First, it must be assumed that a nonlinear process has a state vector $x \in \Re^n$ and is governed by the non-linear stochastic differential equation

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}), \tag{5.9}$$

with a measurement $z \in \Re^m$ that is

$$z_k = h(x_k, v_k), \tag{5.10}$$

where w_k and v_k represent the process and measurement noise. The linearized model equations from (5.9) and (5.10) are

$$x_k \approx \tilde{x}_k + A(x_{k-1} - \hat{x}_{k-1}) + w_{k-1}, \tag{5.11}$$

$$z_k \approx \tilde{z}_k + H(x_k - \tilde{x}_k) + v_k, \tag{5.12}$$

where x_k and z_k are the actual state and measurement vectors, $\tilde{x}_k$ and $\tilde{z}_k$ are the approximated state and measurement vectors, and $\hat{x}_k$ is an a posteriori estimate of the state at step *k*. Note that noises are neglected for state and measurement vectors due to the fact that the individual values of w_k and v_k at each time step are unknown. Thus, the linearized state transition matrix is represented without noises. The Jacobian matrix of the partial derivatives of *f* with respect to *x* is

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_k, 0), \tag{5.13}$$

where the state vector *x* is

$$x = [\beta \quad \psi \quad \psi' \quad v]^T, \tag{5.14}$$

and the Jacobian matrix of partial derivatives of *h* with respect to *x* is

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\tilde{x}_k, 0), \tag{5.15}$$

where *h* is

$$h = [x_1 \quad x_2 \quad x_3 \quad x_4]^T. \tag{5.16}$$

The filtering process starts with the initialization of all state variables and matrices. It is assumed that the process and measurement noises are Gaussian with a mean of zero and are constant throughout the process. The a priori estimate state variable $\hat{x}_k^-$ and the a priori estimate error covariance P_k^- are as follows at time step *k*,

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k, 0), \tag{5.17}$$

$$P_k^- = A_k P_{k-1} A_k^T + Q, \tag{5.18}$$

where *Q* is the process noise constant matrix. After that the Kalman gain K_k, the a posteriori estimate state variable $\hat{x}_k$, and the a posteriori estimate error covariance P_k are calculated as follows:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R)^{-1}, \tag{5.19}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-, 0)), \tag{5.20}$$

$$P_k = (I - K_k H_k) P_k^-, \tag{5.21}$$

where R is the measurement noise covariance.

5.2.2 Calculating the discrete EKF.

The inputs needed from the robot model are the steering angle δ and the driving motor force H. For the discrete EKF, let the input vector at time step k be

$$u_k = [\delta_k \quad H_k]^{\mathrm{T}}, \tag{5.22}$$

and let the measurement vector be

$$z_k = [z_{\beta,k} \quad z_{\psi,k} \quad z_{\psi',k} \quad z_{v,k}]^{\mathrm{T}}, \tag{5.23}$$

where $z_{\beta,k}$, $z_{\psi,k}$, $z_{\psi',k}$, and $z_{v,k}$ are the measured values of the state variables at time step k. As the robot drives very slowly and, thus, the sideslip angle, which is difficult to identify, is of minor importance, the measurement value of the sideslip angle is set to zero for all time steps k. As for the system model in (5.1–5.4), the discrete system is first obtained using Euler's method [FRA 98]. After that the linearized model equations are derived for the Jacobian matrix by using (5.13) and (5.14) as follows:

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial \beta} & \frac{\partial f_2}{\partial \beta} & \frac{\partial f_3}{\partial \beta} & \frac{\partial f_4}{\partial \beta} \\ \frac{\partial f_1}{\partial \psi} & \frac{\partial f_2}{\partial \psi} & \frac{\partial f_3}{\partial \psi} & \frac{\partial f_4}{\partial \psi} \\ \frac{\partial f_1}{\partial \psi'} & \frac{\partial f_2}{\partial \psi'} & \frac{\partial f_3}{\partial \psi'} & \frac{\partial f_4}{\partial \psi'} \\ \frac{\partial f_1}{\partial v} & \frac{\partial f_2}{\partial v} & \frac{\partial f_3}{\partial v} & \frac{\partial f_4}{\partial v} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \tag{5.24}$$

where

$$a_{11} = 1 - \frac{st}{mv_k}\{H\cos\beta_k + c_h\cos\beta_k + c_v\cos(\beta_k + \delta_k)\} + \frac{st}{mv_k}\left\{c_h \sin\beta_k\left(\beta_k + \frac{l_h}{v_k}\dot{\psi}_k\right)\right\}$$

$$+ \frac{st}{mv_k}\left\{c_v \sin(\beta_k + \delta_k)\left(\beta_k - \frac{l_v}{v_k}\dot{\psi}_k + \delta_k\right)\right\},$$

$$a_{13} = st - \frac{h}{mv_k}\left\{\frac{c_h l_h \cos\beta_k}{v_k} - \frac{c_v l_v \cos(\beta_k + \delta_k)}{v_k}\right\},$$

$$a_{14} = \frac{st}{mv_k^2}\{H\sin\beta_k + c_h\beta_k\cos\beta_k + c_v\beta_k\cos(\beta_k + \delta_k)\} + \frac{st}{mv_k^2}\{\delta_k c_v\cos(\beta_k + \delta_k)\}$$

$$+\frac{st}{mv_k^3}\{(c_h l_h \dot{\psi}_k \cos\beta_k)-(c_v l_v \dot{\psi}_k \cos(\beta_k+\delta_k))\},$$

$$a_{22}=1,$$

$$a_{23}=st,$$

$$a_{31}=\frac{st}{\theta}(c_v l_v \cos\delta_k - c_h l_h),$$

$$a_{33}=1+\frac{st}{\theta}\left[-\frac{c_v l_v^2}{v_k}\cos\delta_k-\frac{c_h l_h^2}{v_k}\right],$$

$$a_{34}=\frac{st}{\theta}\left[\frac{c_v l_v^2 \dot{\psi}_k}{v_k^2}\cos\delta_k+\frac{c_h l_h^2 \dot{\psi}_k}{v_k^2}\right],$$

$$a_{41}=\frac{st}{m}\left\{H\sin\beta_k - c_h\cos\beta_k\left(\beta_k+\frac{l_h}{v_k}\dot{\psi}_k\right)\right\}-\frac{st}{m}\left\{c_v\cos(\delta_k+\beta_k)\left(\beta_k-\frac{l_v}{v_k}\dot{\psi}_k+\delta_k\right)\right\}$$

$$-\frac{st}{m}\{c_v\sin(\delta_k+\beta_k)+c_h\sin\beta_k\},$$

$$a_{43}=\frac{st}{m}\left[-\frac{c_h l_h}{v_k}\sin\beta_k+\frac{c_v l_v}{v_k}\sin(\delta_k+\beta_k)\right],$$

$$a_{44}=1+\frac{st}{m}\left[\frac{c_h l_h}{v_k^2}\dot{\psi}_k\sin\beta_k-\frac{c_v l_v}{v_k^2}\dot{\psi}_k\sin(\delta_k+\beta_k)\right],$$

$$a_{12}=a_{21}=a_{24}=a_{32}=a_{42}=0.$$

Similarly, derived from (5.15) and (5.16), the Jacobian matrix of measurement *H* is a diagonal matrix with the element value of 1. The process and measurement noise covariance matrices *Q* and *R* are tuned off-line in the simulation of the appropriate value and kept constant during the iterative process. The selected values are $1x10^{-6}$ for all diagonal matrix elements. These parameters represent the uncertainty of the process and measurement. The improper value of these matrix elements result in a wrong estimated value and rapid divergence from the true measurement.

5.3 Calculating the robot's position and heading.

As shown in Figure 5.3, three different paths are implemented by using measurement data from the odometer, the gyroscope, and the compass. The robot's position and heading based on the odometer's data gives the non-model based path explained in section 5.3.1, whereas the other two path estimations are derived by using

the Kalman filter and the nonlinear dynamic model explained in section 5.3.2.

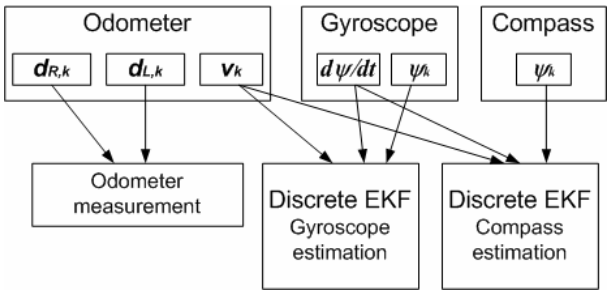


Figure 5.3: Architecture of the robot localization system.

5.3.1 Odometer position and heading calculation.

The position $x_{odo,k}$, $y_{odo,k}$ and heading $\psi_{odo,k}$ at step k are calculated from

$$x_{odo,k} = x_{odo,k-1} + \left(\frac{\Delta d_{R,k} + \Delta d_{L,k}}{2} \right) \cos \psi_{odo,k}, \tag{5.25}$$

$$y_{odo,k} = y_{odo,k-1} + \left(\frac{\Delta d_{R,k} + \Delta d_{L,k}}{2} \right) \sin \psi_{odo,k}, \tag{5.26}$$

$$\psi_{odo,k} = \psi_{odo,k-1} + \tan^{-1} \left(\frac{\Delta d_{R,k} - \Delta d_{L,k}}{W} \right), \tag{5.27}$$

$$\Delta d_{R,k} = d_{R,k} - d_{R,k-1}, \tag{5.28}$$

$$\Delta d_{L,k} = d_{L,k} - d_{L,k-1} \tag{5.29}$$

where $\Delta d_{R,k}$ is the difference between the distances driven by the right wheel between the time steps k and k-1 as shown in Figure 5.4. Note that a similar calculation is applied to the difference between the distances left $\Delta d_{L,k}$. Here, $d_{L,k}$ represents the distance already driven by the left wheel at time step k, while $d_{R,k}$ is the distance driven by the right wheel. These distances driven from the beginning of the ride can be computed from the amount of pulses from the odometer on each wheel. The wheel base W is the distance between the center of the left wheel to the center of the right wheel and is equal to 23.5 cm.

5.3.2 Position and heading estimation using gyroscope and compass.

We calculate the robot x_k and y_k positions at time step k of gyroscope and

compass by using the estimated state variables from discrete EKF. The gyroscope estimation uses the relative yaw angle of the gyroscope whereas the compass estimation uses absolute yaw angle of the compass as ψ_k in

$$x_k = st \cdot v_k \cos(\psi_k - \beta_k) + x_{k-1}, \tag{5.30}$$

$$y_k = st \cdot v_k \sin(\psi_k - \beta_k) + y_{k-1}, \tag{5.31}$$

where v_k, ψ_k, and β_k are posterior estimated state variables as in (5.20). The estimated robot heading is ψ_k and st is the time step.

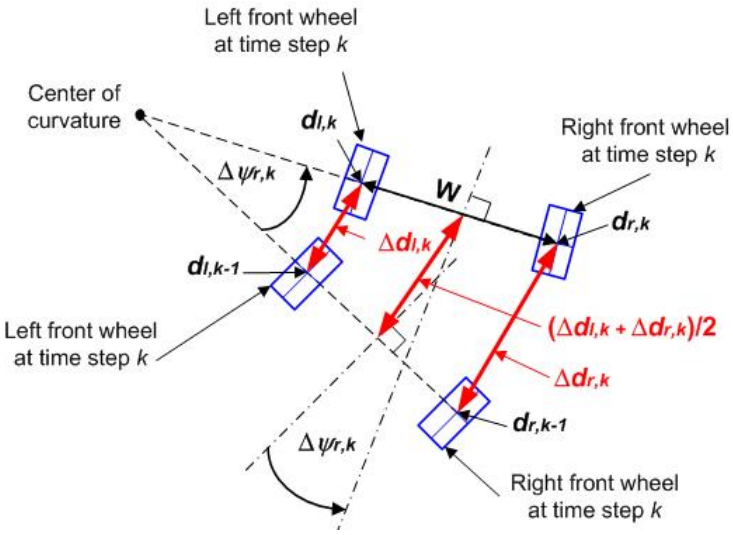


Figure 5.4: Variables based on odometer.

5.4 Experimental results

The localization experiments were performed in an indoor environment and the sensor data was collected for off-line plots using MATLAB. All of the graph scales have units in meter and a data sampling rate of 0.2 sec. Though only the experimental results of the off-line positioning were presented in this chapter, the on-line experiment was already presented for the path following control described in section 4.3. Here, the experiments are designed to test the performance of localization when the robot moves straight on, in a curve path, and a combination of both. Several path types are exploited.

5.4.1 Description of exploited path types.

Rectangular path: By manually tele-operated joystick control, the robot begins from 0 heading, drives straight on, turns at corners, and ends up at final heading of 2π.

Line path: By manually tele-operated joystick control, the robot drives straight forward, stops at 15 meters, and drives backward to its original position.

Wall path: The robot drives autonomously along the wall by using the wall following controller explained in section 4.1.3. The wall has a 15 meter by 15 meter square-shape.

5.4.2 Types of measurements.

As already mentioned, in order to improve the quality of the measurement signals, the data coming from the gyroscope as well as the data from the compass was subsequently processed by the Kalman filter using the measured velocity signal from the odometer. With respect to Figure 5.3, three cases of measurement including the signal estimation by the Kalman filters are compared as follows:

Odometer: Here, the data coming from the odometer is not updated by the Kalman filter. By using (5.25) - (5.27), the relevant odometer data not only contains the average of the wheel movements but also the difference between the movements of the left and right wheels in order to get information about the robot's changed orientation. The dot line represents this *Odometer*'s path.

Gyroscope: The yaw angles measured by the gyroscope are applied as inputs for Kalman filter measurement updates. The positions are obtained from (5.30) and (5.31) and are shown as solid line. Please note that here the odometer only provides the measurement value for the velocity.

Compass: Similarly to the gyroscope measurements, the compass yaw angle measured by the compass is applied as an input for the Kalman filter, such that the position estimation from (5.30) and (5.31) can be represented by a dash-dot line. Please note that here as well the odometer only provides the measurement value for the velocity.

5.4.3 Test results of several path types.

Rectangular path: Due to section 5.4.2, Figure 5.5 presents the three kinds of results for the rectangular path. The robot drives from position (0, 0) with a 0 radian heading in a clockwise direction and stops at a 2π radians heading. Regarding the path

measured only by the odometer measurement (dotted line), suddenly at (6,-3), a slippage in the wheels occurs. The estimated path using the gyroscope estimation (solid line) is described by the solid line. Its final position is very close to the real final position. The estimated position using the compass estimation (dash-dotted line) shows a deviation in the robot's heading along the path. This deviation results in the big error of the final heading obtained from the compass.

Wall path: Figure 5.6 shows the wall-path. Here, the measured final position using only the odometer measurement (dotted line) has a large error.

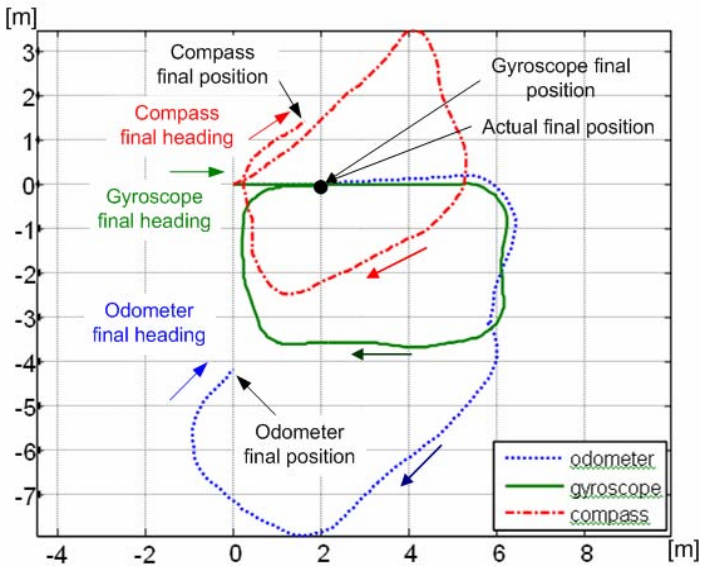


Figure 5.5: Rectangular path estimated positions.

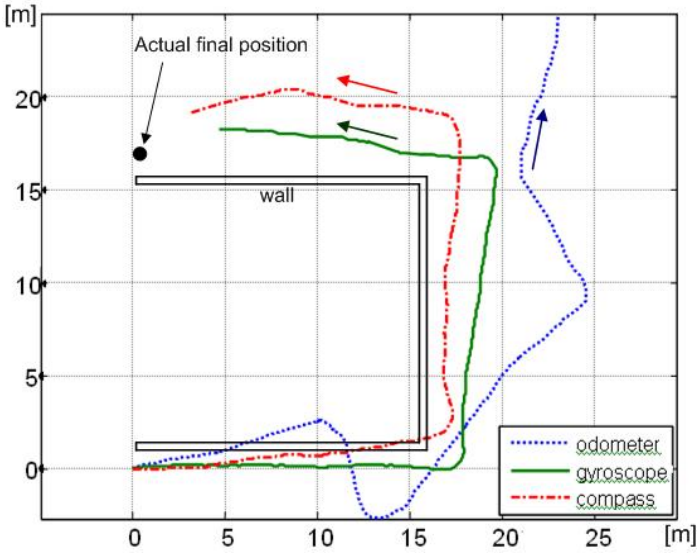


Figure 5.6: Wall path estimated positions.

Improvement using the gyroscope estimation (solid line) estimation shows a drift in some trials, whereas improvement using the compass estimation (dash-dotted line) estimator provides quite constant results for all trials.

Line path: As shown in Figure 5.7, the expected final position of this path is the same as the starting position. Regarding the odometer measurement (dotted line), both, the results for the path and for the final position are not satisfying. The estimated path using the gyroscope estimation (solid line) is in the lateral direction quite good; however, the final position is not that close to the actual final position. Regarding the compass estimation (dash-dotted line), the final position is very exact whereas the trajectory of the path is estimated with certain errors, especially at the coordinates (4, 1). Due to this problem, the robot's heading is also shown in Figure 5.8. The measured values of the yaw angle derived from data from the compass sensor are extremely large at the 55^{th} and at the 250^{th} sampling steps. However, as mentioned, the final compass estimation for the robot's position is in the end close to the actual final position.

It must be noted that this phenomenon doesn't occur in every test. It must be assumed that this error is caused probably from the magnetic field of a power cable in the building.

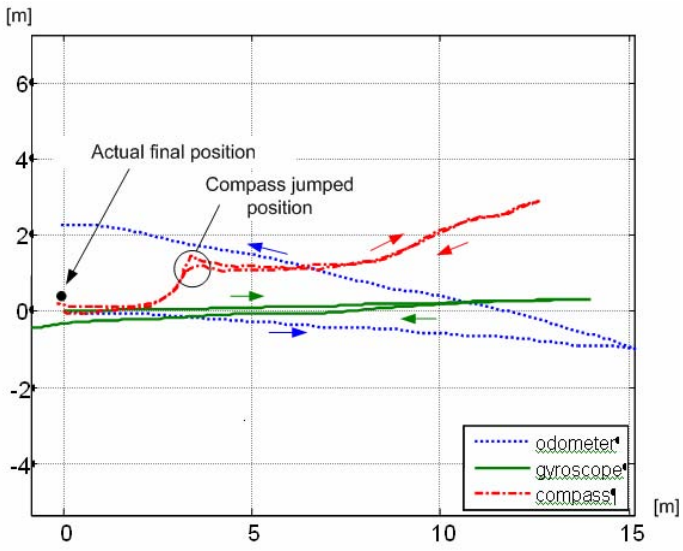


Figure 5.7: Line path estimated positions.

5.4.3 Average errors

Several driving tests were performed for every path type. The real final position determined for each driving test was compared to the position value measured by the

on-board sensors to calculate the resulting errors. The average error values result from the averages of the respective errors of all similar driving tests.

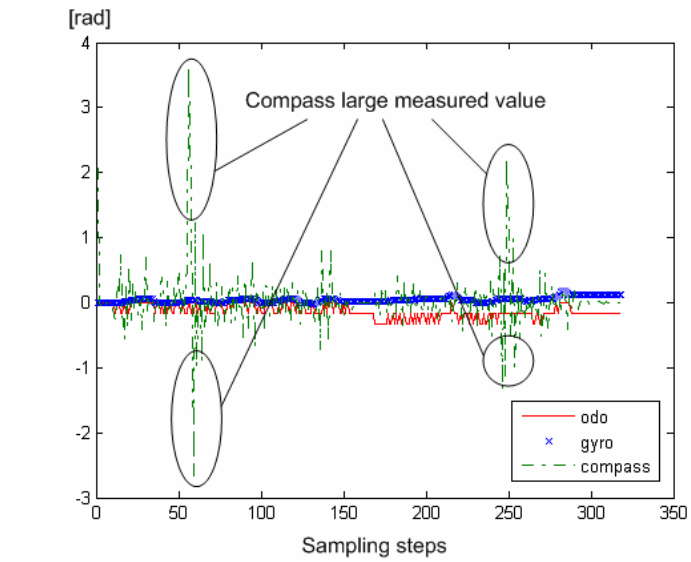


Figure 5.8: The estimated headings of the line path.

The average errors in position and heading are summarized in Tables 5.1 and 5.2, respectively. Several tests were performed for each path type. The position error e_{pos} is the error in measurement between the actual final position and the estimated final position. Based on the actual and estimated values for the yaw angle, the heading error $e_{heading}$ is the error of robot heading at the final position. These errors are calculated by

$$e_{pos} = \sqrt{\bar{e}_x^{\,2} + \bar{e}_y^{\,2}}\,, \tag{5.32}$$

$$\bar{e}_x = \frac{\sum_{i=1}^{n}\left(x_{estimate,i} - x_{actual,i}\right)}{n} \tag{5.33}$$

$$\bar{e}_y = \frac{\sum_{i=1}^{n}\left(y_{estimate,i} - y_{actual,i}\right)}{n} \tag{5.34}$$

$$e_{heading} = \frac{\sum_{i=1}^{n}\left(\psi_{estimate,i} - \psi_{actual,i}\right)}{n}, \tag{5.35}$$

where $x_{estimate,i}$, $y_{estimate,i}$ and $\psi_{estimate,i}$ are the estimated final position and heading of test number i. $x_{actual,i}$, $y_{actual,i}$ and $\psi_{actual,i}$ are the real final values for the actual position and

heading of test number *i*, respectively. $\bar{e}_x$ and $\bar{e}_y$ are the mean of error in *x* and *y* direction, *n* is the total number of tests. Note that the units are meter for position errors and degree for the heading errors.

In Table 5.1, the average of the gyroscope estimation final position errors are not the largest in all of the trajectories. It can be seen that the gyroscope estimation for the final position value is in most cases not as good as the compass estimation. However, due to section 5.4.3, it is stated that the gyroscope estimation provides the most exact path.

