LOCALIZATION AND OBJECT DETECTION FOR OUTDOOR MOBILE ROBOTS WITH 2-D LASER SENSOR

by

Yi Lu, M.A.Sc

A thesis submitted to

The Faculty of Graduate Studies and Research

in Partial Fulfillment of

the requirements for the degree of

Doctor of Philosophy

Department of Mechanical and Aerospace Engineering

Ottawa-Carleton Institute

Carleton University

Ottawa, Ontario

August 20, 2008

Abstract

Applications of outdoor mobile robots can be found in various diversified fields including defense, agriculture, mining and space exploration. To carry out their tasks successfully, the robots have to locate the pose in the environment and detect the objects around them. This thesis presents the development and implementation of localization enhancement and object detection using a laser range sensor and the map service used for localization and object detection intentions.

The problem of gate-like object recognizing and crossing is described at first. The novel concept of signature is proposed and applied. The algorithm based on the signatures from the special objects is developed. A “gating controller” is designed by extending the classical parking controller. The simulations and experiments demonstrate the mobile robot approaching and crossing the gate-like object using the developed algorithms.

For the generalization of the concept of signature from the special object, a data association algorithm based on geometry features is developed. Using this algorithm, the data from a 2-D laser range sensor is compared with a reference scan from the simulator of the physical laser sensor to locate the pose of the robot. Based on the matching algorithm, two JAUS compatible components, Range Based Pose and Object Tracking, are developed and integrated into the mobile robot system. Simulation and experiment about localization and object tracking tasks are discussed.

To handle multiple possibilities of the robot's pose in the environment with repetitive features, a particle filtering algorithm is proposed. The algorithm uses the results from data association to calculate the probability of the particles. Particularly, some of the particles will be distributed through a defined range to ensure catching and maintain the multiple possibilities of the pose during the robot running in the environment. The simulation of the proposed algorithm is presented and discussed.

As one of the important elements for localization, the implementation of the map service functionality for mobile robots is addressed. A JAUS compatible component, the Map Data Service, is developed and integrated into the mobile robot system. This component supplies the required map data to other localization and object detection components like the developed Range Based Pose and Object Tracking components.

The possible future work is discussed at the end.

Acknowledgements

I would like to thank my supervisor Prof. J. Sasiadek for his continuous support, valuable guidance and motivation during these years. I will always remember his patience and generosity showed to me.

I would like to express my gratitude and appreciation to Dr. V. Polotski for giving advice through this research. I would like to thank all the staff in the department for their help.

I would like to thank the Ontario Graduate Scholarship (OGS), the Ontario Centre of Excellence (OCE), and the Department of Mechanical and Aerospace Engineering of Carleton University for their financial supports.

Most of all, I would like to sincerely thank my wife and my son for their continued support and confidence in me!

Nomenclatures and Symbols

APR: Anchor Point Relation;

CLS: Complete Line Segment;

CSM: Combined Scan Matcher;

DARPA: The Defense Advanced Research Projects Agency;

DGPS: Differential GPS;

EKF: The extended Kalman filter;

GIS: Geodetic Information Systems;

GPS: Global Positioning System;

IDC: Iterative Dual Correspondence;

JAUS: the Joint Architecture of Unmanned Systems;

KF: The Kalman filter;

LMS: Laser Measurement Sensor;

MCL: Monte Carlo Localization;

OT: Object Tracking;

PDF: Probability Distribution Function;

RBP: Range Based Pose;

SLAM: Simultaneous Localization and Mapping;

UGV : Unmanned Ground Vehicles;

UAV: Unmanned Aerial Vehicles;

X : the system state;

$[x, y, \theta]$: the robot pose in a planar coordinate system;

$[\Delta x, \Delta y, \Delta\theta]$: the pose offset of the robot;

$[\hat{x}, \hat{y}, \hat{\theta}]$: the updated pose estimate;

$X(k)$: the system state at time k ;

w^i : the weight of the i^{th} particle;

$X^i(k)$: the system state from the i^{th} particle at time k ;

u : the control input;

z : the sensor measurements;

v : the noise in measurements;

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

Traditionally, robotics has played an important role in modern society. Robots were first used for repetitive tasks in industry. From the 1980's, the role of robots was further extended to cover a large number of autonomously controlled mobile agents. Unmanned Ground Vehicles (UGV) and Unmanned Aerial Vehicles (UAV) are the most popular applications among these robot systems.

Security tasks, especially in some dangerous areas, have often resulted in serious personal loss. Many security-like tasks, such as border patrol, infrastructure surveillance and mining area exploration, are very difficult or dangerous for a human security team. Sending a UGV or UAV to do these security tasks will significantly reduce the risk to human life. For this destination, an intelligent security mobile robot, GRUNT, is being developed by Carleton University and a company from industry. The security robot can carry out many outdoor tasks alone or in cooperation with other mobile robots in critical environments such as borders, important infrastructures.

This thesis work is focused on the development of such mobile robot system, especially on the localization related problems.

1.1 Background

Robots help humans perform dull or dangerous tasks. As a member of the robot family, the real size, outdoor mobile robot has numerous potential applications like patrolling of critical infrastructures, working in hazardous environments, providing surveillance of borderlines, and working on construction etc.

There are many challenging problems which have to be solved before people can have a fully operational, real size robot to carry out the above tasks. Many of these problems are related to localization. Localization is a process the robot goes through to find its position and orientation by using the information from the sensors or/and available map. In combination, pose is used to express the position and orientation. This thesis is concerned with the improvement of localization in outdoor environments.

To describe the approach to localization improvement in this work, several basic concepts need to be defined first. Two of them - Map and Data Association are addressed below. Then Localization and JAUS are described in two separate subsections. The last subsection describes the structure of the thesis, the publications and main contributions.

Map

A map is a model of the world. It is an essential ingredient for a robot to implement a successful localization procedure. The maps used for a mobile robot could be of many different types, e.g. occupancy maps, metric maps, vector/feature maps and graphical maps. A fully autonomous system always requires the map. For outdoor application, the map has to meet requirements in presentation, data association, adjustability, computation, and storage. Another issue that needs to be considered is the interface of the map. The

interface has to be compatible with the existing map system, which could reduce the cost and time of map building for a large map.

Data association

All kinds of localization are dependent on data association. Data association is the process of finding a precise correspondence between the current data that the robot collects from its environment using its sensors and the stored map data or the previous collected sensor data. In outdoor environments, the varied weather conditions and bumpy ground make the sensing and pose estimation very tough. The data association algorithm thus becomes much more complex. Also, the algorithms are distinct for different sensor data. For some practical mobile robot applications such as security patrol in particular environment, the robot must have the ability to handle both the known and unknown objects. The noises from varied natural disturbances also increase the difficulties of data association procedure.

1.2 Mobile Robots Localization

An outdoor mobile robot could carry out various complex tasks like the patrolling of critical infrastructures, the surveillance of borderlines or construction works, etc. For these tasks, many sub-tasks are essential such as target attainment, area exploration, crash avoidance and object following. All of these require the robot to know its position and orientation (pose). Localization is a primary capability for a mobile robot. Without the

ability of localization, an outdoor robot can not plan nor perform the actions which require the information out of the immediate sensing range.

1.2.1 Classification of mobile robot localization

Many different localization methods are applied in the mobile robot field. According to the independence or dependence upon the environment map, three categories of localization method can be distinguished: dead reckoning, localization with a prior map and Simultaneous Localization and Mapping (SLAM).

Dead Reckoning

Dead reckoning also known as deduced reckoning is a general localization method for navigation. During the dead reckoning process, the robot's pose is estimated by summing its movements which are measured by the sensors, such as accelerometer or odometer. Because it is simple, inexpensive and independent from a map, dead reckoning has been adopted by many mobile robot systems. The weakness of dead reckoning is that there isn't any internal mechanism to reduce or recover the errors from the measurement procedure. The result is that the measurement errors will be accumulated with the summing process. These accumulated errors make the estimated pose unreliable. In [1, 2, 3], some assistant methods are developed to solve this problem. Normally, dead reckoning is not used as the exclusive localization method for long term running outdoor mobile robots.

Localization with a prior Map

If the increasing uncertainty from dead reckoning could be bounded by some other assistant measurement, the localization procedure would be more reliable. The other

assistant measurement could be some absolute feature matching, which provides one time pose correction. Unlike the sensors used for dead reckoning, some other sensors such as radar, laser range finder, sonar and camera are needed to collect the data for the feature matching process. On the other hand, the objects which provide the absolute features have to be pre-defined. Usually, these featured objects or landmarks are included in a map of the environment. Such an environment map is called a prior map. The procedure of localization with a prior map includes two steps: observation and association. The sensors perceive the environment and collect the information in observation step. Then, the sensor data is associated with the corresponding map data in association step. The estimated pose can be extracted from the association results. Localization with a prior map is a popular localization method for mobile robots. It can be used for the primary localization or as the assistant localization functionality for mobile robots. The limitations for it are map related problems such as map creating, map maintenance, and the availability of the map, data sensing and data association. Later in this thesis, this localization method will be discussed.

Simultaneous Localization and Mapping

To overcome dependency on a map, a new mobile robot localization method, Simultaneous Localization and Mapping (SLAM), was introduced by [4] in 1991. SLAM records the featured objects or landmarks in the environment using the sensors and locates these objects or landmarks with the mobile robot pose estimation. At the same time (in fact later), these located objects or landmarks could be used to improve the robot's pose estimation. Because the robot continues to observe the objects or landmarks, the locating of these objects is done with increasing accuracy and certainty. SLAM allows

mobile robots to carry out navigational tasks in unknown environments (no prior map). For SLAM, the data association algorithm and the computation cost are the main problems.