As Table 5.2 shows, the gyroscope estimation combined with discrete EKF provides very good results. Especially, for the line path, the pure odometer measurement is slightly better. However, a large error in the odometer's position is found, when using the wall path as an example. This large error occurs due to the slippage in the wheels and does not always occur in all driving tests.

On top of that, it must be stated that once in a while drift problems can come up, when using the gyroscope. It is assumed that these problems can especially sometimes accumulate, when the ride takes a long time or when many bends are located along the path.

An improvement of accumulated errors in relative localization is to exploit the vision-based absolute localization technique since this technique gives directly absolute robot position and heading without referring to the start position. The next chapter presents a localization technique called the position calibration.

Table 5.1: Average position errors (e_{pos}).

Path types	Odometer	Gyroscope	Compass
Rectangular path	0.71	0.17	0.33
Wall path	15.09	5.44	3.35
Line path	0.53	1.02	0.27

Table 5.2: Average heading errors ($e_{heading}$).

Path types	Odometer	Gyroscope	Compass
Rectangular path	-23	-1	14
Wall path	-58	-10	-52
Line path	5	6	61

6 Position Calibration using 3D Vision and Artificial Landmark

As in the previous chapter, the robot relative localization contains accumulated errors. As to correct these errors, this chapter presents the position calibration technique using 3D vision and artificial landmark. This technique can be broken down into three sequential procedures: design of the artificial landmark introduced in Section 6.2, landmark recognition, described in Section 6.3, the position prediction and update, described in Section 6.4. The on-line experiment of using combination of the relative and absolute localization techniques is also presented. The first section of this chapter begins by providing an investigation of the characteristics of the measurement of the 16x16 pixel PMD camera.

6.1 The measurement characteristics of the 16x16 pixel PMD camera.

The characteristics of measurement data is a factor for selection of an appropriate technique for landmark recognition. Therefore, our work began with the investigation of the data and its noise characteristics using a statistic approach. We used a flat whiteboard for calibration at a distance of 1 meter and set the integration time of the PMD camera to 5000 us.

Figure 6.1 shows the plot of 100 measured values of the 8^{th} row pixel and of the 1^{st} to the 16^{th} column pixel (c1 - c16). The measured values at each pixel have random noises. It is interesting to observe the mean and standard deviation (STD) of distance values since these values represent the accuracy and uncertainty of measurements of the camera. Therefore, we place this whiteboard at other distance positions and select 4 pixels out of 256 pixels as sample pixels. Let the r^{th} row pixel and the c^{th} column pixel be called pixel (r, c). These selected pixels are (8, 1), (8, 2), (8, 3), and (8, 4).

The mean of distance values at 0.5 - 2.4 meters are plotted as in Figure 6.2a. At the distance of 0.5 to 1.6 meters, the mean values are close to or smaller than the actual distance value. At the distance farer than 1.6 meter, the difference between the mean values and the actual distance values are larger. Figure 6.2b shows the STD values. At 1.6 meters, the STD values are smaller than 50 mm. At a longer distance, the STD values are larger. At 2.4 meters, the maximum STD value among these pixels is 170 mm. At this distance, the difference between the maximum and minimum STD values is 70 mm.

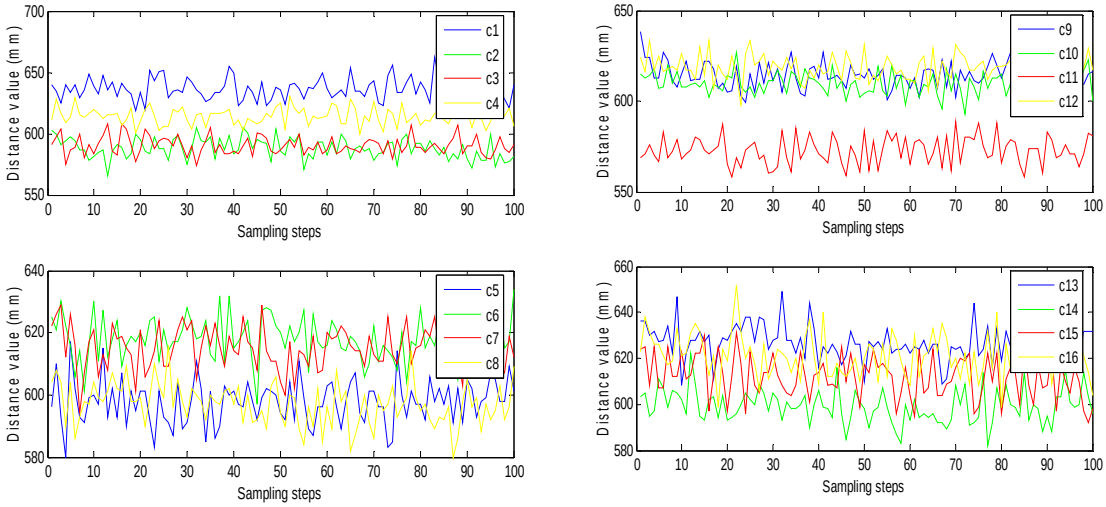
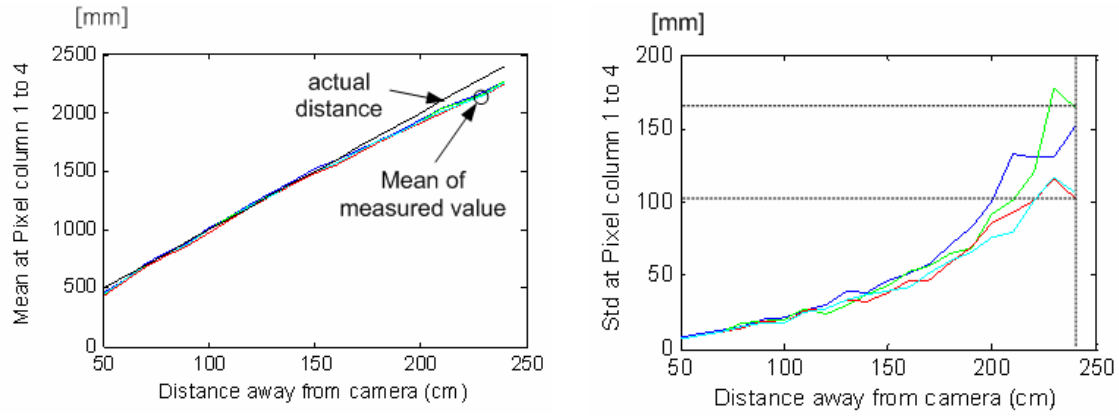


Figure 6.1: The measured values of the 8th row pixel and at each column pixel (c1 – c16).



(a) (b)

Figure 6.2: Mean and STD at the 8th row pixel and at the 1st to 4th column pixel: (a) mean values; (b) standard deviation (STD) values.

Once the distance increases beyond 1.0 meter, the standard deviation increases, as mentioned earlier. Here, the distribution of the measured value at close range, a distance of 0.2 up to 1.0 meter is further investigated. Table 6.1 presents the 3D images plotted by using the mean and standard deviation of all 256 pixels at distances of 1.0, 0.50, and 0.20 meters, respectively. Note that colors represent the distance values. The smooth surface in an image of mean values means that the measured values among 256 pixels are the same and the smooth surface in an image of the STD values means that there is equally distribution of measured values among 256 pixels.

When we look at the mean and STD images at 0.5 meter, the distribution of mean values is random and the surface is non-smooth, whereas the image of STD values is very smooth. At 1.0 meter, middle pixels have smaller STD values than outer pixels. In the opposite, at 0.2 meter, middle pixels have larger STD and mean values than outer pixels. These are the focus property of the optical objective. The infrared light reflects more strongly among the focused pixels than in the pixels around the focus area, which can be detected to be a circular area. Due to this property, measurements taken when the object is too near to the camera cannot be used. It should be noted, however, that the measurement results can be different in terms of integration time, object colour and type of material.

6.2 The design of the artificial landmark.

In mobile robot applications, artificial landmark recognition is designed according to its purpose and according to what is most convenient for the user. The problem of object recognition is simplified by simultaneously designing a landmark suited to the object recognition strategy.

In order that this be possible, the landmark is divided into two parts, an upper part and a lower part. The upper part is always symmetrical in shape, such as the cylinder shown in Figure 6.3. This part must be mounted on top of the lower part according to its own centre of gravity in order that the image taken of this upper part is view invariance. The lower part of the landmark is always rectangular in shape so that it is possible to calculate its distance from the robot and rotational angle position. Note that the background is the pixels that contain the measured distance value of the object behind the landmark. Details of how we get filtered image and how model image are generated are explained later. Here, we continue with the landmark design.

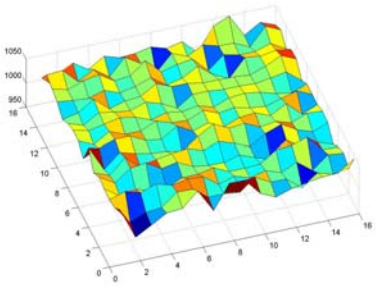
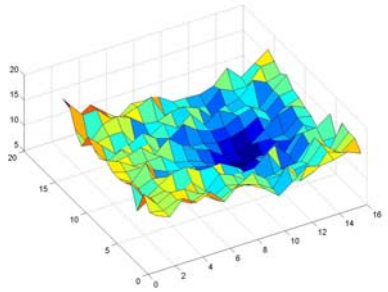
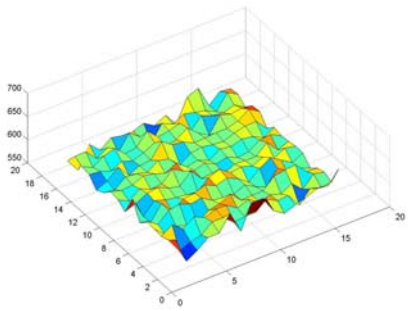
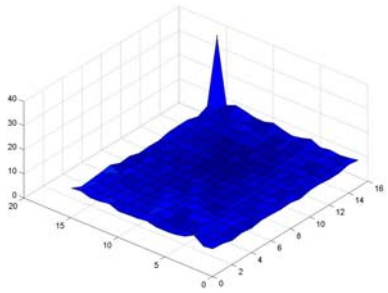
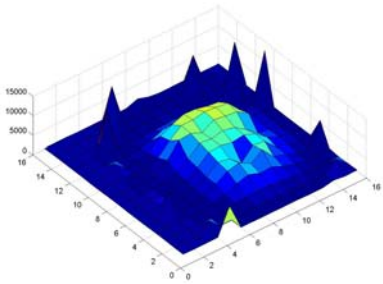
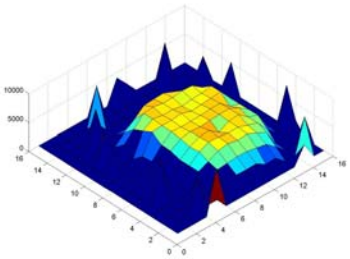
6.2.1 Lower part of the landmark.

When the rectangular box is rotated, the angle of rotation is detectable by using the slope of the box boundary and the box width and length. Varying the angle position changes the detected area, as shown in Figure 6.4a.

The measured value and slope are shown as thick solid lines at the landmark's boundary. One edge exist in the camera's range of vision at only some angle positions, e.g. 11.25°, 22.5° and 33.75°. At other angle positions, no edge exists within the

camera's range of vision but slopes have changed visibly, e.g. 0°, 45°, 56.25°, 67.5°, 76.75°, and 90°.

Table 6.1: Mean and standard deviation (STD) at 1.0, 0.5, and 0.2 meters.

Distance	Mean	Standard deviation (STD)
1.0 m		
0.50 m		
0.20 m		

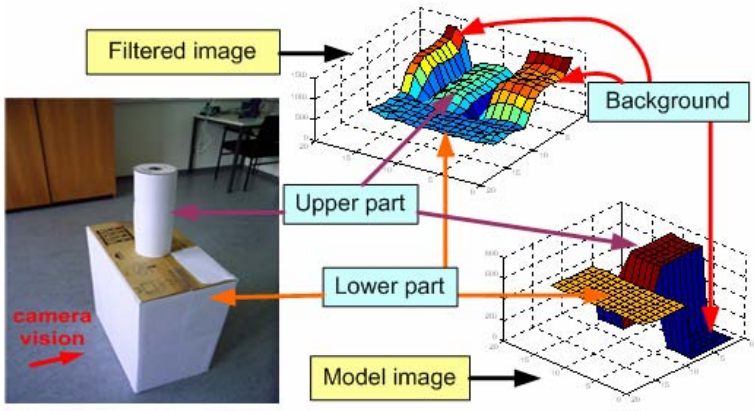
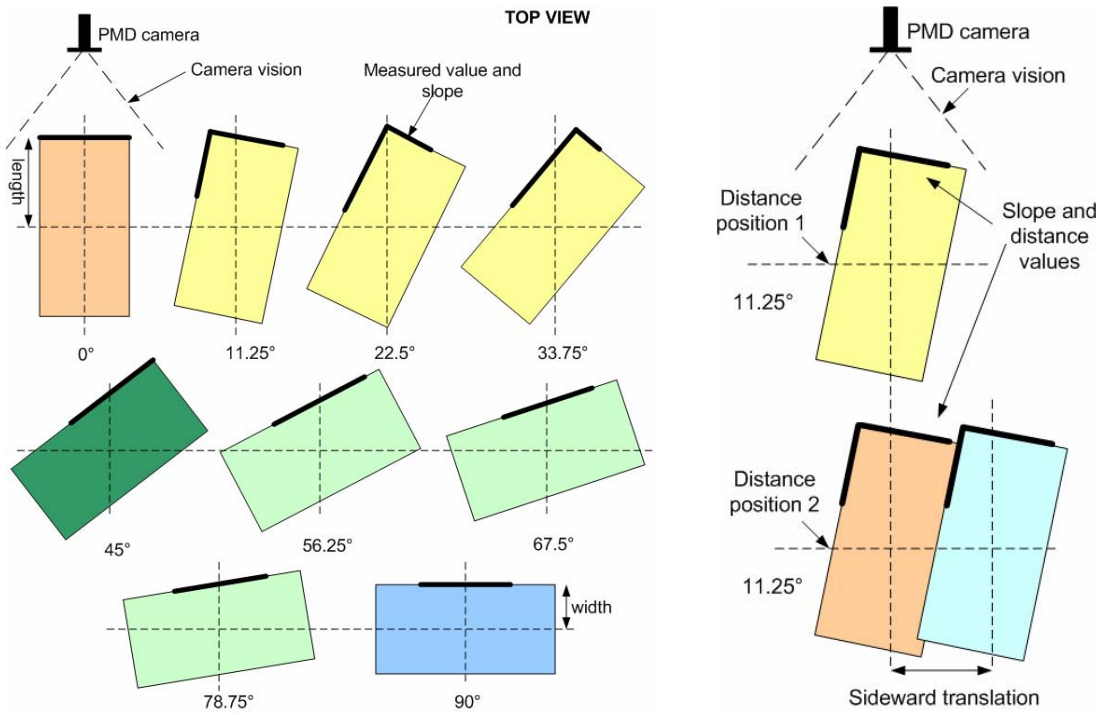


Figure 6.3: An example of the filtered image and the generated model image.

The difference in slope is distinguishable at each angle position and can therefore be set as a criterion for landmark recognition. As shown in the figure, between 0° and 90°, slopes are not differentiable but distance values are differentiable since the width and the length of the rectangular box are not equal. The distance value is therefore another criterion for landmark recognition. Note that the square box is not applicable as a landmark since its width and length are equal.

The reason why we use both the slope value and distance value as criterions is that it is more effective than using only the measured distance value. This advantage can be seen in Figure 6.4b, where two landmarks are at different distance positions but at the same angle position, 11.25°. The slopes of both landmarks are exactly the same no matter the distance position is far away, close, or even there is sideward translation.



(a) (b)

Figure 6.4: The detected surface at various positions: (a) various angle positions; (b) different distance positions.

6.2.2 Upper part of the landmark.

In practice, many landmarks are set within the navigation area. The robot therefore needs to recognize each landmark. Therefore, the upper part of the landmark is designed to have an identity of each landmark. The upper part should have symmetry in shape in order that we get the same view of the object image no matter what angle the

whole landmark is set at. In words, the image of the upper part is view invariance. The geometry of the upper part is also required for convenience in landmark recognition.

The design of the artificial landmarks also takes the choice of material, the landmark's colour and its size into consideration. The material used must reflect enough infrared light to produce a good image and thus it should not be black in colour. White paper was selected as the landmark surface material because it reflects well. The landmark should not be too big or too small in order that the objects lie within the detection area of the camera.

6.2.3 Several designs of 3D artificial landmark.

Examples of the various possible designs in landmark shape are shown in Figure 6.5a. The classes I and II are rectangular boxes which are different in size. They are used as the lower part of the landmark and are called landmark class. There are several symmetrical and geometrical shapes that are applicable for the upper part of the landmark, which distinguishes the landmark type. The figures A to L are sample landmark types. Types A to I are consist of one geometrical part; whereas, types J to L are the combination of two geometrical parts. Figure 6.5b shows two combinations of landmark classes and types. When there are more classes and types available, there is also more variety in landmarks. Further designs as follows can be used:

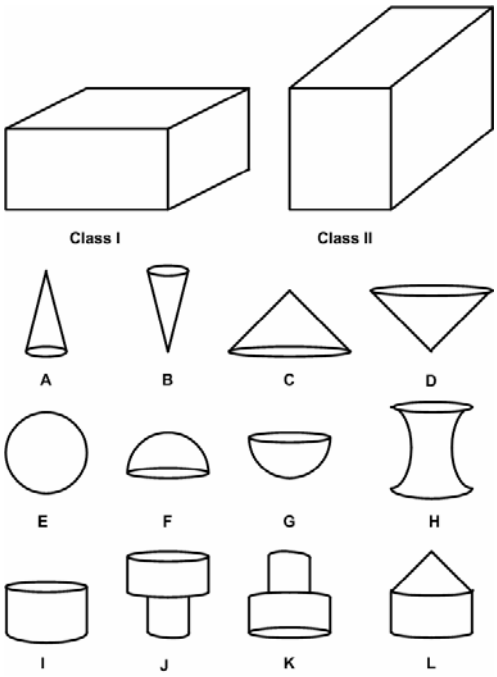
- Size variation: Landmark types of the same shape in different sizes result in different types, e.g. type I with smaller or bigger radius length.
- Shape variation: The variation of the landmark type is possible by combining symmetrical shapes, e.g. the combination of type F and I.

6.2.4 Landmarks for 16x16 pixel PMD camera.

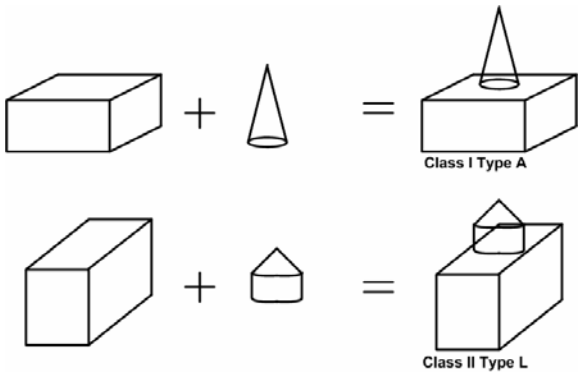
For the 16x16 pixel camera, three landmark types are selected. The first landmark type has a rectangular shape as upper part. That is, the landmark is a tall box. In the opposite, the second landmark type has a short box and is called null shape since the upper part does not exist. The third landmark type is a cylinder.

Table 6.2 shows four designed landmark types called L1, L2, L3 and L4. L1 is a rectangular landmark and L2 is null-type landmark. In fact, L1 and L2 are different only in the height. L3 and L4 are cylinder-type landmarks and are different in the diameter of cylinder. Table 6.2 gives the shapes of the landmarks in diagram form, types of the landmarks, lists the landmarks' sizes. As mentioned earlier, the lower part of the

landmark is used for the recognition of distance position and angle position. If the width of the box is not distinguishable from its length, the distance position cannot be read. The appropriate length and width are about 31 cm and 57 cm, respectively. Note that all four landmark types are in the same class. Therefore, they have the same the lower part. The methodologies used for landmark recognition of the upper and lower part are explained below in the coming sections.



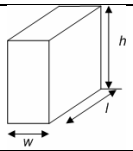
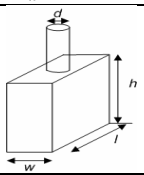
(a)



(b)

Figure 6.5: Landmark examples (a) various classes and types of landmark designs (b) landmark examples from the combinations of the designed landmark classes and types.

Table 6.2: The four types of landmarks: names, shapes, types, and dimensions.

Name	Shape	Type	Dimension: w x l x h x d, (cm)
L1		Rectangular	31 x 57 x 73
L2		Null	31 x 57 x 52.5
L3		Cylinder	31 x 57 x 52.5 x 12.5
L4		Cylinder	31 x 57 x 52.5 x 6.5

6.3 Landmark recognition

When the robot is at any positions referred to the fixed position landmark, by recognizing the distance position and the angle position of landmark, the robot position can be directly calculated as explained later in Section 6.4.1. The landmark recognition problem is to determine the distance position, the angle position and the type of landmark. Landmark recognition is divided into recognition of the upper and lower parts regarding landmark designs. Recognition of the upper part serves to classify landmark types; whereas, recognition of the lower part provides the distance position and angle position of the landmark. These processes start after the image filtering (Section 6.3.1).

The core process in landmark recognition of lower part is shown in Figure 6.6. First, the filtered image is smoothed (Section 6.3.2). After that the smoothed image is further processed for matching preparation: The representative distance values of each column pixel is fitted into a line by using linear regression as described in Section 6.3.5. Meanwhile, the model images are obtained from the model image generation (Section 6.3.3.1) and this image is process by line fitting and matching of upper part (Section 6.3.6.1). The matching results are the distance and angle positions.

The landmark recognition of the upper part is as shown in Figure 6.7. The model image and the smoothed image are detected for edges and matched later. The model image generation and the edge detection are explained in Section 6.3.3.2 and Section 6.3.4, respectively. The method for matching the upper part of smoothed image with the model image is explained in Section 6.3.6.2. The output of the matching process is the landmark type. First of all, the following section explains image filtering.

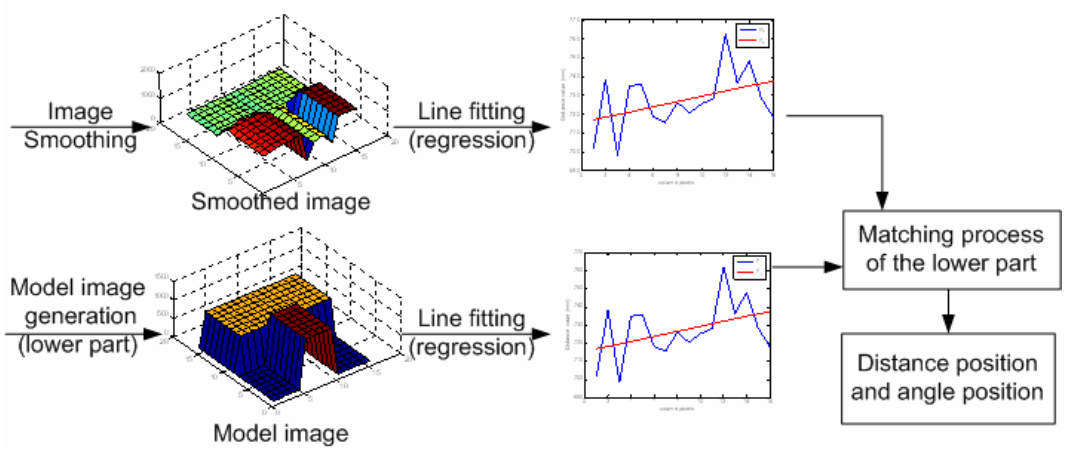


Figure 6.6: The landmark recognition process of the lower part.