Compared to localization with a prior map, SLAM can explore expended areas and is less dependent on existing maps. As such, the required processing is more complex than localization with a prior map. Similar to the localization problem with a prior map, one of the critical steps in SLAM is data association. For many mobile robot applications, if the map related limitations in localization with a prior map could be resolved, localization with a prior map still is the first choice, considering its easier processes and reliability.

1.2.2 Outdoor mobile robots localization and Global Positioning System (GPS)

Mobile robot systems have been applied in both indoor and outdoor environment. The requirements for localization are different for indoor and outdoor tasks. The distinctions are resulted from the scale of the running area, the conditions of the terrain, the types of robotic platforms, the speeds of the robots, the sensor equipments and the tasks to be expected.

Compared to the indoor environments, the outdoor applications usually have a larger scale. The ground outdoor is usually more rugged than the smooth indoor planar surface. These affect the selection of the sensors and the related data processing methods. Most of the robot platforms for outdoor applications are full-size, off-road vehicles. The relatively high speed and the complexity of vehicle model result in much more noise and distortion to the sensing process.

Global Positioning System (GPS), not available for indoors application, has become an essential localization mean for outdoor mobile robot systems. However, the using of GPS still experiences the problem from the signal obstruction, uncertainty and availability [5, 108]. These problems retarded GPS as the exclusive localization sensor for outdoor mobile robots. For many applications, GPS works along with other sensors to ensure the full localization functionality.

1.3 JAUS

Since the 1990s, a large number of unmanned system products have been developed and introduced into different fields. Many of these systems are task dependent and incompatible with others. To standardize the unmanned systems and economize the development of such systems, the Joint Architecture of Unmanned Systems (JAUS) Working Group started to build an open standard for the architecture of unmanned systems in 1998. This open standard defines the structures and the message sets used for unmanned systems and some controlling/monitoring processes.

This section provides a short overview of JAUS based on information available from http://www.jauswg.org/.

1.3.1 JAUS requirements

The objective of JAUS standard is to reduce the cost of development and using through the unmanned systems' life-cycle. For this purpose, all the unmanned systems with JAUS

capabilities have to meet the following requirements according to the official JAUS document [7]:

***Platform Independence*:** The JAUS-compliant components should be interoperable which requires no assumptions about the platforms.

***Mission Independence*:** "JAUS compliant system must isolate mission-specific functions from the generic functions" [7]. Therefore the number and types of applications or missions for systems could be changed by the developers or users.

***Computer Resource Independence*:** JAUS compliant systems must be independent from the computer system hardware. This ensures the system to be applicable to all other unmanned systems.

***Technology Independence*:** JAUS does not include the technology solution, but it gives the requirements for any possible solutions.

***Operator Use Independence*:** For any task, JAUS does not define the way how to perform it. The operator/user of the system has the power to choose the correct approach.

1.3.2 JAUS System Structure

A system based on JAUS has four distinct classes of elements. They are systems, subsystems, nodes, and components. A usual system topology based on JAUS architecture is shown in Figure1.1.

The system groups the sub-systems by logical relations. For most applications, one JAUS system includes one or more subsystems working together. Within a JAUS system, the subsystems, nodes, and components can communicate with each other by the defined input/output messages.

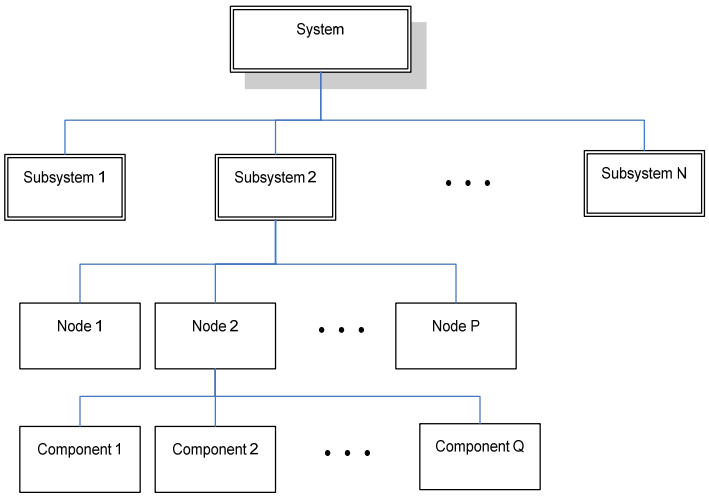


Figure 1.1 JAUS based system topology [7]

"A subsystem is an independent and distinct unit" [7]. To meet the functional requirements of the unit, the subsystem must include the related nodes and components. The referred maximum number of subsystems within one JAUS system is 256.

"A node is composed of all the hardware and software assets necessary to support a well-defined computing capability within the subsystem" [7]. The message flow in JAUS system is managed by node manager in this level. The referred maximum number of node within one subsystem is 256.

The Component is the simplest element in JAUS structure. A component always encapsulates a unique functionality. The related input/output messages which are necessary to run this functionality are provided [43]. 28 components exist in the new JAUS Reference Architecture. Users may develop components themselves to achieve the different tasks. The referred maximum number of the components within one JAUS node is 256.

The purpose of JAUS standard is to generalize the design and structure of different unmanned systems and make these systems interoperable. JAUS can not include all the scenarios for the different unmanned system. However, based on this open standard, developers/users may build their system independent from the special task. A user-defined component can be developed by the user according the JAUS standard. These user-defined components will be eventually recognized by the JAUS community and integrated into the JAUS system. To enable this, JAUS defines the messages which standardize the interface between these components. Later in this thesis, the detailed procedure of how to develop a user-defined JAUS component will be discussed.

1.4 Thesis Organization and Contributions

This thesis is concerned with the development of methods and algorithms to improve the localization performance for mobile robots in large-scale, outdoor environments using a 2-D laser sensor. The developed algorithms have been tested by physical experiments and/or simulations. The algorithms have been implemented in the form of modules or component of JAUS system. These components have been integrated into the mobile system. The robot with the implemented system on it has been used for security tasks.

Chapter 2 introduces a robot platform with sensors that are used for this research. The literature review of laser range sensor methodologies, filtering applications, and JAUS is also presented in chapter 2.

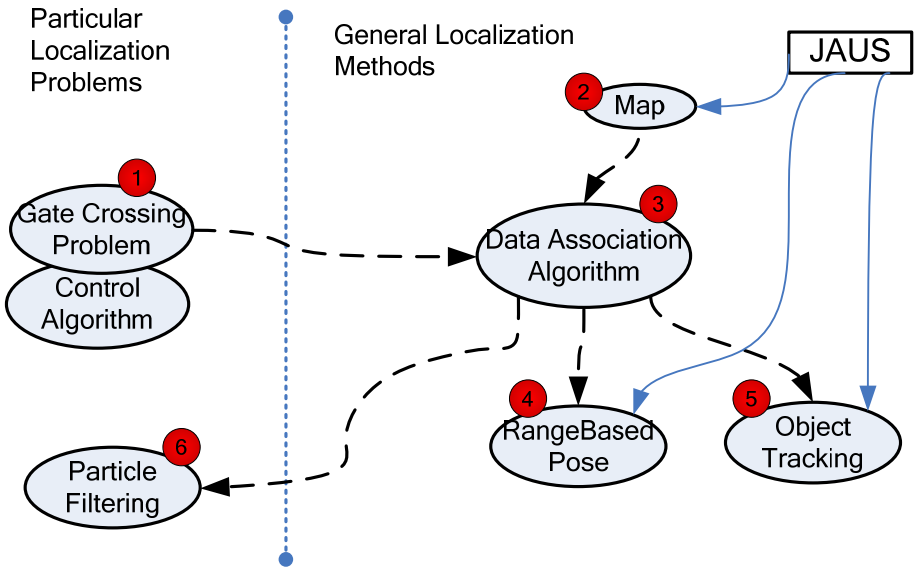


Figure 1.2 Structure of thesis work

The structure of the thesis is illustrated in Figure1.2, and is discussed in chapter 3 to chapter 7.

In chapter 3, gate crossing as one of localization and navigation problems was addressed. For outdoor mobile robots, GPS is a popular method for localization. However, the uncertainty and temporary lack of GPS signals near urban infrastructure always are the matter of the utmost concern for the outdoor mobile systems. Approaching the gate and navigating through it is a challenging problem that often requires the localization and execution accuracy going beyond the accuracy of basic localization capabilities provided by GPS and odometry. To address this problem, the concept of signature which represents the geometric feature of a specific object, an object recognition method using the signature concept and a control algorithm have been proposed. The experimental results are shown. The advantage, disadvantages, and the possible future improvements are discussed.

The gate crossing problem has its particularity and the developed algorithm is dependent on the pre-defined shape of the objects constituting a ‘gate’. To generalize the proposed algorithm to wider class of objects, a data association method has been developed. Instead of signature-based approach, the data association method using a prior map which represents the involved objects in the environment with polygons has been developed.

In chapter 4, using the definition of JAUS structure, a user-defined JAUS component, Map Data Server is developed. It is a geographic information sharing module. It can be incorporated into the mobile robot system as a JAUS component. The Map Data Server can store and supply other components with the geographic information (map data) which is essential for localization procedure. This component has been designed and integrated into the navigation system in 2005. Although the results of this chapter are subordinate, they are important and are extensively used later in the thesis.

In chapter 5, a data association algorithm is developed to find the error of the pose estimation from GPS/INS/Odometry. The algorithm is based on the match of two data sets, one is the real scanning data from the laser sensor, and another is the simulated scanning data from the prior map using the estimated robot’s pose. The results from simulation and experiments show the improvements of localization by using the proposed algorithm as compared to basic (GPS plus odometry) localization accuracy.

Chapter 6 discusses the implementation of the data association algorithm for two popular problems that appear in real applications. Two corresponding user defined JAUS components are developed. The Range Based Pose component is responsible for finding the error of the pose estimation by identifying the near object to the one found in the map. It corrects the robot pose using data association results. It also can find the objects which

do not belong to the prior map and report such an event to the high level command component. The Object Tracking component is responsible for localizing, monitoring and tracking the special object (usually not present in the map) chosen by the higher level component. These two components which apply the developed data association algorithm have been integrated into the robot system in 2007.