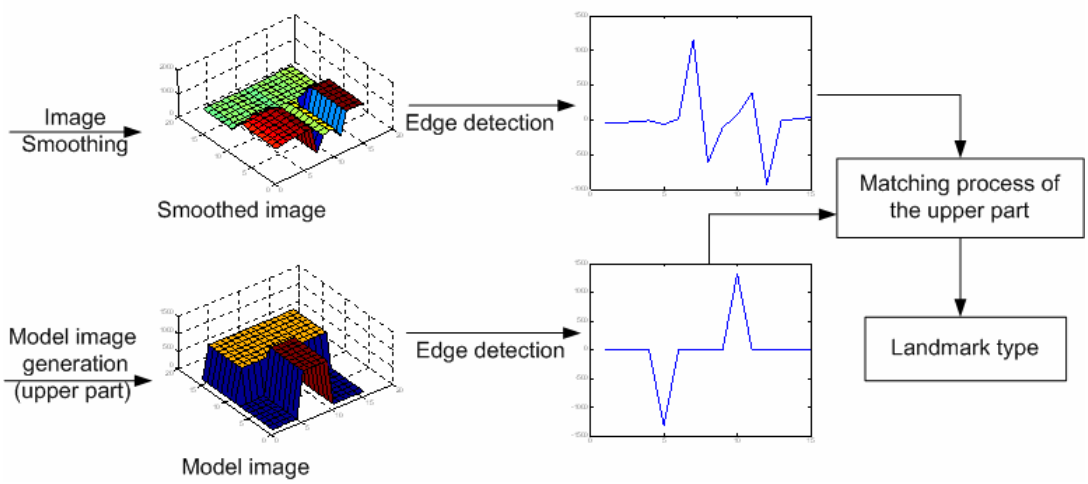


Figure 6.7: The landmark recognition process of the upper part.

6.3.1 Image filtering

The PMD camera used here has 16x16 pixel resolution. The measurement characteristics in Section 6.1 show the PMD measurement data to be noisy, so that this data is a signal that needs to be filtered. We choose the powerful Kalman filtering model to filter the PMD images because Kalman filtering is well known for its high performance. Process noises, measurement noises, and the result from recursive computation yield the convergence of the self-adaptive Kalman gain and estimated state variable [WEL 02].

6.3.1.1 Conducting the Kalman filtering. Two steps of the Kalman filter algorithm are implemented here for estimation and update by using the equations (6.1 - 6.2) and (6.3 - 6.5), respectively.

$$xe_k^- = Axe_{k-1} + Bu_{k-1} \tag{6.1}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{6.2}$$

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{6.3}$$

$$xe_k = xe_k^- + K_k (z_k - H\, xe_k^-) \tag{6.4}$$

$$P_k = (I - K_k H) P_k^- \tag{6.5}$$

At the first iteration $k = 1$, the values of xe_{k-1} and P_{k-1} were initialized previously. At state k, the a priori estimate state variable xe_k^- is calculated from the transition matrices A and B, the previous estimate state variable xe_{k-1} and previous control variable

u_{k-1}. Then, the a priori estimate error P_k^- is predicted from the previous estimate error P_{k-1} and process noise covariance matrix Q. After that the update process is continually performed by calculating Kalman gain K_k from P_k^-, the output matrix H and measurement noise covariance matrix R. The Kalman gain is immediately used again for the calculation of the estimate state variable at the current iteration, xe_k, which is based on the measurement value z_k, and the predicted state variable xe_k^-. Last, the estimate error P_k is updated and the process is repeated again until the last specified number of iteration is reached.

In order that the PMD image can be filtered, the picture is taken when the camera position is static. Since there is a lot of noise due to the camera's movement in this 16x16 pixel camera, the camera has to be at a stand still during the filtering process. The filtering is done at each individual pixel over the sampling period (frames) as shown in Figure 6.8. The sample data of each pixel and each frame are gathered as state variables. Regarding that the camera is standstill, the transition matrix A is a scalar number equal to 1 and matrix B is zero. This transition matrix A represents no movement of the camera. If filtering were to be done while the camera was moving, the matrices A and B would have to be replaced by the proper movement model. As the following, Q and R are also scalar number.

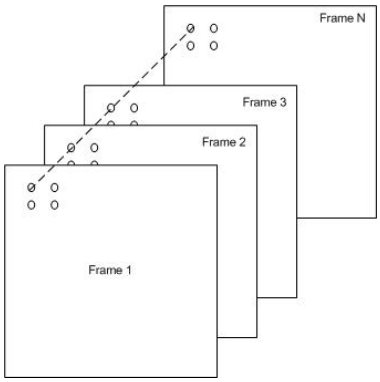
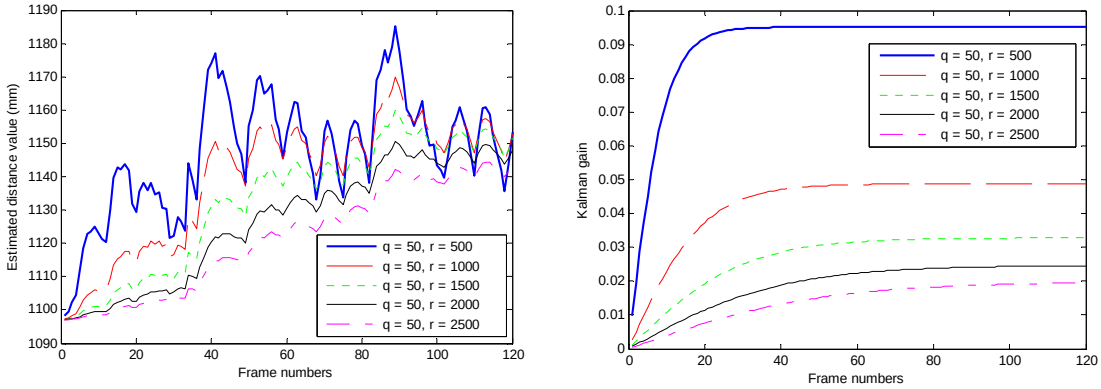


Figure 6.8: Image frames consisting of state variables.

6.3.1.2 Uncertainty and convergence. The process noise covariance Q in (6.2) and the measurement noise covariance R in (6.3) need to be adjusted for obtaining the proper filter since these parameters represent the uncertainty of the process and measurement, respectively. Furthermore, the convergence rate of the Kalman gain is also dependent on these parameters. With the proper adjustment, the convergence rate and the oscillation of the estimated value xe_k at a steady state can be traded off.

According to section 6.1, the outer pixel shows larger noise than the middle pixel, the measurement noise covariance is therefore tuned by using the measured data from pixel (16, 16). The value of q is 50 and r is varied from 500 to 2500. As shown in Figures 6.9a and 6.9b, the larger r value results in a smoother estimated value and slow convergence of Kalman gain. The trade off between the oscillation amplitude of the estimated distance value and convergence time of Kalman gain results in selection of q = 50 and r = 1000. Note that $Q = q^2$ and $R = r^2$. The convergence time can be shortened by further tuning of q. The selected value of q is 100 since the deviation lies within 2 cm and the convergence requires at least 30 frames. Figure 6.10 shows the measured value and estimated value after filtering. Note that the value xe_k is initialized by using the average value of the first 10 frames.



(a) (b)

Figure 6.9: The estimated value Kalman gain: (a) The estimated value; q = 50 and various values of r; (b) The Kalman gain; q = 50 and various values of r.

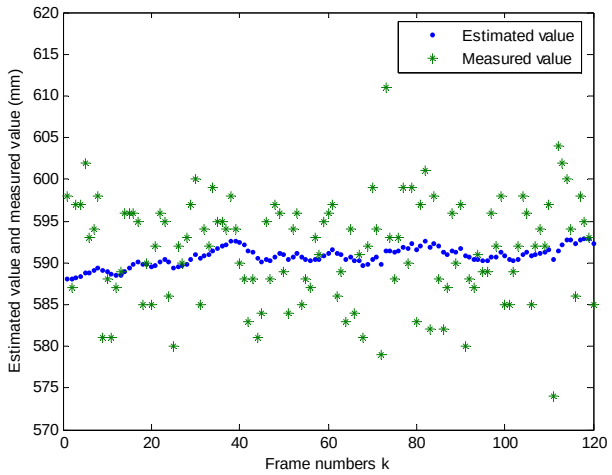


Figure 6.10: The estimated value x_k from filtering and the measured value z_k .

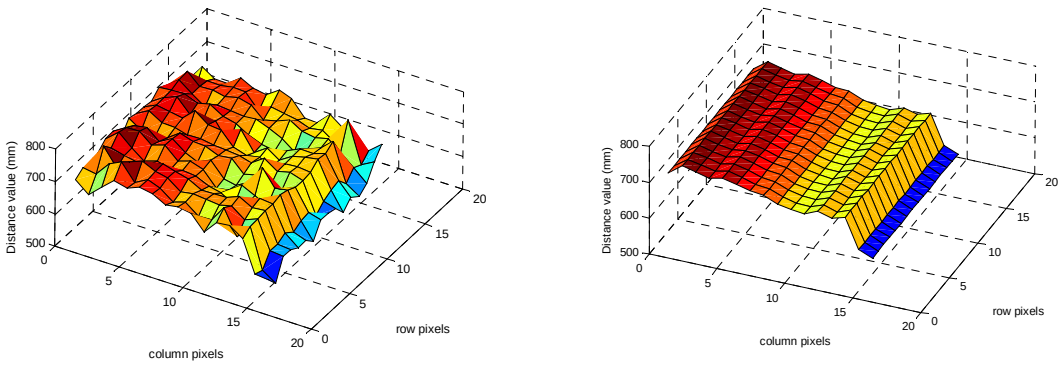
6.3.2 Image smoothing

With regard to the flat surface of rectangular box (the lower part of the landmark), the PMD captured image of this shape should have the same measured value in each column pixel. In practice, there is deviation among pixels in a frame as mentioned in section 6.1. Therefore, the image smoothing gives the image a representative distance value of each column pixel for further use in line fitting and lower part recognition.

The filtered image of 256 pixels is shown in Figure 6.11a. The image is smoothened by using the average of the column neighbourhood pixel value within the same row:

$$x_{j,rep} = \frac{\sum_{i=1}^{n} x_{i,j}}{n} \tag{6.6}$$

where $x_{j,rep}$, is the representative distance value at column j, $x_{i,j}$ is the distance value at the row i and the column j and n is the number of rows that contains the lower part of the landmark. Smoothing is not performed in the upper part of the image (representing the upper part of the landmark) to protect edge destruction. The smoothed image is shown in Figure 6.11b.

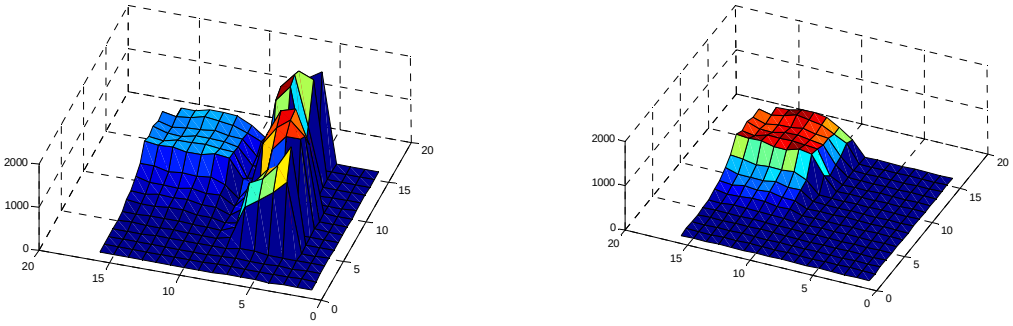


(a) (b)

Figure 6.11: The filtered image before and after smoothing: (a) before; (b) after.

Another function of image smoothing is to discard the unexpected background noises as shown in Figure 6.12a. The noise caused by the background light has very large distance value. Noise can be caused by objects whose surfaces are not flat or by objects with a small surface area since the light scatters away. These measured distance values are undesirable and hinder landmark recognition. Therefore, they are eliminated

out by using the detection of discontinuity of the distance value among neighbour pixel. The pre-defined background value replaces the measured value at the pixel that contains large measured value. After smoothing, the image is as shown in Figure 6.12b.



(a) (b)

Figure 6.12: The PMD image smoothing of random measurement noise: (a) before; (b) after.

6.3.3 Model image generation

Due to the differences in the design of the upper and lower parts of the landmark as described in Section 6.2, the model image is also generated separately for the different parts. The image of the lower part is generated first, followed by that of the upper part. The following sections explain the methodologies used for generation of the model images and show the sample model images.

6.3.3.1 Model image generation for the lower part of the landmark. The model image is generated by simplifying the 3D space into 2D space. The landmark and camera positions are first defined on the xyz coordinate. As shown in Figure 6.13, the coordinates of the camera and the landmark are defined as $P(x_c, y_c, z_c)$ and $P(x', y', z')$, respectively. To generate the model, the 3D space is simplified into 2D space by considering the top view of camera vision and landmark.

In Figure 6.13, the z-axis of both the camera and landmark coordinates is identical. It must be assumed that the translation in z-direction is neglected. These coordinates are transformed onto the $P(x_c, y_c, z_c)$ coordinate by rotation with angle δ, and translation with distance d. The sideward translation distance d_s is

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} \cos\delta & \sin\delta & 0 \\ -\sin\delta & \cos\delta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} + \begin{bmatrix} d \\ d_s \\ 0 \end{bmatrix}. \tag{6.7}$$

When considering the object corners 1 to 4 on the camera coordinate the landmark consists of 4 boundary equations:

$$x = \frac{w}{2};\ \frac{-l}{2} \le y \le \frac{l}{2}, \tag{6.8}$$

$$x = \frac{-w}{2};\ \frac{-l}{2} \le y \le \frac{l}{2}, \tag{6.9}$$

$$y = \frac{l}{2}; \frac{-w}{2} \le x \le \frac{w}{2}, \tag{6.10}$$

$$y = \frac{-l}{2}; \frac{-w}{2} \le x \le \frac{w}{2}. \tag{6.11}$$

Four boundary equations are constructed from the relationship of coordinates at each corner as

$$a_{o,i} = \frac{(y_i - y_{i+1})}{(x_i - x_{i+1})}, \tag{6.12}$$

$$b_{o,i} = y_i - a_{o,i} x_i, \tag{6.13}$$

where $a_{o,i}$, and $b_{o,i}$ are linear equation coefficients of the landmark of the boundary i. The boundary equations are shown again in Figure 6.14. This figure shows how we calculate the distance value for each column pixel of the model image. The camera vision consists of 16 column pixel linear equations. At each column pixel p, the equation is

$$y = a_{c,pc} x + b_{c,pc}, \tag{6.14}$$

$$a_{c,pc} x = \tan(p^c . \alpha_{pc}), \tag{6.15}$$

where $a_{c,pc}$ and $b_{c,pc}$ are linear equation coefficients of the camera at the column pixel pc. The integer value of p^c is the defined pixel position from -8 to 8. The angle between two column pixels is α_{pc}.

This figure also shows the intersection points of the camera equation and the object equation. There are 5 pixels in the middle that have the distance value of the landmark. The other pixels have the background value. The generated model image is

an image of distance position obtained from the intersection of an object and camera equations. By rearranging the equation from (6.13) we have,

$$x = \frac{y - b_{o,i}}{a_{o,i}}. \tag{6.16}$$

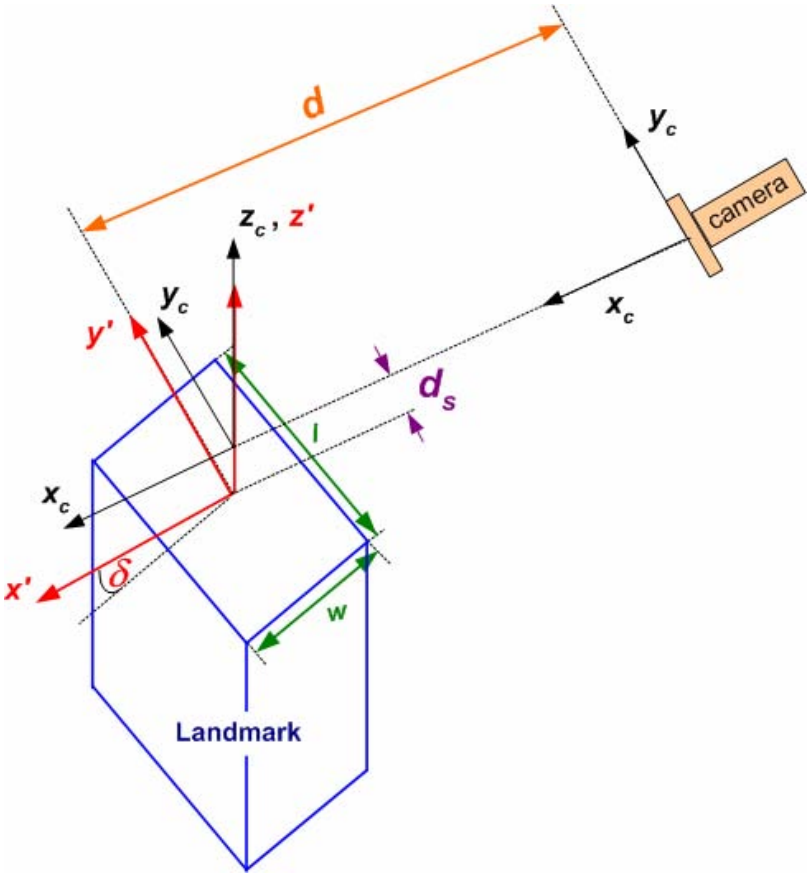


Figure 6.13: Camera and landmark coordinates.

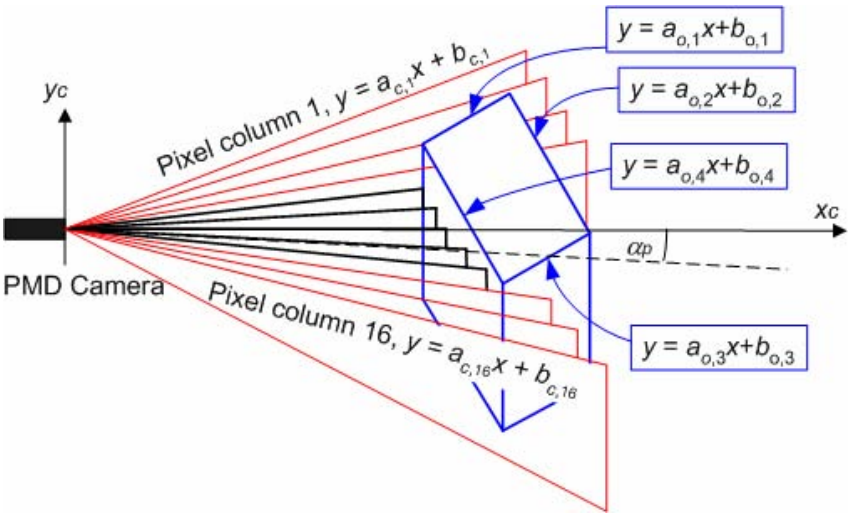


Figure 6.14: Model image generation of the lower part.

The distance value y at the intersection of the camera and the landmark equations is obtained by replacing (6.16) in (6.14),

$$y = \frac{\left(a_{o,i} b_{c,pc} - a_{c,pc} b_{o,i}\right)}{\left(a_{o,i} - a_{c,pc}\right)}. \tag{6.17}$$

In Figure 6.13, when the angle position δ is varied, the upper part remains at the same pixel position since the upper part of image is view invariance as mentioned earlier in the landmark designs. As shown in Figures 6.15a and 6.15b, the upper part remains the same, whereas the lower part is rotated, the edge is shifted and the slope presents clearly.

6.3.3.2 Model image generation for the upper part of the landmark. Sequentially, the upper part of the model image is generated after the image of the lower part has successfully been generated. Since the generated model image is the image of the distance position obtained from the intersection of landmark's boundary equations and camera equations. The intersection points are calculated from the known value of distance d as shown in Figure 6.16. The intersection point is found by replacing d as x in the row pixel equation of the camera. The general camera equation at each row pixel pr is

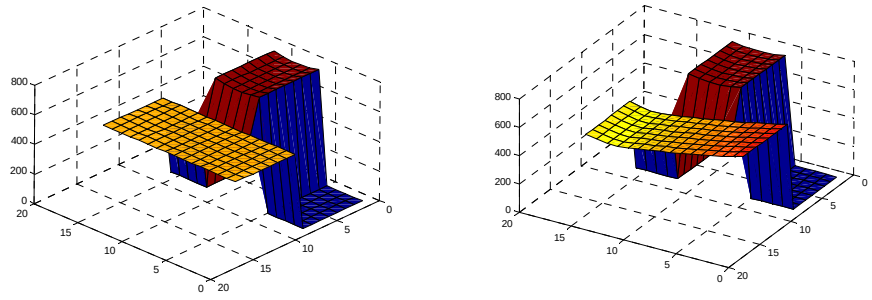
$$z = a_{r,pr}x + b_{r,pr} \tag{6.18}$$

$$a_{r,pr}x = \tan(p^r.\gamma_{pr}). \tag{6.19}$$

where $a_{o,i}$, and $b_{o,i}$ are linear equation coefficients of the landmark boundary i. $a_{r,pr}$ and $b_{r,pr}$ are linear equation coefficients of the camera at pixel p^r. The camera equations are generated by using the angle γ_{pr}, which is obtained from the angle between two row pixels. Note that d is inherited from the model generation of lower part.

The generated model images of the rectangular type landmark, null type, and cylinder type are shown in Figure 6.17. Since the upper part of the rectangular type has the same shape as the lower part, the model image is a plane as shown in Figure 6.17a. For the null type, the upper part is the background as shown in Figure 6.17b. In fact, the rectangular type and the null type have the same shape, just with a different height measurement for the lower part. The height of the null type landmark is less than that of

the rectangular type. By using (6.18) and (6.19), we generate the rectangular and null type landmark. Next, the model image generation of the upper part of the cylinder type as shown in Figure 6.17c can be explained.



(a) (b)

Figure 6.15: The model image at different angle positions: (a) 0°; (b) 45°.