The work discussed in chapter 7 handles a special situation occurred in outdoor applications, in which the multiple pose possibilities are resulted from the repetitive characteristics of the environment. An algorithm based on particle filtering technique is developed to improve the localization quality under this special situation. This work is successfully tested by simulation in 2007.

Chapter 8 presents the conclusions and suggested future directions for the completion and extension of thesis work.

The contributions of thesis work are:

- Development of a particular algorithm for gate navigation procedure using the definition of signatures for gate objects. The algorithm provides the solutions for gate recognition and the localization method for navigation through the gate. A control algorithm capable of driving the robot through the gate based on the obtained localization results has been developed.
- Development of a user-defined JAUS component, the Map Data Server. The Map Data Server is JAUS-compliant, given the current JAUS standard.
- Development and implementation of a general and reliable data association algorithm. Through the generalization of gates crossing algorithm and the signature concept, a simulated map scanner is built to create a reference data set which is used

to improve the accuracy of the localization procedure when mobile robots run in the outdoor environment.

- Further elaboration of the achieved data association algorithm and development of two localization improvement and obstacle detection/avoidance-related user-defined JAUS components, Range Based Pose and Object Tracking.
    - Range Base Pose component is used to improve the localization capability, alarm the potential obstacles and detect the non-map objects.
    - Object Tracking component is used to locate, monitor and track the special objects assigned by high level commanding component.

  Both components apply the developed data association algorithm and have been used in the real application.
- Presentation of a proposed particle filtering algorithm and its simulation. Unlike the other particle filtering algorithms, this proposed algorithm applies the developed range data association method to estimate the particle weights. The particle filtering method may be used to catch the multiple pose probabilities in some environments where the layout of the objects has uniform geometric characteristics.

The publication related to thesis work is listed below:

- Sasiadek, J., Lu, Y., Polotski, V., Navigation of autonomous mobile robot with gate recognition and crossing, 8th International IFAC Conference on Robot Control (SYROCO 06), Bologna, Italy, September 2006.
- Polotski, V., Sasiadek, J., Lu, Y., Gate recognition and crossing by an autonomous security robot, 37th International Symp. on Robotics (ISR 2006), Munich, Germany, May 2006.

- Sasiadek, J., Lu, Y., Polotski, V., Navigation of autonomous mobile robots – Robot Motion and Control 2007, 187-208, Lecture Notes in Control and Inform Science 360, Springer, London, 2007.
- Lu, Y., Polotski, V., Sasiadek, J., Outdoor Mobile Robot Localization with 2-D Laser Range Sensor, Proceedings of the Tenth IASTED International Conference, 622-803, May 26-28, Quebec City, Quebec, Canada, 2008.
- Lu, Y., Polotski, V., Sasiadek, J., Particle Filtering with Range Data Association for Mobile Robot Localization in Environments with Repetitive Geometry, Robot Motion and Control, 2009.

Chapter 2
Background

2.1 Outdoor Mobile Robot Platform

Unlike the indoor mobile robots, the outdoor mobile robots usually carry out patrol, surveillance, explore and transport works in rough terrains. For pose estimation, the wheels of the robot platform must keep good traction and contact with the ground. For controlling of obstacle avoidance, the robot platform must have accurate maneuverability which is related to its dynamic/kinematic model. [8]. All of these factors have to be considered for the selection of the platform.

The platform used in this project is the ARGO conquest 6x6, shown in Figure 2.1. The ARGO platform is manufactured by Ontario Drive & Gear Ltd. (ODG). It is an all-weather, all-terrain outdoor vehicle with maximum land speed of 22 mph / 35 km per hour.

ARGO is equipped with a four cycle overhead valve V-Twin gasoline engine, a belt-driven, continuously variable transmission (CVT) with high, low, neutral and reverse bands. The specification of ARGO Conquest 6x6 is listed in Appendix B. Figure 2.2 shows the dimensions of the ARGO.

Figure 2. 1 ARGO conquest 6x6 (published by ODG)

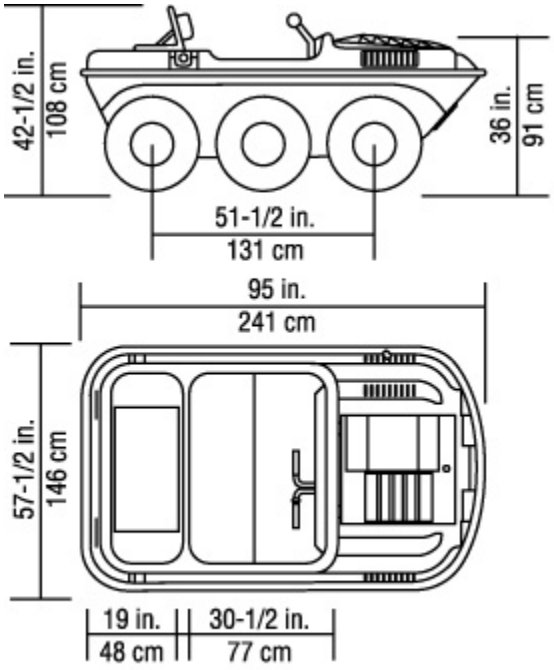


Figure 2. 2 Dimension of ARGO (published by ODG)

2.2 The Sensors Used on Mobile Robot

2.2.1 Sensors used for elementary localization

The robot platform used for this research equipped Odometry, INS and GPS to execute the elemental localization. The details of the principles of these sensors and the algorithms of implementation are beyond this thesis.