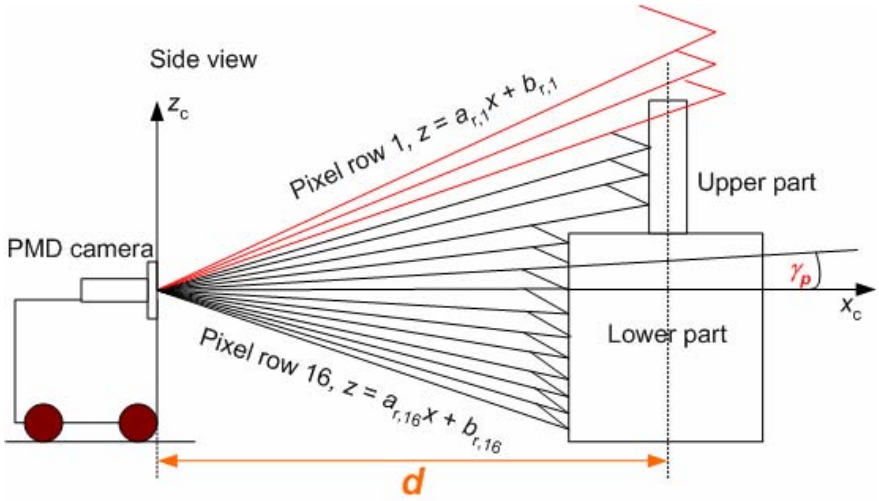
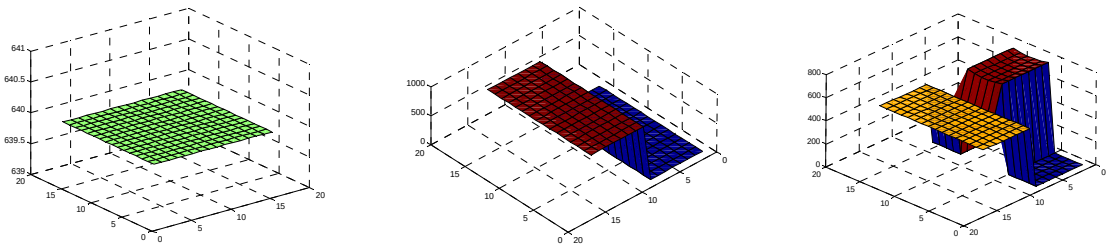


Figure 6.16: Model image generation of the upper part.



(a) (b) (c)

Figure 6.17: Generated model of landmark types: (a) Rectangular; (b) Null; (c) Cylinder.

Figure 6.18 shows the top view of the column pixel positions and the cylinder. Following the same approach as for the model image generation of the lower part, the intersection points between the 16 column pixel equations form a circle equation to give the distance value. The translation d and d_s is applied, whereas rotation is not present, $\delta = 0$. In this case, rotation is not considered, regarding view invariance as mentioned earlier in Section 6.2. In Figure 6.18, the circular equation represents the top view of the landmark:

$$(x\text{-}d)^2 + (y\text{-}d_s)^2 = r^2. \tag{6.20}$$

The distance value is equal to the value of x at the intersection points between the camera column pixel equation (6.14) and the circular equation (6.20). Replacing (6.14) in (6.20) results in:

$$\left(1 + a_{c,p}^2\right)x^2 - 2\left(d - a_{c,p}b_{c,p} + a_{c,p}d_s\right)x + d^2 + b_{c,p}^2 + d_s^2 - 2b_{c,p}d_s - r^2 = 0 \tag{6.21}$$

By solving this equation, we get the distance value of each column pixel of the upper part.

6.3.4 Edge detection

The edge detection of PMD images is detecting a sudden change in distance value among column pixel neighbourhoods. The difference of distance value in columns *diff_c* at the column i and in rows *diff_r* at the row j are calculated by

$$diff\_c(i) = x_e(i) - x_e\,(i+1), \tag{6.22}$$

$$diff\_r(j) = x_e\,(j) - x_e\,(j+1), \tag{6.23}$$

where $x_e(i)$ and $x_e(i+1)$ are the distance values at the i^{th} column and $(i+1)^{th}$ column. $x_e(j)$ and $x_e(j+1)$ are the distance values at the j^{th} column and $(j+1)^{th}$ row.

Table 6.3 shows the detected edges of a model image and a smoothed image. The model image has two clear edges but the smoothed image has sub edges. The edge detection is exploited later to classify the landmark according to type. For landmark recognition in Section 6.3.6.2, the column edge width W_c is defined as the width of the image in columns and the row edge width W_r is defined as the width of the edge width

in rows. For example, the model image has a rising edge at the 5th column and a falling edge at the 10th column. The column edge width is 5 (W_c = 5).

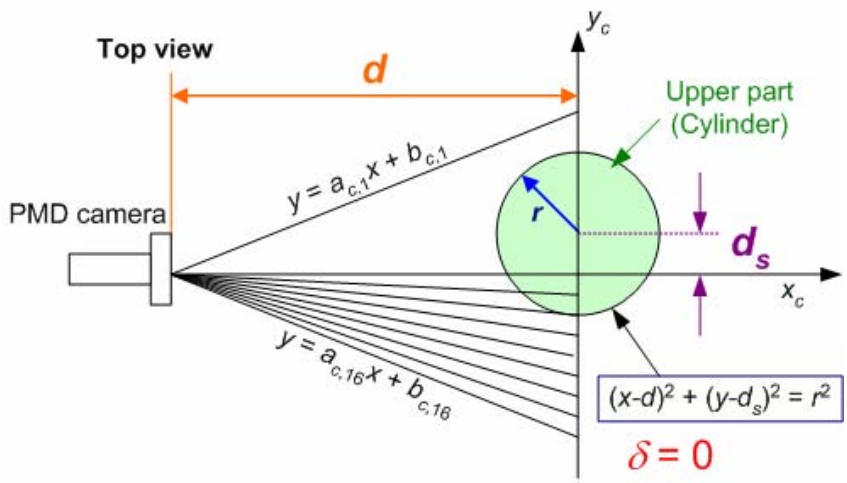
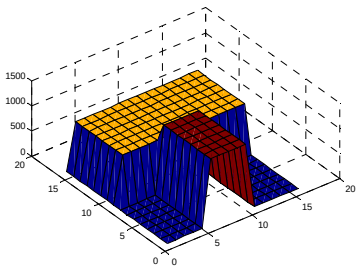
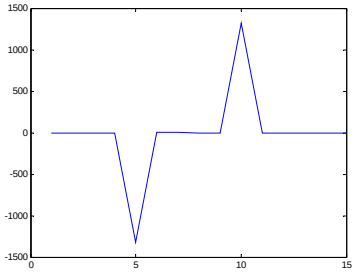
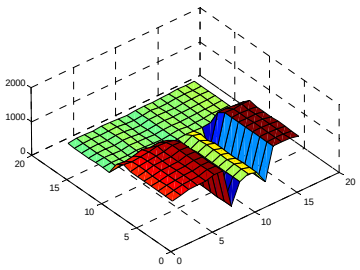
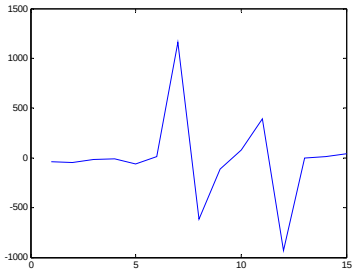


Figure 6.18: Model image generation of a cylinder.

Table 6.3: Images and edge detection in columns.

Type	Images	Edges (*diff_c* versus column pixels)
Model image		
Smoothed image		

6.3.5 Line fitting

Line fitting is exploited to determine the slope of the lower part as shown in Figure 6.4. The slopes of the lower part of the model images and smoothed images are both calculated by using linear regression. When a representative distance value from each of the column's pixels is taken, there is a result of 16 distance values for each image. These representative values can be taken from any row in the lower part since

the image was smoothed and thus has the same distance value for all rows in the same column of the lower part. These values are arranged into the vector matrix for solving the slope and constant of the linear equation. Let A_s and B_s be matrices that contain the column numbers $x_{s,1}$, $x_{s,2}$, …, $x_{s,max}$ and the distance values $y_{s,1}$, $y_{s,2}$, …, $y_{s,max}$, respectively.

$$A_s = \begin{bmatrix} 1 & x_{s,1} \\ 1 & x_{s,2} \\ \vdots & \vdots \\ 1 & x_{s,\max} \end{bmatrix}, B_s = \begin{bmatrix} y_{s,1} \\ y_{s,2} \\ \vdots \\ y_{s,\max} \end{bmatrix} \tag{6.24}$$

$$\begin{bmatrix} c_s \\ m_s \end{bmatrix} = ((A_s^T A_s)^{-1} A_s^T) B_s \tag{6.25}$$

where c_s is the constant value of the linear equation and m_s is the slope value. A sample graph of the result from line fitting, $y_{sf} = m_s x + c_s$, and the raw data y_s are shown in Figure 6.19.

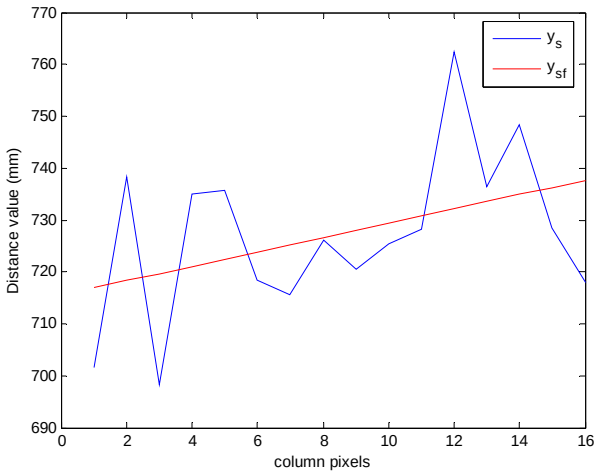


Figure 6.19: A sample graph of line fitting result.

6.3.6 Model matching

Model matching is processed in concatenation with the image smoothing, model image generation, edge detection and line fitting. Model matching consists of two sub-processes, matching the upper part and the lower part.

6.3.6.1 Model matching for the lower part of the landmark image. To match the lower part of the landmark image, the matched distance position *D* and the matched

angle position P have to be found. The matching process is as follows: The model images are generated by varying D and P from the minimum to the maximum value in the iterative loop. For each model image, the slope value and distance value of the lower part needs to be calculated and compared to those of the smoothed image. The model that has the smallest slope error and distance error among the other model images can best be fitted to the smoothed image and D and P of that model image are taken as the result of the matching process.

As already explained in section 6.2.1, the slope value and the distance value are criterions for recognizing the distance position and angle position. These criterions are applied as the slope error and the distance error. The slope error ES (D, P) is the square of the different value between the slope value of the smoothed image xs and the slope value of the model image at distance position D and at angle position P, ms (D, P). The slope error is calculated by using

$$ES(D,P) = (xs - ms(D,P))^2 . \tag{6.26}$$

The xs and ms(D, P) are obtained by using (6.24) and (6.25). The distance error ED (D, P) is the summation of the distance error from 16 column pixels. This error is calculated by

$$ED(D,P) = \sum_{j=1}^{16} (xd_{i,j} - md(D,P)_{i,j})^2 , \tag{6.27}$$

where $xd_{i,j}$ is the distance value taken from the lower pixel image of the smoothed image at row pixel i and at column pixel j. $md(D, P)_{i,j}$ is the generated distance value from the model image at row pixel i and at column pixel j of the distance position D and angle position P. As shown in Figure 6.20, the process starts obtaining xs and calculating ms (D, P) for each position D and P. The minimum error $minED$ is initialized as a large value and D is set to the first step that is 1. The iterative loop starts at D = 1 and stops at $D = D_{max}$. At each step D, the error $ED(D)$ is calculated by using

$$ED(D) = \sum_{P=1}^{max\_P} ED(D,P) , \tag{6.28}$$

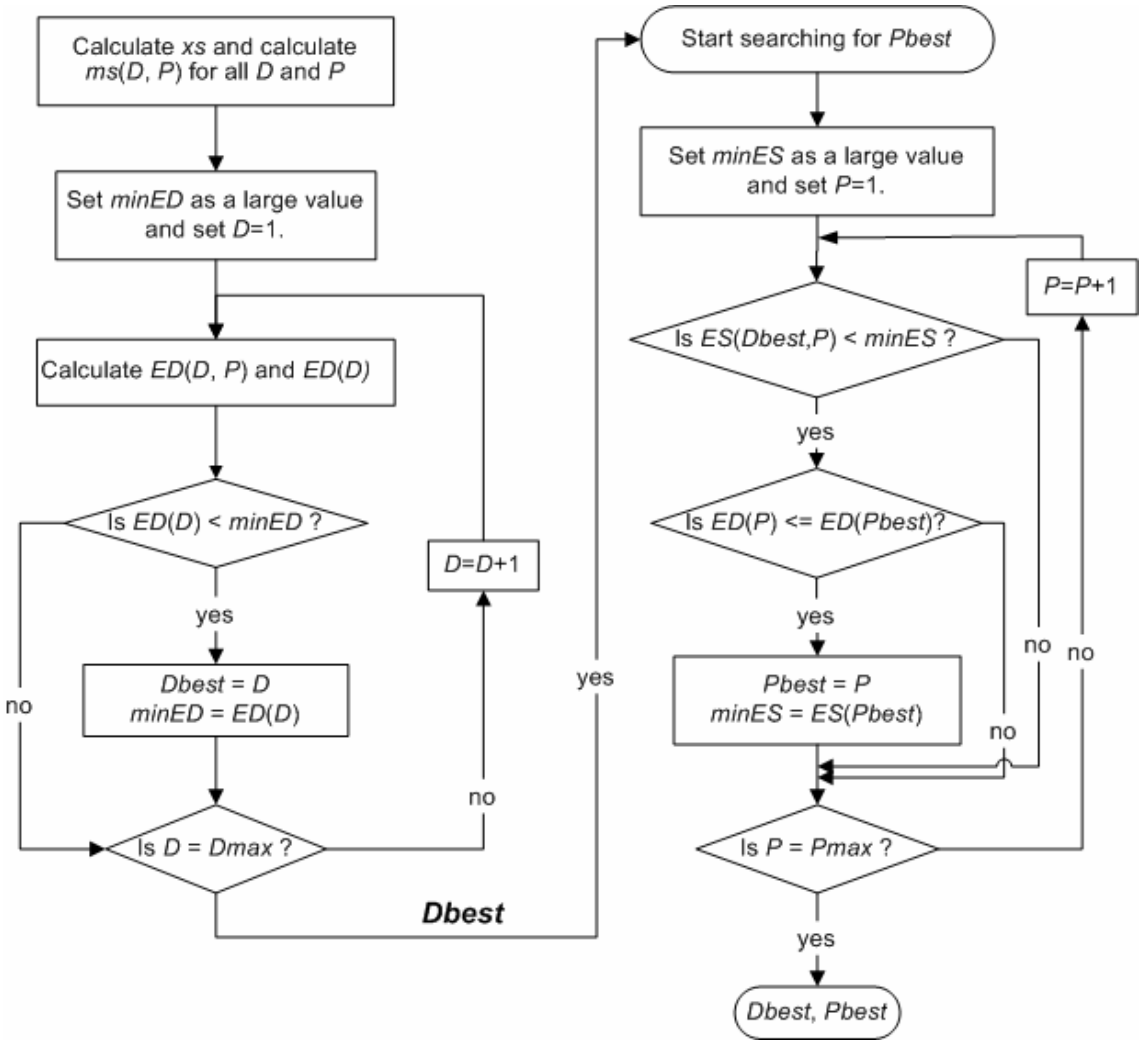


Figure 6.20: Matching process of the lower part.

where max_*P* is the maximum angle position. When *ED*(*D*) is smaller than the minimum error *minED*, *minED* is updated and at the last step D_{max}, the best distance step D_{best} is further used for searching for the best angle position P_{best}.

The right hand side of the flowchart shows the process by which the best fit angle position can be found. It is started by the initialization of the *minES* to a large value and then setting *P* = 1. The angle position *P* is varied until the maximum position P_{max}. If the error $ES(D_{best},P)$ is less than *minES* and *ED*(*P*) is less than or equal to $ED(P_{best})$, the new solution P_{best} is updated. Here, the first decision-making step is checking whether the error $ES(D_{best},P)$ is less than *minES* and the second decision-making step is checking whether *ED*(*P*) is less than or equal $ED(P_{best})$. The second decision-making step should also involve because the slope error is sometimes

insufficient as mentioned in section 6.2.1, the slopes of the images of 0° and 90° are the same. Finally, at P_{max}, the outputs of the process D_{best} and P_{best} are obtained.

6.3.6.2 Classification of the landmark types using the upper pixel processing (Model matching for the upper part). In this process, upper pixel processing is used for recognizing the type of landmark. The landmark was designed as described in Section 6.2, such that it is divided into four different types of landmark, L1- L4.

The process to distinguish the landmark types by using tree search and edge detection is as is shown in the diagram in Figure 6.21. It must be assumed that the landmarks L1 - L4 are belong to class A. In class A, there are three groups of landmark types: rectangular, null shape, and cylinder. The landmark types are recognizable by using the edge detection described above in Section 6.4.3.

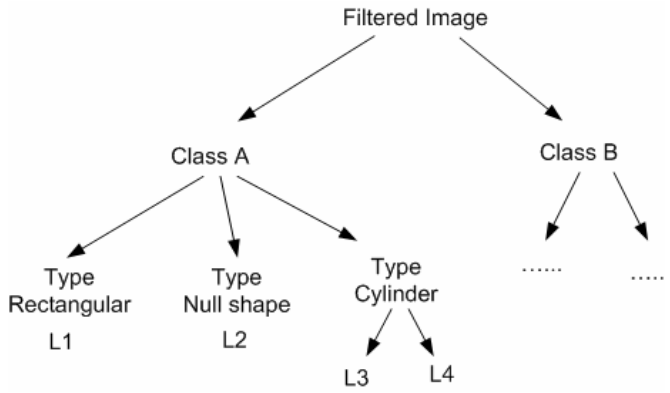


Figure 6.21: The tree diagram for landmark type classification.

The criteria for distinguishing between landmark types are the edge widths that are the column edge width W_c and the row edge width W_r of the smoothed image and of the generated model image. The characteristics of the edges of four landmark types L1 - L4 have to be investigated first (below) in order to define the distinguishing classifiers. The criteria for determining a classifier for landmark type are:

- If there is no column edge present (W_c = 0), check the row edges in order to distinguish between L1 and L2. If there is no row edge (W_r = 0), the match is L1. Otherwise, the match is L2.
- If there are column edges ($W_c \neq 0$) present but no row edge (W_r = 0), check for the column edge widths for L3 and L4. By defining the two errors as $ErrorW_{c3}$

and $ErrorW_{c4}$, the minimum error for each type is matched to L3 and L4, respectively. These errors are calculated by

$$ErrorW_{c3} = W_c - W_{c3,model}, \quad (6.29)$$

$$ErrorW_{c4} = W_c - W_{c4,model}, \quad (6.30)$$

where $W_{c3,model}$ and $W_{c4,model}$ are the column edge widths of the model images of L3 and L4, respectively. Note that when the resolution of PMD camera is higher, landmarks can have more complicated shapes and the object recognition of range images [BES 88] [JAI 90] [FAN 90] [SUK 92] can be exploited for upper part recognition.

6.3.7 Tests of landmark recognition

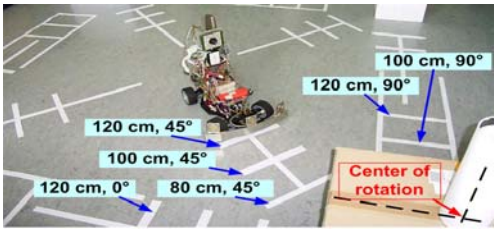
Since landmark recognition is divided into two parts, recognition of the upper part and the lower part, the experiments are set up to test the two parts of landmark recognition separately by using off-line image data. In section 6.3.7.1, the first test is to check the performance of the lower part recognition strategy that was explained earlier in Section 6.3.6.1. A landmark was placed in front of the robot and in varying positions. In section 6.3.7.2, the second test is to check the performance of the upper part recognition strategy. The images of four different landmark types were tested using the method described above in Section 6.3.6.2.

6.3.7.1 A test of the lower part recognition strategy. The landmark recognition strategy was tested off-line with the 16x16 pixel camera. In practice, we want to use the result from landmark recognition for the position calibration as explained later in Section 6.4. Here, the actual positions of robot for the position calibration are marked on the ground as shown in Figure 6.22a. The centre of rotation represents the landmark's centre of gravity.

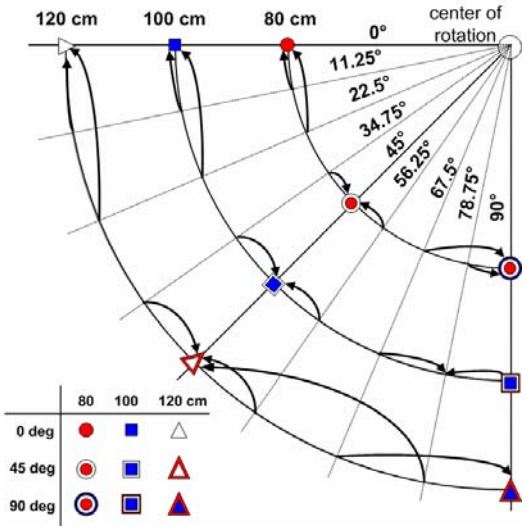
In fact, we do the experiment in the inverse way. We put the landmark on a rotary stage and fix the camera in order that we can set the distance positions and angle positions as we want exactly. Then, we start the first experiment with course angle step of 11.25°. The tested angle positions are at 0° to 90° with 11.25° step and the distance positions are at 80 cm, 100 cm, and 120 cm. In total, we are matching 10 angle positions x 3 distance positions = 30 calibrating positions. Note that since the rotation of

landmark occurs around its centre of gravity, the sideward translation of the coordinate could be neglected.

Figure 6.22b shows symbols of the main distance and angle positions as rectangles, circles, and triangles. The results obtained from the landmark recognition processes can be explained as follows: At a radius of 80 cm, at 0°, the matching result is 0° angle position shown as a circle. At 11.25° and 22.5°, the arrows point from the angle positions 11.25° to 0° position and from 22.5° to 0° position since the matching result of 11.25° and 22.5° angle positions are both 0°. In the same way, the matching result of 34.75°, 45°, and 56.25° angle positions are 45°. Also, the matching results of 67.5°, 78.75°, and 90° angle positions are 90°. For 100 cm and 120 cm, the rectangles and triangles represent the matching results and the results have similar matching errors. These errors are from the undistinguishable of neighbourhood positions regarding the resolution of image and uncertainty in measurement of the 16x16 pixel PMD camera. However, at the main distance and angle positions marked as symbols, we get correct matching results.



(a)



(b)

Figure 6.22: Matching results of the lower part using 16x16 pixel PMD camera: (a) actual positions; (b) symbolic representation of the results.

Up to this point, we conclude that the recognition of smoothed images are qualified at 0°, 45°, and 90° angle positions of 80 cm, 100 cm, and 120 cm distance positions. For other angle positions, the matching results are not satisfied.

Next, we do a new test by using courser angle step of 45°, instead of 11.25°. We do matching for only 9 positions, Pos1 to Pos9 defined in Table 6.4. Since these positions are the positions where robot perform the landmark recognition and the position calibration, they are called calibrating positions. We investigate further for the numerical results of the matching process at these calibrating positions. The process is as explained above. Matching according distance position was performed first, in keeping with the matching process described above in Section 6.3.6.1. The summations of distance errors $ED(D)$ in (6.28) are shown in Table 6.5a. Let the wildcard '*', '$^{\#}$', and '^{x}' represents the candidate distance positions 80cm, 100cm, and 120cm, respectively. As shown in the table, at 80 cm distance position, the summation of error ED (*), is smaller than ED ($^{\#}$) and ED (^{x}). Therefore, the matched distance position D_{best} of Pos1 is 80 cm. For Pos2 – Pos9, the bold numbers are the minimum errors. The matching results of all 9 positions are correct. These results D_{best} will be used further for matching the angle positions in Table 6.5b.

The second step in the matching process is to find a match for the angle position P_{best}. Table 6.5b consists of the slope error $ES(D_{best}, P)$ and distance error $ED(D_{best}, P)$ at each candidates for P_{best}; 0°, 45°, and 90°. The matched angle position is the position that has the minimum error for both ES (D_{best}, P) from (6.26) and ED (D_{best}, P) from (6.27).

Table 6.4 Names of calibrating positions.

Positions	**80 cm**	**100 cm**	**120 cm**
0°	Pos1	Pos4	Pos7
45°	Pos2	Pos5	Pos8
90°	Pos3	Pos6	Pos9

Table 6.5a: Matching result of the distance position.

Position	**Candidates for D_{best}**			**Matching result: D_{best}**
	ED (*)	**ED ($^{\#}$)**	**ED (^{x})**	
Pos1	**19.95**	105.11	482.84	80cm
Pos2	**30.91**	271.83	807.01	80cm
Pos3	**57.69**	364.81	965.95	80cm
Pos4	287.76	**36.35**	75.29	100cm
Pos5	135.20	**18.82**	194.30	100cm
Pos6	63.80	**39.99**	307.99	100cm
Pos7	889.12	324.27	**47.72**	120cm
Pos8	633.62	182.01	**19.96**	120cm
Pos9	428.64	86.30	**33.31**	120cm

Table 6.5b: Matched result of the angle position.

Position	Candidates for P_{best}						Matching result:
	***ES*(,0°)**	***ED*(,0°)**	***ES*(,45°)**	***ED*(,45°)**	***ES*(,90°)**	***ED*(,90°)**	P_{best}
Pos1	**0.25***	**5.95***	203.04*	49.68*	0.25*	143.84*	0°
Pos2	166.11*	226.87*	**0.75***	**58.07***	166.11*	24.11*	45°
Pos3	0.33*	366.53*	205.16*	185.72*	**0.33***	**24.63***	90°
Pos4	**0.72#**	**2.53#**	375.14#	117.01#	0.72#	244.01#	0°
Pos5	243.75#	117.70#	**8.44#**	**8.11#**	243.75#	62.42#	45°
Pos6	0.55#	253.89#	370.92#	142.59#	**0.55#**	**3.39#**	90°
Pos7	**0.72x**	**11.15x**	493.99x	159.87x	0.72x	306.17x	0°
Pos8	294.91x	75.88x	**17.68x**	**3.03x**	294.91x	120.71x	45°
Pos9	6.83x	187.78x	575.57x	143.03x	**6.83x**	**2.30x**	90°

As shown in Table 6.5b, at Pos1, when comparing *ES*(*, 0°) with *ES* (*, 90°), they have equal slope errors of 0.25 but their distance errors are different. Since, the distance error *ED* (*, 90°) is larger than *ED* (*, 0°), the matching result P_{best} of Pos1 is 0°. This double decision-making step was mentioned in section 6.3.6.1. For Pos3, Pos4, Pos5, Pos7, and Pos9, the distance errors at 0° and at 90° are compared since the slope errors are equal.

All matching results are correct. Therefore, we call the resolution of position calibration is of 45° step and 9 calibrating positions. Next, we proceed with the experimental results of matching of the upper part.

6.3.7.2 A test of the upper part recognition strategy. This test makes use of the classification categories for the landmark types described in Section 6.3.6.2. Eight filtered images, two images of each of landmark L1 – L4, are taken for test purposes. These images in Figure 6.23-6.26 are named I1 - I8, respectively.

As shown in Figure 6.23a and 6.23b, the filtered image of L1 is a flat plane across all pixel rows, since L1 is a rectangular type landmark. Figure 6.24a and 6.24b shows landmark L2, which has null shape. The distance values present in the upper part of the image reflect the background distance. Figure 6.25a is a filtered image of L3, which is a cylinder type landmark. The upper part image shows the convex of the cylinder, instead of the concave as in the model image since the light scattering at the edge of the cylinder causes small reflected light and therefore smaller distance value than the actual distance. Figure 6.25b is also an image of L3, but the cylinder part appears smaller since the image was taken from farther away. Figure 6.26a depicts L4, also a cylinder type landmark in a different size. The image shows the upper part of L4 as a small cylinder tab which is farther away than the lower part of the landmark.

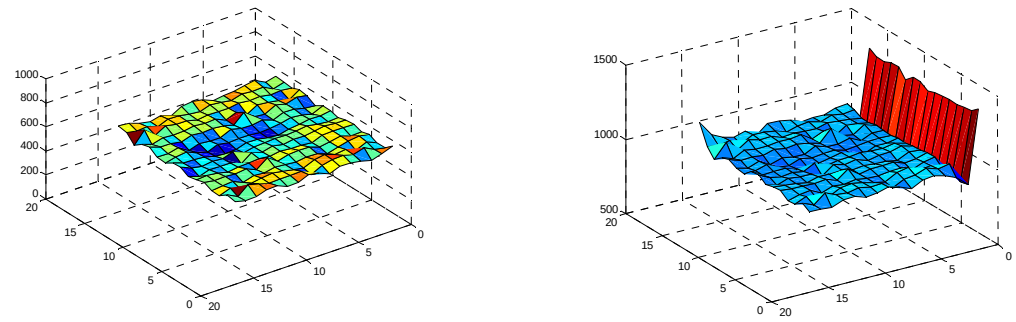
Similarly, in Figure 6.26b, the upper part is almost not detectable since the cylinder is even farther away from the camera. The light is scattered away from the landmark due to the curved shape of the cylinder's surface.

The matched type results are shown in Table 6.7. In I1 and I2, the column edge width W_c is zero and the row edge width W_r is zero. There are no edges present in the image. From this information, the matching strategy deduces that the landmark type is rectangular that is L1. I3 and I4 show the column edges to be zero and the row edge widths as 8. These are the measurements for the null type landmark and so the matched result is L2. In I5 and I6, the row edge widths are also zero but unlike the two previous examples there is a detectable column edge width. It must then be determined whether or not this edge width matches the measurements of the generated model for L3 or L4. As the diameters of the cylinders are different for the two landmarks, the column edge widths are also different.

The difference in edge widths is the criterion for matching and these differences are called $ErrorW_{c3}$ and $ErrorW_{c4}$ as in (6.29 – 6.30). In I5, $ErrorW_{c3}$ is less than $ErrorW_{c4}$. Therefore, the matched type is L3. The same criterion is applied for I6, I7, and I8. We obtain the correct landmark types for all images. The matching tests are therefore successful. Further, during the real-time matching process, the D_{best}, P_{best} and landmark types are exploited in the position calibration explained in the coming section.

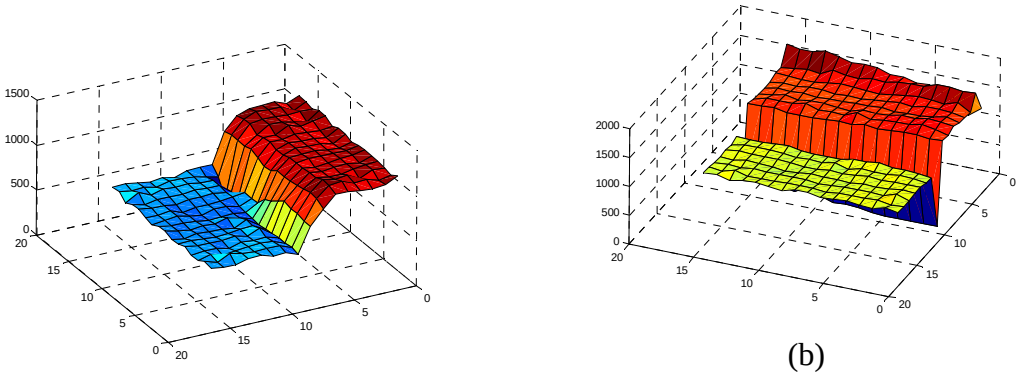
Table 6.6: The matching results of landmark type classifier.

Image name	W_c	W_r	$ErrorW_{c3}$	$ErrorW_{c4}$	Matching result: Landmark type
I 1	0	0	0	0	L1
I 2	0	0	0	0	L1
I 3	0	8	0	0	L2
I 4	0	8	0	0	L2
I 5	8	0	1	3	L3
I 6	5	0	0	2	L3
I 7	4	0	3	1	L4
I 8	3	0	2	0	L4



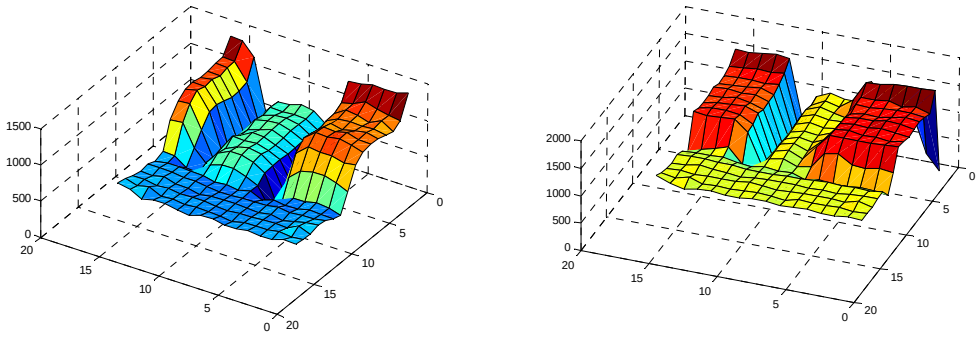
(a) (b)

Figure 6.23: The filtered images of L1: (a) I1; (b) I2



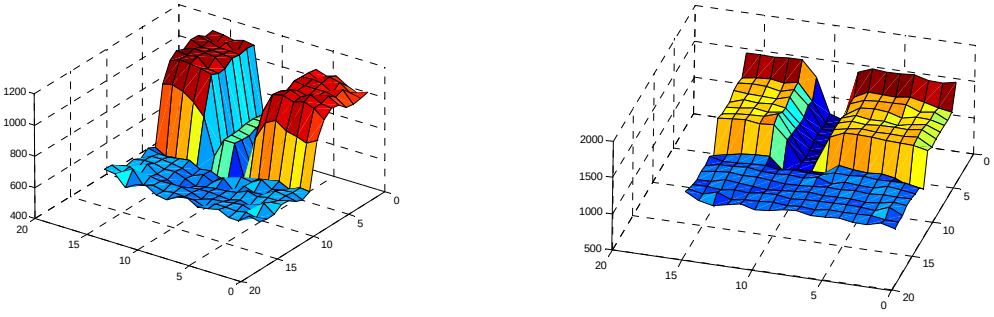
(a) (b)

Figure 6.24: The filtered images of L2: (a) I3; (b) I4



(a) (b)

Figure 6.25: The filtered images of L3: (a) I5; (b) I6



(a) (b)

Figure 6.26: The filtered images of L4: (a) I7; (b) I8

6.4 Position calibration

The idea behind doing position calibration is to reduce the accumulated errors of relative localization that are the position and heading errors. A position calibration is an absolute localization technique. This technique corrects the robot position and heading by using the artificial landmark and 3D vision. In practice, the artificial landmark is placed in the environment before the robot starts its navigation. Since the landmark position is known already and the relative position of the robot to the landmark is obtained from the landmark recognition, the robot position is directly calculable. This section explains the process of calculation of the robot's position and heading using the output of landmark recognition process from the previous section.

6.4.1 Position prediction and update

As shown in Figure 6.27, the vector for robot position V_r is calculated by

$$V_r = V_o + V_d\,, \tag{6.31}$$

where V_o is known from the a priori known landmark position $P(X_o, Y_o)$ and V_d is calculated from the rotational angles δ, the camera heading β, and the radius D. Let us define the related variables here.

In the figure, the global coordinates, camera coordinates, and robot coordinates are denoted by (X_g, Y_g), (X_c, Y_c), and (X_r, Y_r), respectively. δ is the rotational angle between the camera and the object coordinates assumed to be the global coordinates. The camera heading angle β is the angle between the robot and the camera headings. The radius D has centre at the landmark coordinate $P(X_o, Y_o)$. α is the angle of the robot's heading referred to by the global y axis Y_g. β is already known from the orientation and positioning of the camera motor, which is mounted on the robot. If β equals -90°, the camera heading is the same as the robot heading.

With the a priori known object position $P(X_o, Y_o)$ on the global coordinate and the rotational angles δ and radius D obtained from the previous section, the robot position $P(X_r, Y_r)$ using the relationship in equation (6.31), can be calculated by

$$\begin{bmatrix} X_r \\ Y_r \end{bmatrix} = \begin{bmatrix} X_o \\ Y_o \end{bmatrix} + \begin{bmatrix} D\cos(180+\delta) \\ D\sin(180+\delta) \end{bmatrix} \tag{6.32}$$

The robot's heading α, the robot's angle position, and the landmark angle position, to which the global coordinate refers, are obtained from

$$\alpha = \delta - \beta. \tag{6.33}$$

where X_r, Y_r, X_o and Y_o are the position coordinates of the robot and landmark on the global coordinates, respectively. Finally, the position can be updated by replacing the robot position $P(X_r, Y_r)$ with the current estimated robot position on the positioning system.

6.4.2 The on-line position calibration experiment.

The on-line position calibration starts with landmark recognition and outputs from landmark recognition are further exploited as inputs for the position prediction and update processes.

Sample of landmark image taken with the PMD camera are shown in Figure 6.21. The black pixels in the images represent a large distance value (far away). White pixels represent the surface area of the landmark, which is closer than the background wall (black pixels). The blue pixels represent a distance value between white and black. There are some pixels on the edges and corners which are a mixture of white and blue. This shows light reflected at the edges: some of it scatters away and some returns to the camera. Note that a sample L1 image is not shown here since it is an overall white image.

In the tele-operated mode, the robot is controllable by using joystick or arrow keys. In this experiment, we drive the robot to the landmark and stop there. Then, we start image filtering and do the position calibration. Using the PMD camera's GUI, the operator can command the robot to do the position calibration. Appendix B provides details of PMD camera’s GUI. The on-line experimental results are presented here.

The result from different phases of image filtering is shown in Figure 6.29. The top middle panel shows the filtered image and the bottom middle panel shows the captured image from PMD camera. At the beginning, in Figure 6.29a, the top panel is empty. Later, in Figures 6.29b and 6.29c, the filtered images are similar to the captured images. Since the position calibration is integrated into the tele-operated control, after the image filtering, the robot position calibration is performed on-line.

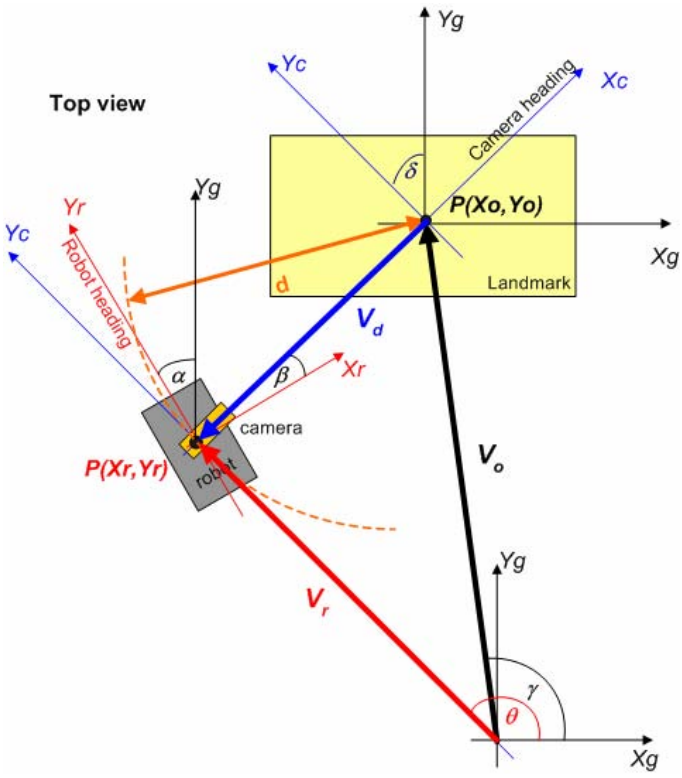
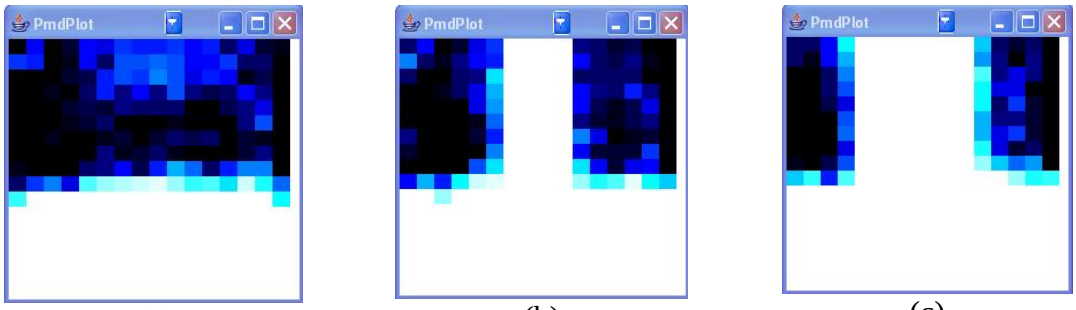
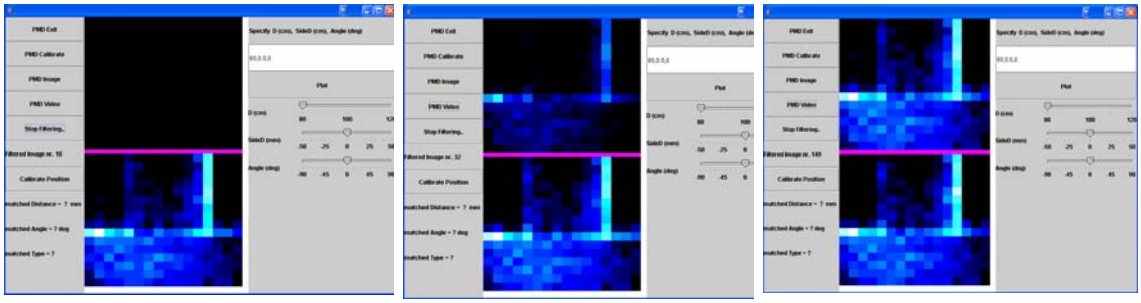


Figure 6.27: The reference coordinates and definition of related variables for position calibration.



(a) (b) (c)

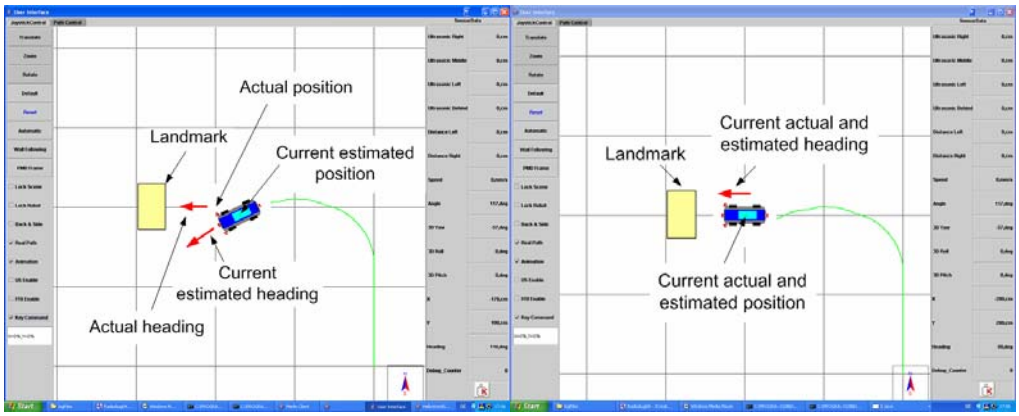
Figure 6.28: Sample PMD images plotted on Java platform: (a) L2; (b) L3; (c) L4.



(a) (b) (c)

Figure 6.29: Filtered images (upper) and captured images (lower): (a) Initial; (b) after some time; (c) finish filtering.

Here, we also present the results of the integration of relative and absolute localization on-line. On GUI, the robot driven path is shown as solid line. This driven path is obtained from the relative localization. Figure 6.30a shows the robot position and heading. Before the position calibration has been done, the robot position has not yet been correctly deduced. After the position calibration, in Figure 6.30b, the estimated robot heading and pose are the same as the actual position. The absolute localization was processed and the robot absolute position and heading replaces the relative position and heading. After the robot calibrates its position, in Figure 6.31, the driven path is shown. When compared to the position and heading before calibration, the error is suddenly eliminated at the calibration point. The position calibration can be activated manually according to the wish of the user.



(a) (b)

Figure 6.30: The robot driven path, position and heading during on-line experiment: (a) before calibrating position; (b) after calibrating position.

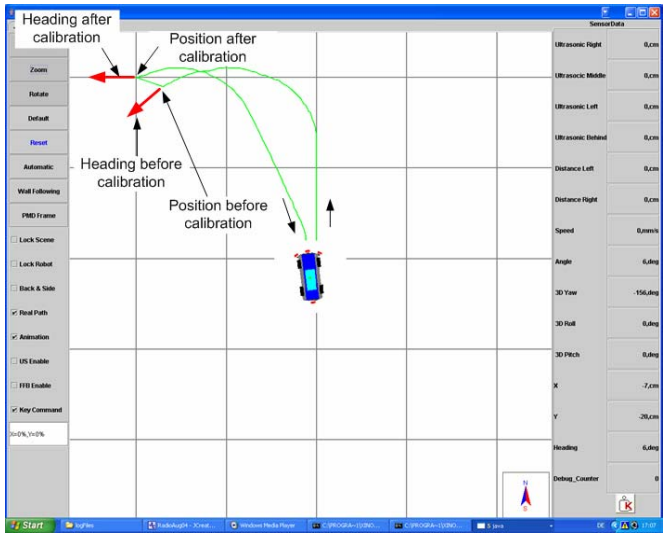


Figure 6.31: The integration of relative and absolute localization on GUI.