The inertial navigation system used in this application is a MEMS sensor, 3DM-GX1 from MicroStrain©. 3DM-GX1 combines three angular rate gyros with three orthogonal DC accelerometers, three orthogonal magnetic compasses, and an embedded microcontroller.

The equipped OEM4 GPS receiver and antenna are from NovAtel Inc. which has the positioning accuracy up to centimeter-level for ideal operation conditions.

The elementary localization is realized by the fusion of multiple sensor data. GPS will supply the position information continually when the GPS signals are available. The INS will provide the velocity (linear and angular) and heading information. The odometry measures the displacement and rotation of the robot. A sensor data processing module using Kalman Filter algorithm fuses the available sensor data and provides the optimal pose and velocity information.

2.2.2 Laser Range Sensor

The laser range finder or the Laser Measurement Sensor (LMS) is one of the laser-based Time-of-flight (TOF) systems. The laser-based TOF systems are based on “measuring the

TOF of a pulse of emitted energy traveling to a reflecting object, then echoing back to a receiver."[9].

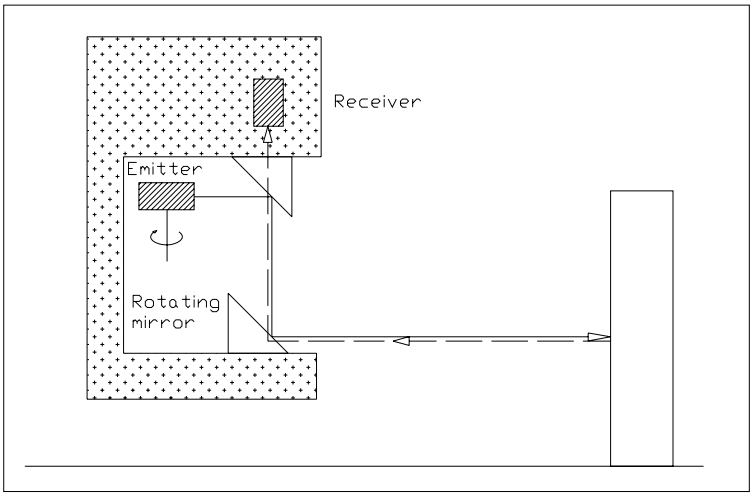


Figure 2. 3 Working principle of laser range sensor [10]

The accuracy of the laser range finder is determined by the frequency of the laser pulse and the receiver. For application, the accuracy is also related to the distance between the sensor and the target.

As depicted in Figure 2.3, a laser pulse emitted for a short time is reflected by a target object. A count starts while the pulse is transmitted, and stops when the reflected signal is received. The emitted pulse is diverted by an inside rotating mirror.

The laser range sensor adopted here is SICK LMS221 from SICK Ltd. Germany. Some important specifications/configurations are listed below:

The angular resolution of the scanner was set to 0.5°. The maximum range is $80\,m$, although it has been limited to $30\,m$ in order to get reliable and meaningful data. Therefore, a full 180° scan provides 361 range values (indexed according to a scanning angle). The data is transferred to the controlling computer by RS422 interface (500

Kbaud) at which rate the data transfer of a full scan occurs within 13.3 ms. According to the manufacturer's specifications, the scanner's error is up to $\pm 1\,cm$ within the maximum range of 80 m.

2.3 Kalman Filter

The Kalman Filter (KF) has become a common tool used in localization and mapping for mobile robots [11, 12]. The Kalman Filter method is used to estimate the robot's pose or the sensor data collected by the robots. The KF enables the simultaneous localization and mapping (SLAM). As a linear, recursive estimator, the KF produces an optimal estimation under the assumption of white, Gaussian noise processes. There are two important steps in KF, predict and update. During prediction step, the vehicle's pose uncertainty increases according to the propagation noise. In the update step, the KF reduces the uncertainty of the state variables and minimizes the mean squared error using the new measurement from the environment. The nonlinear version of the KF is the Extended Kalman Filter (EKF) in which the measurement and the transition models are not linear functions.

For discrete time, the evolution of the state vector X is defined as:

$$X(k) = f(X(k-1), u(k-1), w(k-1)) \tag{2.1}$$

where f is a nonlinear function of the state X, u is the control input and w is the process noise with zero-mean.

A measurement Z at time $t = k$ is defined as:

$$Z(k) = h(X(k), v(k)) \tag{2.2}$$

where h is a nonlinear measurement function of the state X, and v is the measurement noise.

So, the predict step may be described using:

$$\hat{X}^{-}(k \mid k-1) = f_k(\hat{X}(k-1), u(k-1), 0) \tag{2.3}$$

$$P^{-}(k \mid k-1) = A(k)P(k-1)A^{T}(k) + W(k)Q(k-1)W^{T}(k) \tag{2.4}$$

$\hat{X}^{-}(k \mid k-1)$ is the priori estimate of the state X at step $t = k$, given the inputs up to step $k-1$. $\hat{X}(k-1)$ is the posteriori state estimate at time step $k-1$. $P(k-1)$ represents the posteriori estimate error covariance at time step $k-1$. If considering zero process noise, equation 2.3 is same as equation 2.1. $P^{-}(k \mid k-1)$ is the priori estimate error covariance matrix at time k. $A(k)$ and $W(k)$ are Jacobian matrices of partial derivatives of f with respect to X and w.

The update step may be described as following:

$$K(k) = P^{-}(k \mid k-1)H^{T}(k)(H(k)P^{-}(k \mid k-1)H^{T}(k) + V(k)R(k)V^{T}(k))^{-1} \tag{2.5}$$

$$\hat{X}(k) = \hat{X}^{-}(k \mid k-1) + K(k)(Z(k) - h(\hat{X}^{-}(k \mid k-1), 0)) \tag{2.6}$$

$$P(k) = (I - K(k)H(k))P^{-}(k \mid k-1) \tag{2.7}$$

where $K(k)$ is the Kalman gain. H and V are the Jacobian matrices of partial derivatives of h respect to X and v calculated at each time step. I is the identity matrix. The detailed derivation may be found in [13].

2.4 Particle Filter

The Bayesian method provides a general approach to the estimation of the state of the system. The estimation is represented using a probability distribution function (PDF) which is based on the available sensing data. If the problem is the linear-Gaussian estimation and the PDF remains Gaussian, the Kalman Filter provides an optimal solution. If it is the non-linear case, the Extended Kalman Filter (EKF) has been successfully applied. The intention of using particle filter in mobile robot localization is to track the multiple possible poses.

Particle filters are a variant of Bayes filters which represent the state vector X by a set $S(k)$ of n weighted samples distributed around X.

$$S(k) = \{(X^i(k), w^i(k)), i = 1, ..., n\} \tag{2.8}$$

where $X^i(t)$ is the i^{th} sample at time k, and $w^i(t)$ is the corresponding weight, which sum up to one. Bayes filters estimate the state vector recursively – the initial distribution at time $k = 0$ characterizes the initial state of the system. In some cases such as the global localization where the initial state is not available, a uniform distribution over the state space will be used [22].

For the localization application, there are two procedures that affect the system state, sensor data update and motion data update. Let the motion information be denoted by m and the sensor data be denoted by z. According to the Bayes Filters, the posterior probability density can be estimated over the state space based on the data.

$$p(X(k) \mid z(k), m(k-1), z(k-1), m(k-2), ..., z(0)) = \gamma\, p(z(k) \mid X(k)) \int p(X(k) \mid X(k-1), m(k-1)) * p(X(k-1) \mid z(k-1), m(k-2), z(k-2), m(k-3), ..., z(0)) dX(k-1) \tag{2.9}$$

where γ is the normalization constant. $p(z(k) \mid X(k))$ and $p(X(k) \mid X(k-1), m(k-1))$ are probabilistic models referring to the sensor model and to the motion model.

Such models are under Gaussian assumption and can be derived from equation 2.1 and 2.2. If the functions f and h are linear, they are reduced to Gaussian distribution. If the functions are non-linear, their explicit computation remains complex.

The particle filter algorithm is recursive. The algorithm includes three phases: prediction, update, and re-sampling. In the prediction phase, after each operation, the particles are modified according to the model with the random noise. In the update phase, the particles' weights are re-calculated according to the latest sensor measurements. In the re-sampling phase, the particles with small weights are eliminated and the particles with big weights are duplicated. In chapter 7, the procedure of mobile robot localization using particle filtering technique will be demonstrated using an example from [85,86] and a particle filter algorithm using laser data matching method to handle the repetitive environmental features will be proposed.

2.5 Literature Review

2.5.1 Laser Range Sensor

The laser range sensor is usually used for obstacle detection, avoidance, localization and mapping. Some developed methods and algorithms are presented here.

Gowdy [15] builds an elevation map from laser range scanner images. The elevation map is quantified in a two dimensional array with a resolution of 10 *cm* per element which includes a value of height at that point. The data from different sensors are fused by comparing average elevation values from the map and the real world. A terrain pyramid is built to represent the terrain more efficiently. The bottom layer of the terrain pyramid is the elevation map. The other layers are the maximum and minimum elevation values over a group of four elements. The planner requests the range of a feature over a bounding box area and receives the approximate elevation.