At present, the camera is fixed to the robot with no axis to allow for rotation. When the camera was rigged such that it could rotate horizontally, the robot would be more flexible to perform position calibrations since the robot heading would not necessarily have to face the landmark. The robot would only need to stop somewhere near the landmark and the PMD camera could then scan for the landmark in the area around the robot. Further developments should be made so that the robot can distinguish a landmark from other obstacles. If the object found is recognized as a landmark, the robot calibrates its position automatically; otherwise, the robot tries to avoid those objects by getting out of the way.

Besides, the improvement for resolution of the position calibration is necessary since the 16x16 pixel camera provides only 45° angle step and 9 calibrating positions. The robot can only calibrate its position at the calibrating positions as in Table 6.4. The higher resolution for position calibration can be achieved by using the higher resolution camera. In the next chapter, we present the improvement for the resolution of the position calibration.

7 Improvement for the Resolution of the Position Calibration

The previous chapter explained the position calibration by using the 16x16 pixel PMD camera. Since the resolution of the 3D image limits the performance of the landmark recognition, this chapter focuses on the improving the resolution of the position calibration. With the 48x64 pixel PMD camera, we propose a new technique for landmark recognition by using both 3D image and also 2D image. The following sections give a short introduction about the measurement of the 48x64 pixel PMD camera and explain the usage of 2D and 3D images for landmark recognition.

7.1 The measurement characteristics of 48x64 pixel PMD camera.

We start with the investigation of characteristics of measurement data of this PMD camera. We set the integration time to 1000us. Since the deviation of measured value is also depended on the integration time, throughout this chapter we let the integration time be constant. At 2500 mm, the distance value from 100 frames of the centre pixel is shown in Figure 7.1. The largest deviation is 35 mm and the mean value is 2409 mm.

As mentioned in section 6.1, the camera measures the distance to a flat white board but each pixel has different mean values regarding uncertainty of measurement. It is interesting to see how large the different among these mean values is. We take 9 pixels out of 3072 pixels. Figure 7.2 and 7.3 present how the mean and the standard deviation of the distance value are. These pixels are the centre pixel (24, 32), the outer pixels (8, 8), (8, 54), (40, 8), and (40, 54), and the inner pixels (12, 16), (12, 48), (36, 16), and (36, 48).

In Figure 7.2, the mean value at each distances from 500 mm to 2500 mm of all selected pixels are closed to the actual distance. Between 500-1800 mm, the measured values are exact. The mean of the measured distance values farer than 1800mm are shorter than actual distances. At the 2500 mm distance, the different between the measured distance value and the actual distance is in the range of 100 – 130 mm. These mean values present the higher accuracy in measurements, when compared to those of the 16x16 pixel camera.

In Figure 7.3, the STD values of all pixels are proportional to the distances. When the distance is farer, the STD value is also larger. The STD represents the

uncertainty in measurement. At 500 mm, the maximum STD is 2.5mm only. When comparing to the 16x16 pixel camera, this value is much smaller. Moreover, at 2400 mm, the maximum STD is approximately 30 mm whereas that of the 16x16 pixel camera is 170 mm. Therefore, the 48x64 pixel camera has not only higher resolution and faster frame rate than the 16x16 pixel camera, but it also provides smaller uncertainty and higher accuracy in measurement.

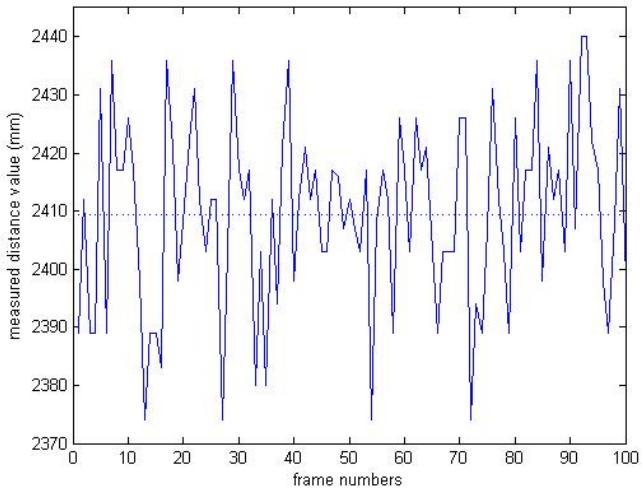


Figure 7.1: Measured distance values of the 48x64 pixel PMD camera at 2500 mm.

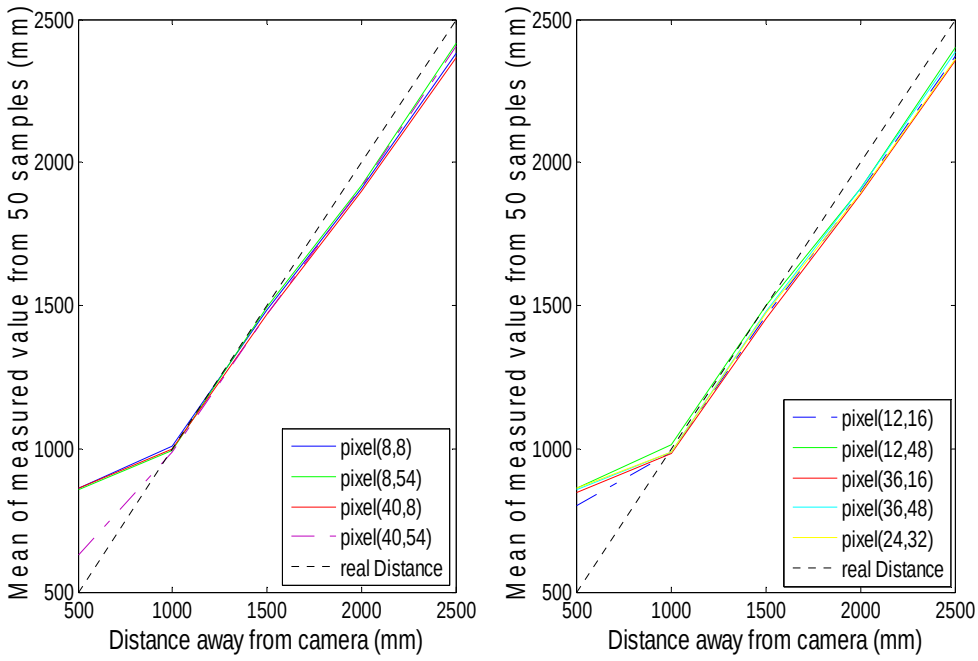


Figure 7.2: Mean of the measured values of the selected pixels.

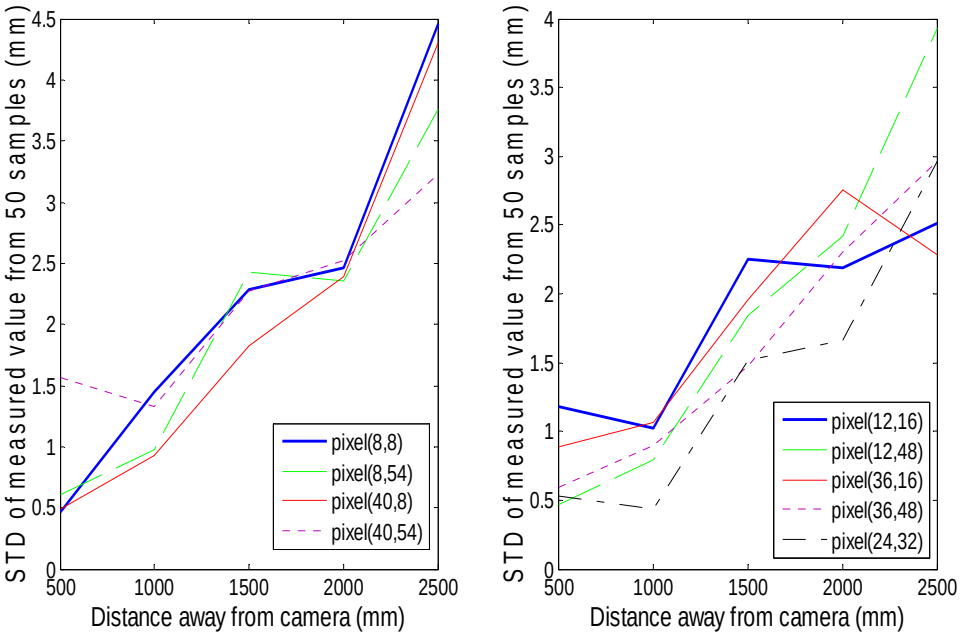


Figure 7.3: Standard deviation (STD) of the measured value of the selected pixels.

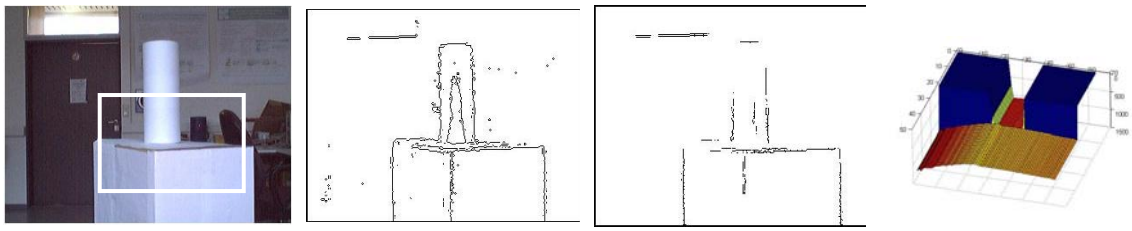
For the landmark recognition, we work in the near distance range of up to 1.5 meter, where the maximum STD is about 7 mm only. Therefore, when the camera is standstill, no filtering is required and only one frame of the scene is sufficient for landmark recognition. We also obtain a profit in time consumption since we do not loose time in filtering process. The landmark recognition process for the 48x64 pixel camera is slightly different from that of the 16x16 pixel camera. The process will be later explained in section 7.3. Here, we proceed with the 2D image processing for landmark recognition.

7.2 The 2D image processing for landmark recognition.

Here, we present the idea of exploitation both 2D and 3D images for improving the resolution of position calibration. We apply the horizontal edge of 2D image in image smoothing of the landmark recognition. Previously, we can do edge detection from the PMD image by using the differential value among neighbour pixel but the detected edges are sometimes not satisfied since it causes error in smoothed image explained later in section 7.3.2. Therefore, the edge from 2D image is exploited as an auxiliary edge for smoothing of PMD image since the 2D image has higher resolution than the PMD image. As to this purpose, what we need are the proper method for 2D edge detection and the relationship of pixel positions of the 2D and 3D images.

7.2.1 The horizontal edge detection of 2D color images.

The Logitech Quickcam® Pro 4000 web camera provides 240x320 pixel resolutions with the captured half angle of 17° vertically and 22.5° horizontally. We use the image processing toolbox in MATLAB to detect edges of the landmark in the 2D image as shown in Figure 7.4a. The image is first converted from RGB into the standard L*a*b* color space created by the international commission of illumination (CIE). After that K-means clustering classifies the color, we perform both horizontal and vertical edge detection as shown in Figure 7.4b. After noise elimination, we get sharp edges as shown Figure 7.4c. Since the vision of the PMD camera is narrower than the web camera, the rectangle frame in Figure 7.4a indicates the detection area of the PMD camera, in which the PMD image is shown in Figure 7.4d. This method of edge detection does not perform well, when the color of landmark is closed to that of the wall in the background, since there is much noise in edge detection.



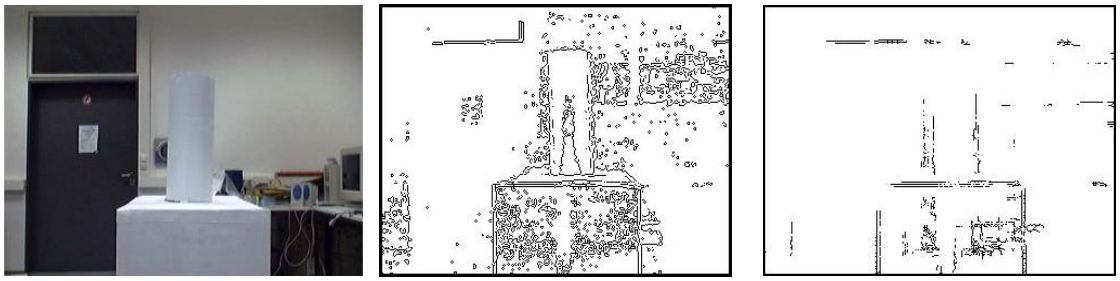
(a) (b) (c) (d)

Figure 7.4: The first result of edge detection of 2D image: (a) original image; (b) edges; (c) edges after noise elimination; (d) 3D PMD image.

Since the contrast of the color is not clearly distinguishable, the white poster on the wall and the door behind the landmark are classified in the same color class as landmark and the detected edges are noisy as shown in Figure 7.5b. When we adjust the threshold value for noise elimination, the edges are as shown in Figure 7.5c. The noises are not easily taken out since when we further remove noises by adjusting the threshold value, the edges are also missing.

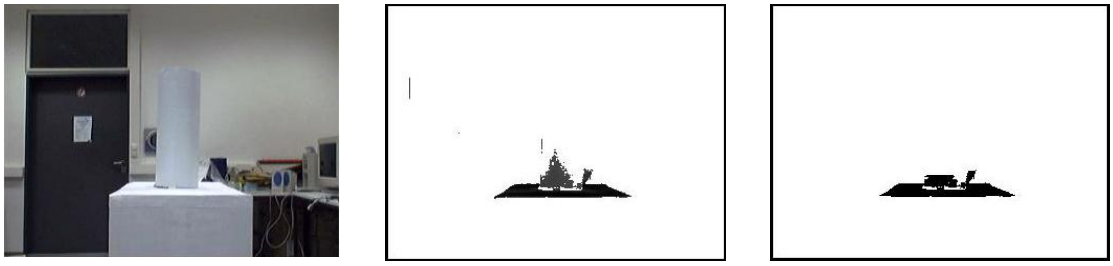
Regarding the above mentioned problem in noise elimination, a new approach is to detect the horizontal edge of the landmark by scanning for the specified RGB threshold value. We select an appropriate threshold value for white color that is obviously distinguishable from the background wall. The result image of Figure 7.6a is shown in Figure 7.6b. Here, the image contains only the landmark. After that the color

image is converted into gray scale image and the noise elimination is performed. The final gray image is shown in Figure 7.6c. The edge detection from this image is qualified since the noises in the background are all eliminated. Next, we take only the horizontal edge that is the border line of lower part of the landmark and place this horizontal edge in the 3D image for the image smoothing described later in section 7.3.3. Here, we explain how the pixel positions of the PMD image are related to those of the 2D image.



(a) (b) (c)

Figure 7.5: The noise in edge detection of 2D image: (a) original 2D image; (b) edges; (c) edges after noise reduction.



(a) (b) (c)

Figure 7.6: The horizontal edge detection of 2D image; (a) original 2D image; (b) gray scale image; (c) the noise eliminated image.

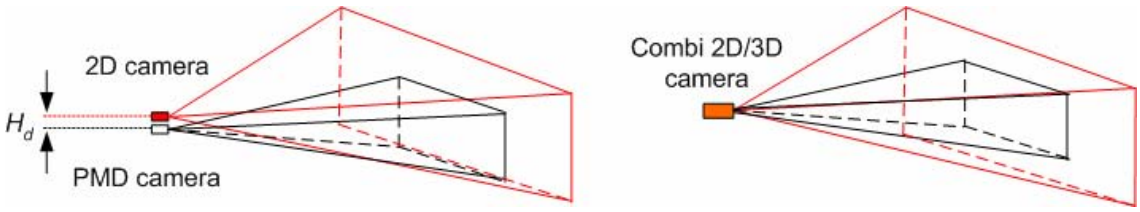
7.2.2 The relationship of pixel positions of 2D and PMD images.

Our aim is to locate the 2D horizontal edges on the 3D image. Therefore, we need to find out the relationship between 2D and 3D row pixels. The relationships of pixel positions depend on camera positions. The first version of camera position is as shown in Figure 7.7a. Though cameras have superposition of the detection area but the altitude level is differed by H_d and the horizontal pixel position are not linearly proportional.

Instead of solving the relationship of pixel position in Figure 7.7a, we found a better solution. An innovative idea is to combine 2D and PMD camera and this new camera is called a Combi 2D/3D camera [PRU 05]. As shown in Figure 7.7b, since the different in altitude is zero (H_d = 0), the 3D detection area lies in the middle of 2D detection area. The pixel positions of the 2D and PMD images are linearly proportional. Since the Combi camera is in production, this work is performed by using the concept of Combi camera. In the experiment, we place the Logitech camera and PMD camera at the same position, one at a time, in order that the optical axis of both cameras is identical. As a result, the relationship between row pixels of the Logitech camera and PMD camera are as following:

$$RP_{pmd} = \left(\frac{RP_{2D} - 120.5}{3.5} \right) + 24.5 \qquad (7.1)$$

where RP_{2D} and RP_{pmd} are the row pixel of the Logitech camera and PMD camera, respectively. The exploitation of this relationship is explained in the coming sections.



(a) (b)

Figure 7.7: The detection area of 2D and PMD cameras: (a) 2D and PMD camera; (b) Combi 2D/3D camera.

7.3 Recognition of the lower part of landmark by using 2D and PMD images.

The recognition of lower part of landmark by using 2D and 3D images is shown in Figure 7.8. First, the 2D captured image is processed for horizontal edge detection as explained earlier in section 7.2.1. This horizontal edge is an input for image smoothing. The 3D captured image is smoothed by using the horizontal edges of 2D image and (7.1). Meanwhile, the model image is generated.

After that the line fitting of both smoothed image and generated model image are processed and are furthered used for matching process. The edge detection and model matching are identical to section 6.3.4 and 6.3.6.1, respectively. Therefore, here, we

explain only the different processes. The model image generation is slightly different from what was explained earlier. For the image smoothing, an additional process is the replacement of horizontal edge of 2D image on the PMD image.

7.3.1 Model image generation

As explain in section 6.3.3, the model image is generated by knowing the half angle of the camera and the landmark dimension. The 48x64 pixel camera has 6.4x4.8 mm^2 chip with 16 mm focus. Therefore, the horizontal and vertical half angle is 11.31° and 8.53°, respectively. The sample of the model images at several angle positions are shown in Figure 7.9. When compared to the generated model image from 16x16 pixel camera, the edge or corner of the lower part of the 48x64 pixel is clearer and the corner of the lower part is easier to recognize. Besides, the surface of image is much smoother regarding less uncertainty in measurement. In Figure 7.9a, the slope value is zero regarding 0° angle position. When the landmark is rotated to 50° angle position, the slope of lower part exists but the edge of corner doesn't exist as shown in Figure 7.9b. When the landmark is further rotated to 60° angle position, a corner exists as shown in Figure 7.9c.

7.3.2 Image smoothing

For this 48x64 PMD camera, the method is identical to as explained in section 6.3.2. The sample captured image before smoothing is shown in Figure 7.10a. In this figure, the background values are those large distance values on the left upper corner and on the right upper corner.

When we detect edges by using the differences of measured distance value among neighbourhood pixels, we get sometimes improper horizontal edges and we obtain the smoothed image as shown in Figure 7.10b, instead of Figure 7.10c. The problem is that the number of row n in equation (6.6) is not proper. Therefore, the average value is wrong since it consists of the large distance values in the upper corners of background values in Figure 7.10a.

A solution for this problem is replacing the detected horizontal edge from 2D image to the horizontal edge of PMD image. After the 2D edge detection as explained in section 7.2.1 and (7.1) are applied. The proper smoothed image is shown in Figure 7.10c. This smoothed image is qualified and is further processed for the landmark

recognition. The next section presents the experimental results from landmark recognition of the lower part.

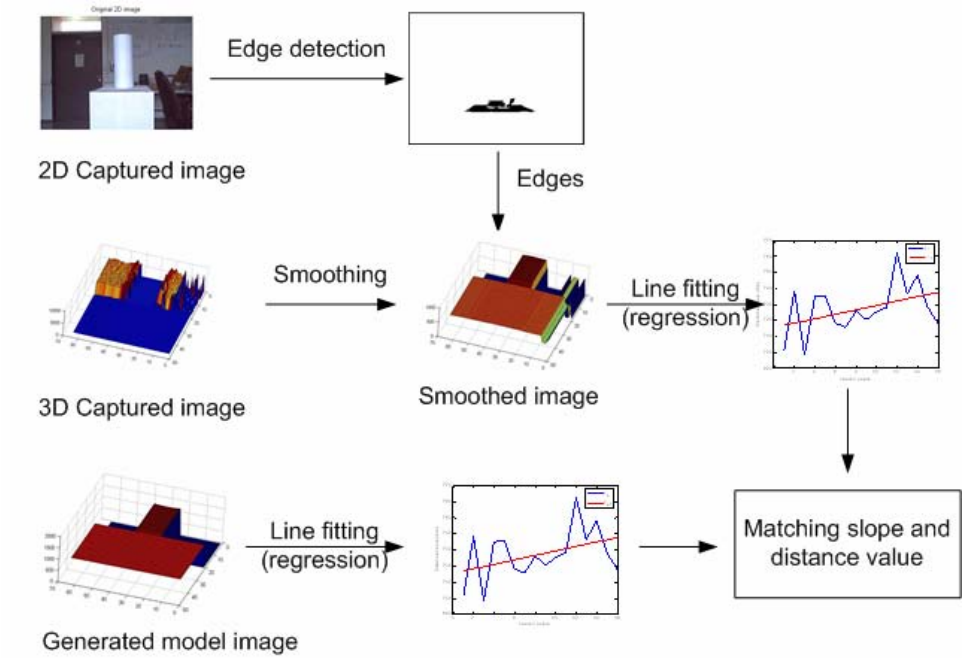
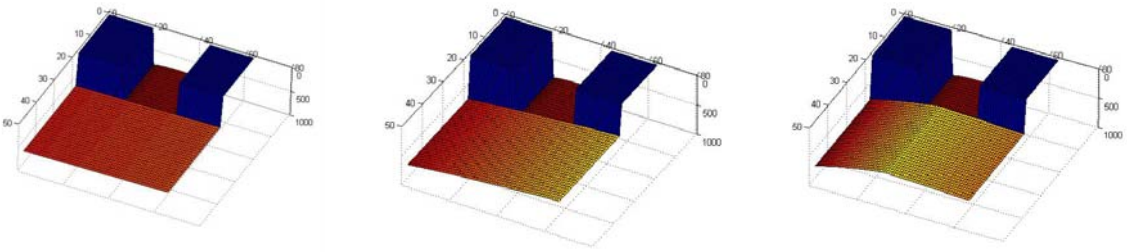
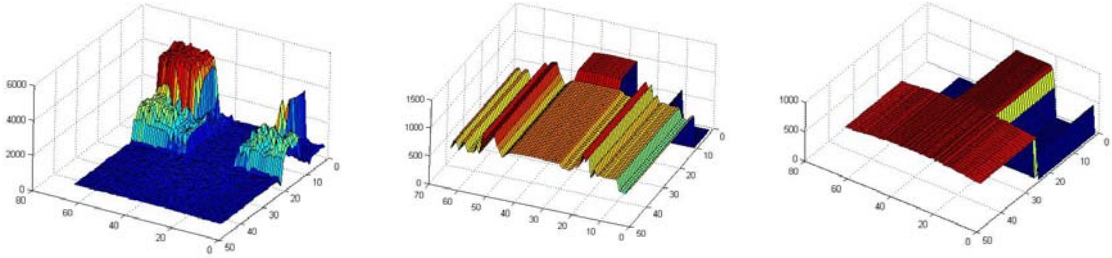


Figure 7.8: Landmark recognition of the lower part using 2D and PMD images.



(a) (b) (c)

Figure 7.9: 3072 pixel model images at 800 mm; (a) 0°; (b) 50°; (c) 60°.



(a) (b) (c)

Figure 7.10: Image smoothing; (a) before; (b) after; (c) after.

7.4 Experimental results

Since we want to compare the experimental results with those of the 16x16 pixel camera in Chapter 6, we set up the same experiment. The landmark is again on a rotary stage and the cameras are standstill. The only different is that we turn the landmark with finer angle position of 5° step, instead of 11.25° step. We capture 2D and 3D images from the Logitech web camera and the PMD camera in the range of from 0° to 90° angle position. The distance position is the same as previously; 800 mm, 1000 mm, and 1200 mm. The test results are broken down into two parts; the horizontal edge detection of 2D image and the model matching.

7.4.1 The results of 2D horizontal edge detection.

Figure 7.11a shows the landmark photos at several angle positions. From the edge detection as explained in section 7.2.1, we find the horizontal edge and the row pixel number of the edge for image smoothing. Here, we present the detected row pixel numbers at each angle position and at 800 mm, 1000 mm, and 1200 mm distance position as shown in Figure 7.11b.

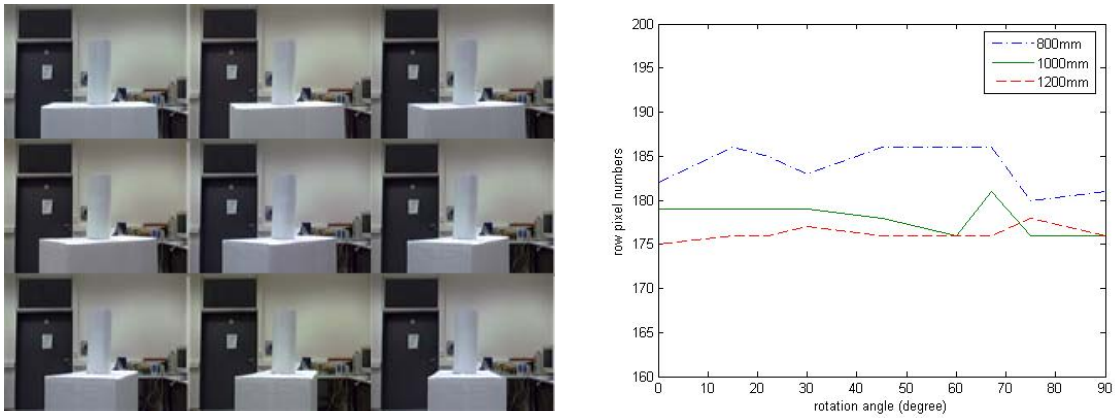
At the 800 mm distance position, edges of all angle positions lie between the 179^{th} and the 186^{th} row pixel. As expected, the row pixel numbers at various rotation angles are closed to each other but they are not exactly equal. Similarly, at 1000 mm, the horizontal edges lie between the 176^{th} and the 181^{th} row pixel. At 1200 mm, the horizontal edges lie between the 175^{th} and the 178^{th} row pixel. These 2D row pixel numbers are converted into the 3D row pixel number using (7.1). By using these results for the image smoothing, we get the matching result as presented in the section below.

7.4.2 The results of model matching.

The model matching result in this section should be compared with that of section 6.3.7.1. In the experiment, the model images are generated at fine steps, at every 1° rotation angle for fine searching. According to the recognition process in section 7.3, the matching result is as shown in Figure 7.12.

The smoothed image of 0°, 5°, 10°, …, 90°, are matched to the generated model image. In total, we have 19x3 = 57 calibrating positions. The matched angles are closed to the reference (actual) angle for all angle positions. At the distance of 800 mm, among 0° to 90°, the largest error is 7° at the angle position of 80° and the STD of the error

among all angle positions is 2.7°. At 1000 mm and 1200 mm, the maximum matching errors is 5° and the STDs of error are only 1.3° and 1.5°, respectively.



(a) (b)

Figure 7.11: Results of edge detection of 2D image: (a) 2D captured image at various positions; (b) the row pixel number from edge detection of each angle position and distance position.

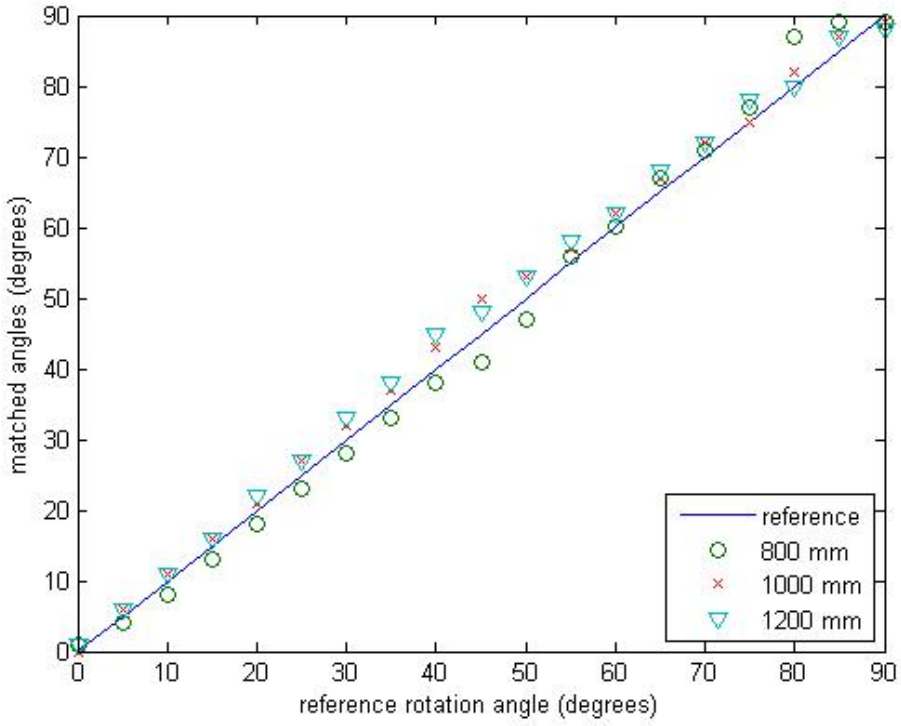


Figure 7.12: Results of matching of the lower part using 2D and PMD images.

These results from 48x64 pixel PMD camera outperform the matching results of the 16x16 pixel camera in the previous chapter since the matched result of 5° step was not possible for the 16x16 pixel camera due to resolution of the camera. Further, the resolution of position calibration was 9 calibrating positions with 45° step as described in section 6.3.7.1. By using the 48x64 pixel PMD image and a 2D image, the resolution is improved from 9 to 57 calibrating positions with the angle positions of 5° step, instead of 45°. The idea of usage of the 2D and 3D images to improve the resolution of the position calibration is therefore successfully implemented.

8 Summary and Perspective

This research was done using various prototypes of the car-like mobile robot platforms and its tele-operated control mechanisms. The developments were made to the robot hardware of several prototypes and robot's autonomous features and robot localization techniques.

Autonomous features: Autonomous features are obstacle avoidance, the 180° turning system for narrow corridors, and the path following control. Obstacle avoidance is categorized as the obstacle collision avoidance and the wall following control. The fuzzy controller and the if-then controller are applied for these obstacle avoidance strategies using four ultrasonic sensors and six infrared sensors. The performance of the algorithms was tested in several different scenarios. The obstacle collision avoidance algorithm is suitable for environments with obstacles that have short boundaries, whereas the wall following algorithm is suitable for long continuous boundary obstacles such as the wall of a building. The obstacle collision avoidance is further exploited for the 180° turn in a narrow corridor and the wall following algorithm is further exploited for the path following control function.

By integrating the obstacle avoidance into the turning process, the robot was able to turn into 180°, when there are unknown obstacles during the operation. The results show that the robot heading is 180° as expected. The path following control for the car-like mobile robot in an unknown environment integrates the basic path following control, and implements both the wall following and the trajectory generation function. The on-line experimental results show that the robot performs the designed path following process successfully and can reach a final desired position and heading.

Robot localization techniques: The robot localization is classified into two types; relative and absolute localization. For the relative localization, we used the discrete extended Kalman filter along with the nonlinear dynamic model as the robot position estimator. We performed experiments on several path types and compared the final position errors and final heading errors, when using odometer, gyroscope, or compass data. The gyroscope estimation provides minority performance in robot final position for the line path. For the other path types, the gyroscope estimation position errors are at least smaller than those of the odometer measurement. The performance in final heading of the gyroscope estimation dominates in most cases the compass estimation and

odometer measurement. As here the gyroscope provides also very good estimation of the whole path, it is considered as the best method to be used for relative robot localization in indoor environment.

For the absolute localization, we exploited the PMD camera's 3D vision and built artificial landmarks for the robot's position calibration. This involved the design and implementation of artificial landmarks, and the development of strategies for landmark recognition and position calibration. The results show that the landmark design and the developed landmark recognition can be applied under the restraints of 16x16 pixels resolution and the time consumption is mainly due to the process of filtering the images. We tested the on-line position calibration and the results show the integrated relative localization with the absolute localization techniques successfully. Nevertheless, the improvement for resolution of the position calibration is necessary.

The resolution of the position calibration is improved by using the 48x64 pixels PMD camera and a 2D camera. We investigate how to do edge detection on the 2D image and to place this edge on the 3D image such that the smoothing process is efficient. Regarding the high resolution of PMD camera, the results show much improvement in the position calibration and also convince that the designed landmark recognition is applicable for higher resolution PMD camera.

The application of the position calibration using 3D artificial landmarks is not restricted to use by the car-like mobile robots but also applicable for other mobile robots. The 3D artificial landmark and the proposed recognition technique would also function with PMD camera with higher resolution and also with other types of 3D vision range sensors.

Appendix A: Microcontroller

The microcontroller specification and its developed software structure are described here in details. As shown in Figure A.1, the minimodule of the 16-bit microcontroller 80C167 CR-LM is manufactured by Infineon and is programmable using C programming language. For our purposes, we exploited the compiler "Keil" from μvision and the code downloader "Flashtool".

Figure A.1: Infineon Minimodule C167 CR-LM

A.1 Hardware specification

The features on the microcontroller are listed as follows:

- 20MHz CPU Clock
- 4 channels of pulse width modulation (PWM) generator
- General Purpose Timer (GPT) and Universal Asynchronous Receiver Transmitter (UART)
- 32 CAPTURE and COMPARE channels for real-time signal processing and generation
- Four additional 2.5MHz 16-bit timers
- A sixteen channel 10-bit AD convertor
- 11/29-bit part 2.0B controller area network (CAN) peripheral
- 111 total pins of input/output (I/O)
- 256kb off-chip flash erasable programmable read only memory (EPROM)
- 64kb SRAM

A.2 Software structure

One of the problems with the control software we designed is the question of how to deal with interrupt routines. During the main loop functions, the interrupt routines have higher priority and are run immediately, when they are called. Therefore, the management of the main loop functions, the appropriate interrupt priority level, and

the frequency of the interrupt routines must be well coordinated. When functions are not organized properly to operate compatibly the microcontroller gets too busy with the interrupt routines and has no time for its main operations. When this is the case, the function on the main operation is delayed. We organized the interrupt settings for solving this problem.

A.2.1 Main loop function

A flow chart showing MERLIN's main operations when under tele-operated joystick and a path control function is shown in Figure A.2. First, the robot initializes all variables, flags, interrupts settings, and PWM settings, and calibrates the gyroscope measurement. After that the robot waits for a command from MerlinClient telling it which operation mode is to be activated the joystick or path control. In the path control mode, the robot performs either its basic path following routine or the path following with obstacle avoidance. In the joystick control mode, the robot performs left hand side or right hand side wall following, or its obstacle collision avoidance routine. After each process is done, the process in question is repeated until the operation mode is changed or until the reset command is issued.

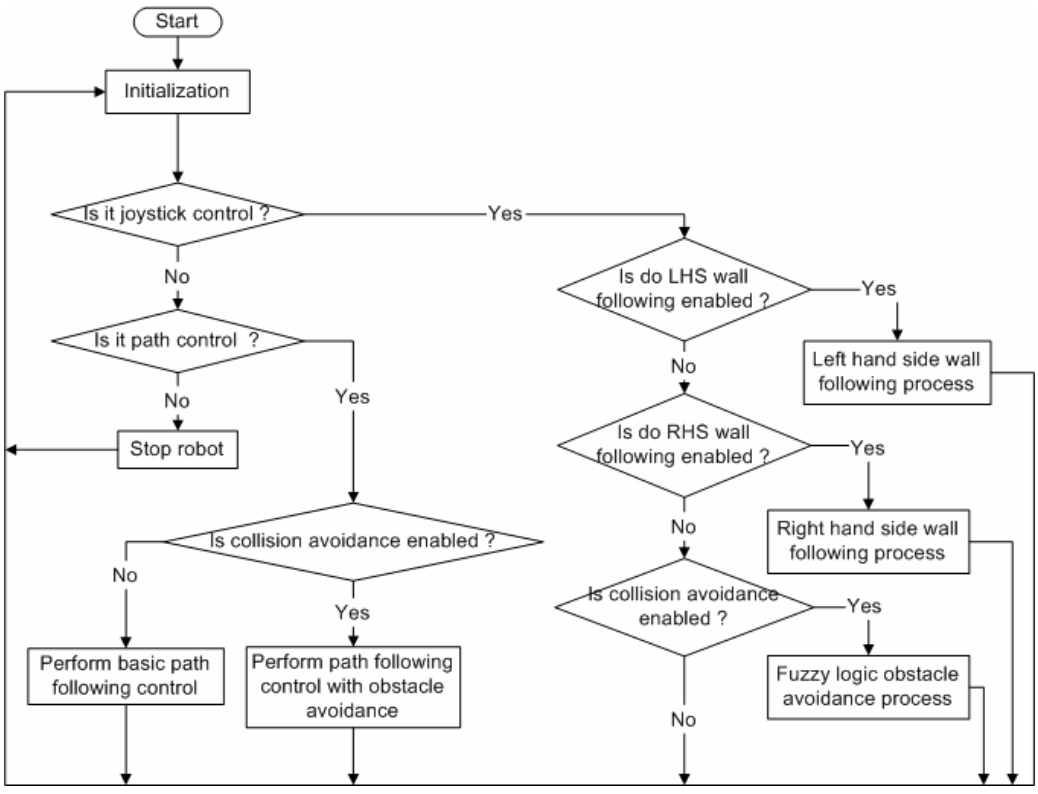


Figure A.2: The main operations of the microcontroller.

A.2.2 Path following control functions

The path control function has two main task categories, line path control and arc path control. Figure A.3 shows which types of paths are recognizable and which commands can be given.

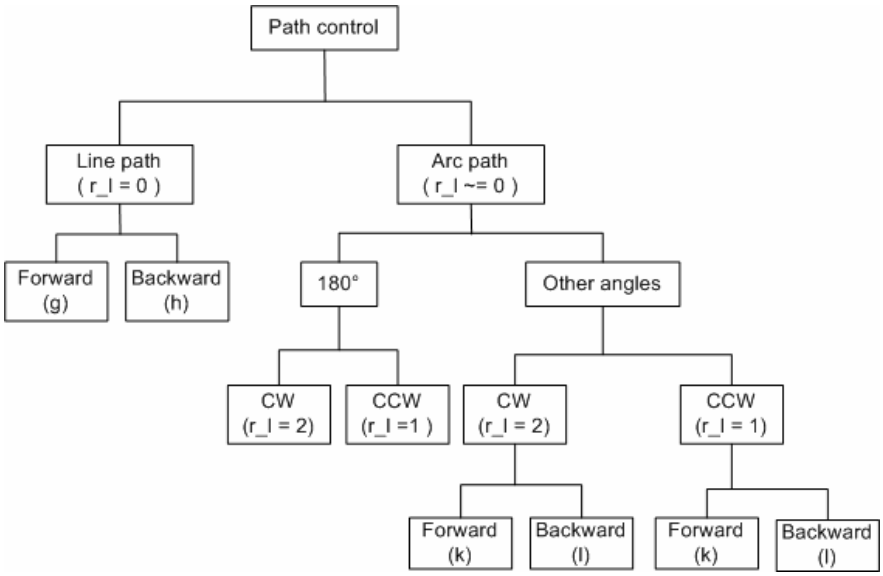


Figure A.3: Recognized path types.

A.2.3 Data packets

The data packet is defined as in Table A.1. There are several types of commands that can be issued from the client PC, such as the path control commands, or the joystick control commands. Conversely, there is only one data packet which can be sent from the robot to MerlinClient.

A.2.4 Interrupt timers and A/D converter

As mentioned earlier, the handling of the interrupt routine is important to ensure the smooth functioning of the main operations. The details of the interrupt priority settings and timer settings of MERLIN are given here. There are 0 to 15 interrupt priority levels for the microcontroller, where 15 is the highest priority level and 0 the lowest. On each priority level, there is also a group priority level ranging from the lowest possible level of 0 the highest possible level with priority 3. The interrupt priority settings are shown in Table A.2 and the timer period settings are shown in Table A.3. The seven channels of the analog to digital (A/D) converter are exploited to

transmit the gyroscope and the infrared signals. Table A.4 presents the channel numbers and their corresponding sensor signals.

Table A.1: Sent and received data packets (on robot).

Type	**Flow direction**	**Length (bytes)**	**Description**	**Details**
's'	robot ←	10	Start joystick control	To reset the variables, calibrate gyro measurement, and start the joystick control.
'S'	robot ←	10	Start path control	To reset the variables, calibrate gyro measurement, and start the path control..
'p'	robot ←	10	Path begin packet	This is the first part of the path command, which specifies the number of total line and arc paths.
'e'	robot ←	10	Path end packer	This is the last part of the path command packet.
'g'	robot ←	10	Path line forward	The data packet consists of the length (cm) and the direction (forward).
'h'	robot ←	10	Path line backward	The data packet consists of the length (cm) and the direction (backward).
'k'	robot ←	10	Path arc forward	The data packet consists of the angle, radius, the arc direction (cw or ccw), and movement direction (forward).
'l'	robot ←	10	Path arc backward	The data packet consists of the angle, radius, the arc direction (cw or ccw), and movement direction (backward).
'j'	robot ←	10	Joystick control command	To set the steering and the speed of the robot, to enable/disable the ultrasonic sensors, to enable wall following, and to enable obstacle collision avoidance.
'i'	robot ←	10	Path control command	To enable path control with obstacle avoidance, to enable/disable the ultrasonic sensors, and to send robot to position x_f, y_f or to set the generated trajectory for the path following process.
'd'	robot →	26 or 42	Sensor data via radio link or via serial port	To update the sensor data that are the driven distance on the left and on the right wheels, the angle position, the velocity of the robot, the angular velocity, the roll, pitch, and yaw angle, and the detected distances to the obstacle. It is also used to update the flags for human-robot operation, to update the robot position, and to request trajectory generation.

Table A.2: Interrupt priority level settings.

Interrupt Vector	Description	Interrupt routine name	Priority level	Group priority
0x26	Sending timer task T6	SendSensorData()	1	0
0x36	Receiving from transceiver	receive()	3	0
0x23	Main timer task T3	Time_interval()	7	0
0x1C	Left-up hall sensor	hallsensor_left_upper()	8	1
0x1D	Right-down hall sensor	hallsensor_right_lower()	8	3
0x1E	Left-down hall sensor	hallsensor_left_upper()	9	0
0x1F	Right-up hall sensor	hallsensor_right_lower()	9	1
0x22	Hall sensor counter T2 timeout	car_not_moving()	10	0
0x15	Ultrasonic sensor triggered	Echo_signal()	11	0
0x1A	Bumper triggered	Bumper()	15	3

Table A.3: Timer period setting.

Timer	Mode	Functions	Time period settings
T2	Timer	Measuring time between hall-sensor interrupts.	Counting forward. T2 increases by 1 every 12.8us. Overflow interrupt 0x22 happens if the time is longer than 12.8us*65535=0.84s.
T3	Reload-mode timer	Main timer task. Gyroscope, ultrasonic, and infrared measurement updates. Speed control, robot direction update, check obstacle, check stop position, set relays on/off for ultrasonic sensors.	Counting downward. T3 decreases by 1 every 0.4us. Underflow causes 0x23 interrupt every 2ms. Reload value from T4 is 4999.
T4	Reload register for T3	T3 reloads the value of T4 after T3 underflows.	T4 = 4999
T5	Timer	Measuring time for ultrasonic sensors.	Counting forward. T5 increases by 1 every 25.6us.
T6	Reload-mode timer	Sending sensor data through transceiver.	Counting downward. T6 decreases by 1 every 0.2us. Underflow causes 0x26 interrupt every 5ms. Reload value from CARPEL is 24999.

Table A.4: A/D converter and specified channels.

Channel	Sensors
1	Gyroscope
2	Infrared front right
3	Infrared front middle
4	Infrared front left
5	Infrared front behind
6	Infrared side right
7	Infrared side left

Appendix B: Graphic User Interface (GUI)

The first step in using the graphic user interface (GUI) is to run the MERLINServer and then start MERLINClient. A connection dialog frame then pops as shown in Figure B.1. The user has to activate a default option enables to show the graphic user interface (GUI). The user clicks on the “Connect” button to have MERLINClient establish a connection to the MERLINServer. The IP address and port number for MERLINServer is shown in the text fields. The dialog tab shows the how the connection attempt is progressing. A full dialog tab indicates that the connection has been successfully established. The joystick control mode can then be initiated. The operator can disconnect when finished by clicking on the “Exit” button. The GUI has is needed for joystick control, path control and for the functions of the PMD camera. The role of GUI in each of these three areas is explained in the following sections.

Figure B.1: The connection dialog frame of GUI.

B.1 The GUI for the joystick control function.

In an unknown environment, where the robot has no prior information about the characteristics of its navigation area, the semi-autonomous operation can be used and is a reliable navigation method. Semi-autonomous operation means that the robot is mainly controlled manually with the joystick as the method of human interface control. Joysticks are well-known plug and play devices, and are readily available on the market. We chose the Microsoft Sidewinder Joystick for the tele-operated control of MERLIN. In the tele-operated control mode, the user's movement of the joystick is translated sent via MERLINClient as x and y direction coordinates. MERLINClient sends the joystick command to the robot and, at the same time, receives continuous sensor data from the robot. Because the robot is in semi-autonomous operation mode, and its movements are being controlled manually by the operator, the automatic obstacle avoidance function is not on. The robot follows the joystick command without the obstacle detection function.

This can sometimes be dangerous for the robot when the user doesn't recognize obstacles but and keeps moving the joystick. The user can cause the robot to crash into an obstacle and destroy itself. An alternative is to activate the automatic obstacle avoidance function while the joystick is being used. The robot then obeys the joystick command and also avoids collision by using the automatic obstacle avoidance until it is free from obstacles. Once the robot has gotten out of the way of the obstacle, it continues following the joystick command afterwards.

MERLIN can also be used with a sensing or force feedback joystick. The haptic interface enables the user to feel what the robot feels. When the robot drives fast, it feels air resistance to its body. The force feedback joystick emulates the aerodynamic force by generating resistance in the joystick that is proportional to the speed of the robot. The users can than feel this resistance when he or she tries to drive the robot forward fast. As a result, the user has to push the joystick to make the robot go faster. The haptic interface feature is implemented on MERLIN using the Immersion Studio java library. The measurements we made show the maximum joystick torque to be –0.165 N*m. In accordance with the concept of aerodynamic force, the generated force within the joystick is proportional to the feedback measured velocity of the robot. The generated feedback force is calculated by using

$$F_{generated} = \frac{1}{2} \times factor \times v^2 \qquad \text{(B.1)}$$

This factor was tuned in experiments and set to 0.3. Figure B.2 shows the GUI for the joystick control mode. The operator can make adjustments to animation screen and command the robot by clicking at the buttons and checkboxes. The functions of the buttons and checkboxes are summarized in Table B.1.

The data panel on the right hand side of the GUI shows the numerical values from the following measurements:

- the ultrasonic sensors at the right, middle, left, and rear of the robot,
- the distances driven by both the left and the right wheels, and robot speed,
- angle position (heading),
- the roll, pitch, and yaw angles from the 3 DM compass,

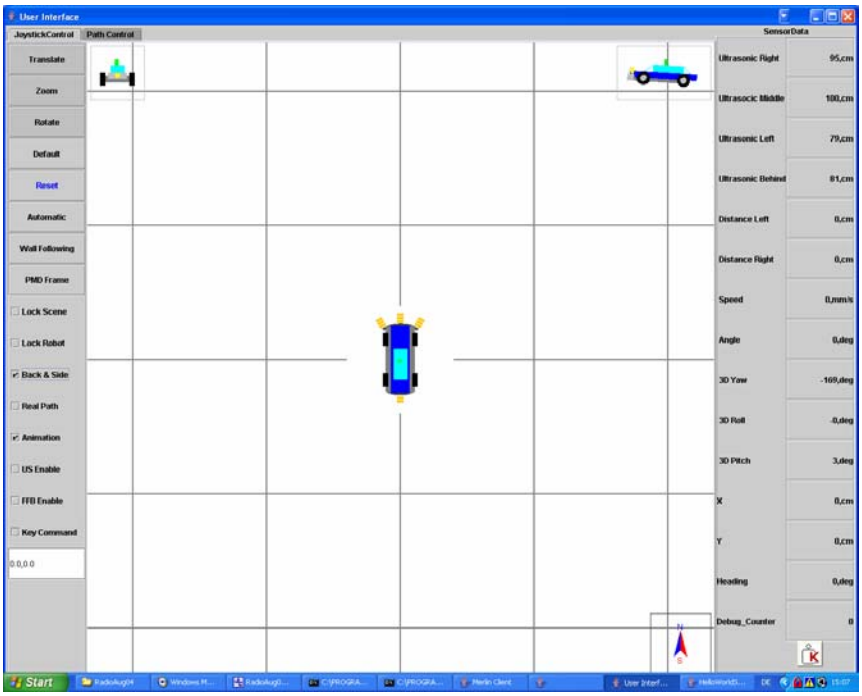


Figure B.2: The joystick control panel.

Table B.1: The functions of the buttons and checkboxes in the GUI for the joystick.

Name	Function
Translate	To translate the scene into the directions: up, down, left and right.
Zoom	To zoom in and zoom out by clicking on the scene and dragging the mouse.
Rotate	To rotate the scene by clicking on its image and dragging the mouse either in a clockwise or counter clockwise direction.
Default	To change the scene back to the original scene.
Reset	To stop the robot and clear all robot data.
Automatic	To have the robot perform automatic obstacle avoidance using its ultrasonic sensors and infrared sensors. The robot follows the joystick commands completely until it comes across an obstacle. When there is an obstacle present, the robot obeys the joystick command but uses its fuzzy logic controller to avoid collision. The fuzzy logic controller is deactivated when there are no more obstacles present. The car then continues to follow the joystick commands. However, the robot obeys the joystick except when the user wants to move the robot backward movement.
Wall following	To have the robot perform automatic wall following in the left hand or right hand side mode.
PMD Frame	To open the panels for the PMD camera.
Lock Robot	To have the robot image shown always in the middle of the scene.
Back & Side	To show the back view and side view tilting and rolling positions.
Real Path	To show the estimated path that the robot has driven.
Animation	To show the robot animation.
US Enable	To enable the ultrasonic sensors.
FFB Enable	To enable the force feedback interface.
Key Command	To use the arrow cursors on keyboard instead of the joystick, “up” increases the speed in the forward direction, “down” increases the robot speed in the backward direction, “left” turns the left front wheels, and “right” turns the right front wheels.

- the estimated position X, Y, and estimated robot heading from the Kalman filter and the nonlinear dynamic model.

Note that there are also warning and status messages available, such as “Merlin is crashing into objects” or “Gyro is calibrating”.

B.2 The GUI for the path control function.

The operator can switch between the joystick control mode and the path control mode by clicking at the tabbed panel. In the path control mode, the robot stands still until it receives the path commands. The operator starts by drawing the desired path on the panel and then clicks "send" to transmit the data to the robot. After that MERLINServer sends the path commands to the robot and the robot starts moving under the path following control. The path control panel is shown in Figure B.3.

There are four types of path commands: line forward, line backward, arc forward, and arc backward. The obstacle collision avoidance function is also an option for the path following mode. The operator can click on the checkbox “Collision Avoidance” for automatic obstacle avoidance during the path following mode. The functions of the buttons and checkboxes on the left panel are summarized in Table B.2.

B.3 The GUI for the PMD camera.

The graphic user interface for the PMD camera consists of three parts, the left command panel, the graphics panel, and the right command panel, as shown in Figure B.4.

Table B.2: The functions of the buttons and checkboxes for the path following control commands.

Name	Function
Line forward	To specify the start and end of the line path in a forward direction.
Line backward	To specify the start and end of the line path in a backward direction.
Arc forward	To specify the start and end of the arc path in a forward direction.
Arc backward	To specify the start and end of the arc path in a backward direction.
Clear Line-Arc	To delete the paths drawn.
RealPath	To show the estimated driven path.
Clear RealPath	To delete the estimated driven path from the scene.
Send	To send the paths to the robot and start the path following function.
Reset	To stop the robot and clear the robot data.
Collision Avoidance	To enable the automatic obstacle collision avoidance during the path following.
US Enable	To enable the ultrasonic sensors.

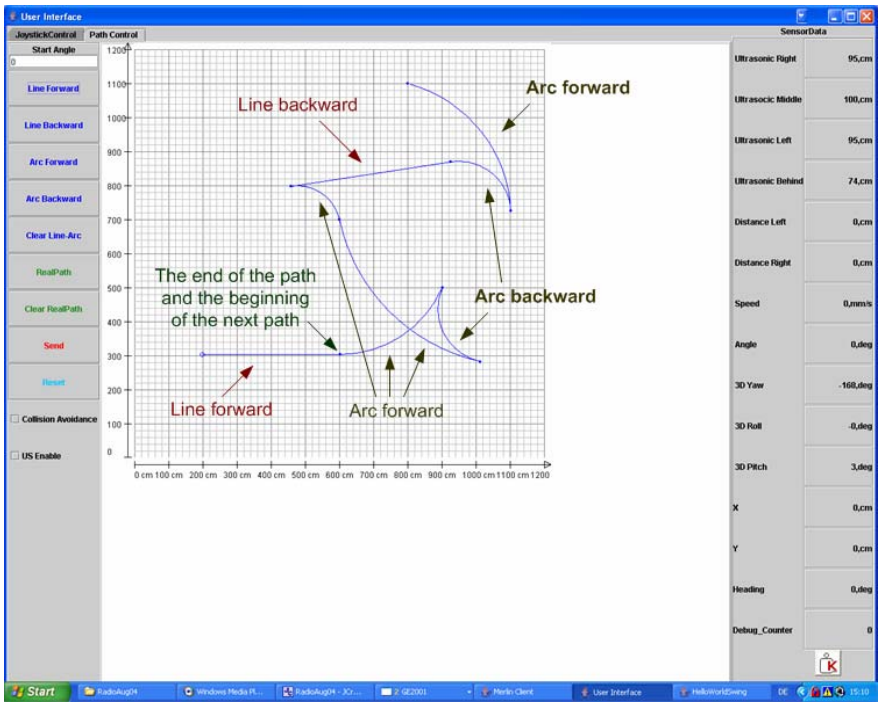


Figure B.3: Path following control panel.

The operator establishes a connection to the camera by clicking on “PMD Connect” on the left hand panel. The PMD server must already be running for this to function. After the connection is established, the operator has to calibrate the PMD camera by clicking “PMD Calibrate” and wait until the current image is updated on the panel. As of this moment, the camera is ready and the operator can choose to take measurements by clicking on “PMD Image” or “PMD Video”. The filtered image number is counted automatically after each measurement.

The operator can calibrate the robot's position by starting the Kalman filter and waiting until the filtered image looks like the current measured image. At this point the filtered image is ready for use. The operator can click “Calibrate Position” and wait until the robot position is updated on the GUI of the joystick control panel. At the same time as the position update is given on the joystick control panel, the matched results will appear on the PMD panel. Please note that the position calibration is available only for the joystick control mode. It is not recommended that a position calibration be done when the filtered image is not yet ready. This would cause an error in the position calibration. The functions of the buttons and message boxes on the control command panel are described in Table B.3.

The middle panel plots two images. The upper image is the generated model image and the lower image is the actual image from the measurement. Before the measurement starts, the lower image is empty. On the right command panel, the user can specify the distance, side distance, and angle of translation for generating the model image. When the user clicks the “Plot” button, the generated model image is plotted in the upper part of the graphics panel. The slide bar are also provides for convenience in fast and rough observation of the various measurement values, the distance, side distance, and angle positions.

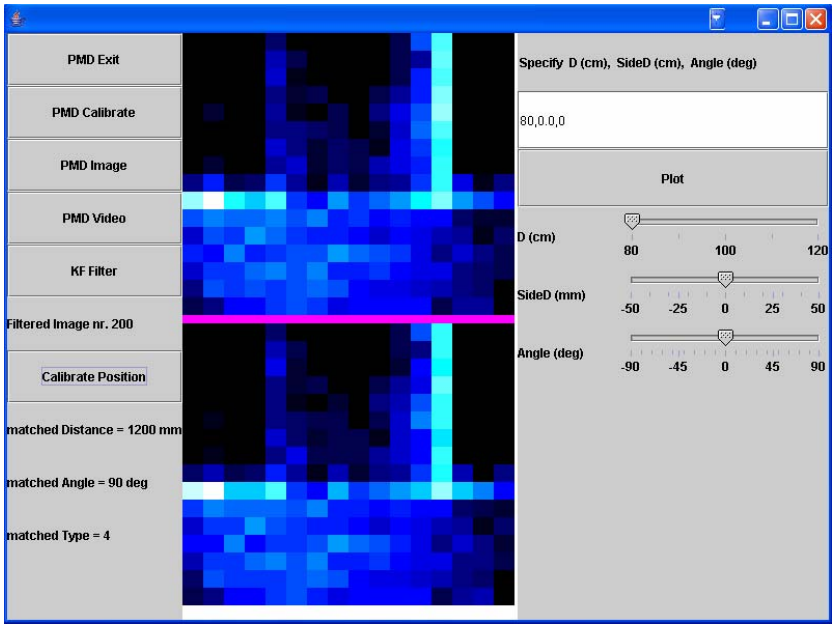


Figure B.4: The PMD camera control panel.

Table B.3: The functions of the buttons on the left command panel of the GUI for the PMD camera.

Name	Function
PMD Connect	To connect the PMDClient to PMDServer.
PMD Calibrate	To calibrate the PMD camera.
PMD Image	To measure and plot on the graphic panel.
PMD Video	To measure continuously as video.
Filtered image nr.	To show the number of filtered images.
Start KF Filter	To start the Kalman filtering.
Calibrate Position	To calibrate the robot position.
matched Distance	The distance position as determined by the landmark recognition function.
matched Angle	The angle position derived from the result of landmark .
matched Type	The landmark type result determined by the landmark matching function of the landmark recognition process.

Appendix C: PC104

The specification of the selected PC104 and technical notes are given here. The CPU-M1 has the following features:

- CPU: Pentium III 933 MHz
- IDE: PCI-IDE, 2 drives
- Memory: 384 MB
- Video: Intel 815E integrated 3D graphic controller
- Ethernet: 10/100 Mbit
- PS/2 keyboard and mouse port
- 3 ½" floppy support
- 2 serial ports RS-232 compatible
- 1 parallel port
- 2 USB ports
- PC/104-plus compatible
- Size: 96 x 90 mm
- Linux Operating System

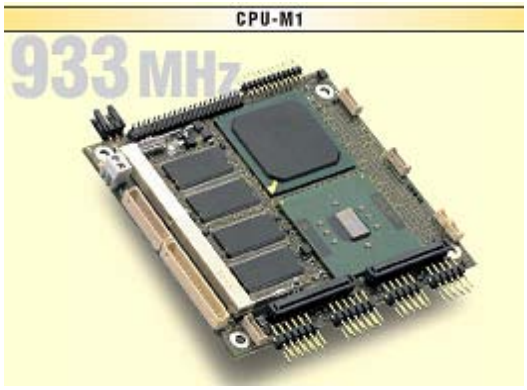


Figure C.1: The CPU-M1

When Redhat 9.0 OS is used, it is necessary to install the Javax Comm for java serial port interface software, the Orinoco Linux driver for Orinoco Gold USB WLAN adapter, and the java runtime environment. The two com ports located on the PC104 are for the microcontroller and the PMD camera. The USB is expanded into 2xUSB and 2xRS232 by using the Edgeport/22c from Digi International. The Linux driver of this device is installed. Note that after connectin Edgeport/22c the names of the COM ports under the Linux system are also changed from "/dev/ttyS0" to "/dev/ttyUSB0" and from "/dev/ttyS1" to "/dev/ttyUSB1". An RXTX library for javax comm is also required.

References

[BES 88] Besl P.J. (1988), Surfaces in range image understanding, Springer-Verlag, chapter 2, p. 34, ISBN: 0-387-96773-7.

[BOR 96] Borenstein, J., Everett, H.R., Feng, L., and Wehe, D. (1996), Mobile Robot Positioning: Sensors and Techniques, Invited paper for the Journal of Robotic Systems, Special Issue on Mobile Robots. Vol. 14, No. 4, April 1997, pp. 231-249.

[BRO 83] Brown, R.G. (1983), Introduction to random signal analysis and Kalman filtering, John Wiley & Sons, USA, p. 300, ISBN: 0-471-08732-7.

[CHO 01] Choset,H., Nagatani, K. (2001), Topological Simultaneous Localization and Mapping, (SLAM): Toward Exact Localization Without Explicit Localization, IEEE Trans. on robotics and automation, 17(2), APRIL 2001, pp. 125 – 137.

[DRI 04] Driewer, F., Baier, H., Schilling, K., Pavlicek, J., Preucil, L., Ruangpayoongsak, N., Roth, H., Saarinen, J., Suomela, J. , Halme, A. , Kulich, M., Kout, J. (2004), Hybrid Telematic Teams for Search and Rescue Operations, IEEE International Workshop on Safety, Security and Rescue Robotics, Gustav Stresemann Institut, Bonn, Germany, May 24-26, 2004, ISBN: 3-8167-6556-4.

[DRI 93] Driankov, D., Hellendoorn, H., Reinfrank, M.(1993), An Introduction to Fuzzy Control, Springer-Verlag, ISBN: 0-387-56362-8.

[FAN 90] Fan T. J. (1990), Describing and recognizing 3-D objects using surface properties, Springer-Verlag, chapter 4, p. 55, ISBN: 0-387-97179-3.

[FER 96] Ferretti, E., Oriolo, G., Panzieri, S., Ulivi, G. (1996), Learning Nice Robust Trajectories for a Car-Like Robot, Int. Symp on Intelligent Robotic Systems, Lisbon, PT, July 22-26.

[FOX 99] Fox, D., Burgard, W., Thrun, S. (1999), Markov Localization for Mobile Robots in Dynamic Environments, Journal of Artificial Intelligence Research 11 (1999), pp. 391-427.

[FRA 98] Franklin, G. F., Powell, J. D., Workman, M. (1998), Digital Control of Dynamic Systems, 3rd edition, Addison Wesley Longman, p. 59, ISBN: 0-201-82054-4.

[FUC 04] Fuchikawa, Y., Kurogi, S., Matsuo, K., Miyamoto, S., Nishida, T. (2004), A Vision-Based Navigation System Using Guideposts for Mobile Robot, The 8th World Multiconference on Systemics, Cybernetics and Informatics, July 18 - 21, 2004 Orlando, Florida, USA.

[GOE 99] Goel, P., Roumeliotis, S. I., Sukhatme, G. S. (1999), Robot Localization Using Relative and Absolute Position Estimates In Proc. 1999 IEEE/RSJ

International Conference on Intelligent Robots and Systems, Kyongju, Korea, Oct. 17-21, pp. 1134-1140.

[HAN 02] Han, M., Lee, S., Park, S.K., Kim, M. (2002), A New Landmark-Based Visual Servoing with Stereo Camera for Door Opening, International Conference on Control, Automation and Systems, pp. 1892-1896.

[HES 02] Hess, H., Albrecht, M., Schwarte, R. (2002), PMD – New detector for flourescence lifetime measurement, Opto 2002.

[JAI 90] Jain, R. C., Jain, A. K. (1990), Analysis and interpretation of range images, Springer-Verlag, chapter 5, page 225, ISBN: 0-387-97200-5.

[JAN 02] Jang, G., Kim, S., Lee, W., Kweon, I. (2002), Color Landmark Based Self-Localization for Indoor Mobile Robots, International Conference on Robotics & Automation, May 11-15, 2002, Washington, DC, USA, pp1032-1042.

[KLA 02] K. Schilling, H. Roth, O. Rösch (2002), Mobile Mini-Robots for Engineering Education, Global Journal of Engineering Education 6 (2002), p. 79–84.

[KUH 04] Kuhle, J., Roth, H., Ruangpayoongsak, N. (2004), Mobile Robots and airships in a multi-robot team. The 1st IFAC Symposium on Telematics Applications in Automation and Robotics, Helsinki University of Technology, Finland, pp. 67-72.

[MAY 02] Mayer, G., Utz, H., Kraetzschmar, G.K. (2002), Towards Autonomous Vision Self-Calibration for Soccer Robots, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2002).

[MAY 91] Mayr, R. (1991), Verfahren zur Bahnfolgeregelung für ein automatisch geführtes Fahrzeug, Dissertation, University of Dortmund, Dortmund, Germany.

[MEL 02] Mellodge, P. (2002), Feedback Control for a Path Following Robotic Car, Master Thesis, Virginia Polytechnic Institute and State University.

[PRU 05] Prusak, A., Roth, H., Schwarte, R. (2005), Application of 3D-PMD Video Cameras for Tasks in the Autonomous Mobile Robotics, 16th IFAC World Congress, July 4-8, Prague, Czech Republic.

[RIE 40] Riekert, P. and Schunck, T.E. (1940), Zur Fahrmechanik des gummibereiften Kraftfahrzeugs. Ing. Arch., Vol. XI, pp. 210-224.

[ROE 03] Roefer, T., Jüngel, M. (2003). Vision-Based Fast and Reactive Monte-Carlo Localization. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-2003), Taipei, Taiwan. pp. 856-861.

[RUA 05a] Ruangpayoongsak, N., Roth, H., Chudoba, J. (2005), Mobile Robots for

Search and Rescue, IEEE International Workshop on Safety, Security and Rescue Robotics, International Rescue System Institute, Kobe, Japan, June 6-9, 2005.

[SIM 05] Sim, R., Roy, N.(2005), Global A-Optimal Robot Exploration in SLAM, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Barcelona, Spain.

[SUK 92] Suk, M., Bhandarkar, S.M. (1992), Three-Dimensional Object Recognition from Range Images, Springer-Verlag, Chapter 5, p. 103, ISBN: 4-431-70107-9.

[SUR 03] Surmann, H., Nüchter, A., Hertzberg, J. (2003), An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments, Robotics and Autonomous Systems 45, pp. 181–198.

[SCH 04] Schwarte, R. (2004), Breakthrough in Fast 3D-Imaging Using PMD- and OEP-Technology, Duisburg, 2004, IMS Workshop in Duisburg, May 25-26 2004.

[TIM 04] Timmer, F. (2004), Der Entwurf eines mathematischen Fahrzeugdynamikmodells für einen autonomen mobilen Roboter und die Regelung seines Kurswinkels seiner Geschwindigkeit und seiner Schwerpunktposition im Ruam, Master Thesis, University of Siegen.

[TRE 04] Trevai, C., Ueda, R., Moriya, T., Arai, T. (2004), Integration of Uniform Monte Carlo Localization Method for Mobile Robot with Sonar Array, Intelligent Autonomous Systems 8, IOS, ISBN 1-58603-414-6.

[USH 02] Usher, K., Ridley, P., Corke, P. (2002), Visual Servoing of a car-like vehicle – An application of omnidirectional vision, Proc. 2002 Australian conference on Robotics and Automation, Auckland, 27-29 November 2002, pp. 37-42.

[WEL 02] Welch, G., and Bishop, G. (2002), An Introduction to the Kalman Filter, UNC-Chapel Hill, USA, TR 95-041.

[YOO 02] Yoon, K.J. , Kweon, I.S. (2002), Landmark Design and Real-Time Landmark Tracking using Color Histogram for Mobile Robot Localization, Mobile Robots XVI, Douglas W. Gage, Howie M. Choset, Editors, Proceedings of SPIE Vol. 4573 pp. 219- 226.

[ZHA 04] Zhang, P., Milios, E.E., Gu, J. (2004), Underwater Robot Localization using Artificial Visual Landmarks, IEEE International Conference on Robotics and Biomimetics, Shenyang, China, 2004, Aug. 22-26. Paper no 67.