Olin and Hughes [16] use a color video and laser scanner to develop planning software. The software uses digital maps to plan a preferred route, and then obtain the scene descriptions by classifying the terrain and obstacles while the vehicle traverses the route. With minimum delay, the vehicle may detect and avoid the obstacles in cross-country terrain.

Nashashibi [17] builds a rough geometric model of a 3D indoor terrain using a laser range sensor. The terrain model has two grid-based representations: elevation map and navigation map. The elevation map is built directly from range sensor data. The navigation map is a symbolic representation of the terrain built from the elevation map.

According to the capability of the vehicle's motion, the terrain is partitioned into homogeneous regions corresponding to different classes of terrain difficulties.

Kweon and Kanade [18] use multiple laser sensors incrementally to build an accurate 3D representation of rugged terrain. The sensor data are exploited by the locus method and then used to build the terrain map.

Arakawa and Krotkov [19] use fractal geometry to represent, analyze, and model the rugged natural shapes. Fractal Brownian functions are applied to estimate the fractal dimensions.

Stuck et al [20] develop a navigation system which includes three capabilities: obstacles avoidance, mapping, and path planning. The system also updates the histogram grid map for path planning.

A navigation system developed by Lacroix et al [21] includes three different navigation modes. When performing autonomous navigation, the system chooses a mode according to the nature of the terrains. A 2D mode is for a flat terrain, a 3D mode is used for an uneven terrain, and a reflex mode is used for an unknown terrain.

Castellanos and Tardos [22] present a segmentation technique of laser range data. A constraint-based matching scheme uses the segmentation method and a prior map to locate the vehicle in working space. The pose of the mobile robot is calculated by matching observed segments, obtained by a range sensor.

The work of Hancock et al [23] is related to the autonomous vehicle for a highway environment. They use the laser intensity information to detect obstacles at long range. At long distance, the horizontal surface would return weak signals and the vertical surface would give strong returns.

Mázl and Přeučil [24] combine odometry and a laser range sensor to generate a 2D polygonal approximation of an indoor environment. Segmentation methods are used to convert the points from laser range sensor to polygon edge. The calculated segments from the laser sensor then are merged with the prior environmental map.

Cox et al [25] develops an algorithm for laser sensor data matching. It assumes that both translation and rotation from the two sets are small. This matching method does not require a feature extraction stage. The reference set uses the prior map segments and the current set uses directly the range data points from the laser sensor. The restriction of the Cox method is the small error assumption and the convergence is very slow in some case. Also, the covariance matrix used has a high computational cost.

Bengtsson and Baerveldt [26] create an iterative algorithm, called Iterative Dual Correspondence (IDC), which matches two scans in a point-to-point manner. IDC takes the scan points, one by one, from the reference scan and tries to find a corresponding point from the current scan. Line segments are created to connect each pair of corresponding points. The length of each line segment is an estimate of how good the pair of points corresponds. Considering the existence of noisy points, IDC removes the points from the current scanning which have a longer segment than a certain threshold. Just the remaining points are used for matching.

Xiang and Liu [27] use complete line segment (CLS) method to match both reference and current data. The algorithm procedure includes line segment extraction, complete line segment finding, complete line segment matching, and an estimate of matching results. To find the best matching pair, the estimate step in CLS has to check all the possible matching pairs to get the best pair.

Weber et al [28] develops a matching algorithm - Anchor Point Relation (APR). The algorithm matches the current laser scan to the reference scan. The algorithm's output is a certain number of weighted hypotheses. The basic idea of the APR algorithm is the matching of two sets of characteristic 2D coordinates which is a reproducible object feature position (anchor points). There are three types of anchor points: jump edge, angle, and virtual edge. APR selects the best match by fully connected graphs constructed from their sets of anchor points. Then using the pair of matched sets, an alignment step finds a coordinate transformation that aligns one anchor point graph with another graph.

Gutmann and Schlegel [29] present a combined algorithm - Combined Scan Matcher (CSM). It combines the Cox [25] matching approach and the IDC [26] matching method. There are two matching algorithms in CSM. One is points-to-segments assignment from a modified Cox method. CSM extracts line segments from the reference scan and uses them as a prior model. The second is points-to-points assignment from the IDC method.

Jensfelt and Christensen [30] use the Kalman Filter method to estimate the robot's pose by the data from laser range sensor. The scan data from the laser sensor is processed by data filtering and creates a series of segments. Then an estimated pose is calculated.

2.5.2 Particle Filter

Many difficult problems in the robotics field have been solved using the particle filtering technique [31]. Some of the achieved applications and algorithms are listed here.

In [31], Thrun reviews the recent innovations of particle filter applications, and provides some guidances on how to use particle filters in robotics field.

Adams et al [32] present a new method of natural feature extraction of laser scan data. This method is used for the purpose of mobile robot localization in outdoor environments. A batch process carries out the feature extraction from the laser scanning data. The algorithm includes two steps: data segmentation and parameter acquisition. A modified Gauss-Newton method is adopted to fit the circle parameters iteratively. The experiments show that the proposed new method is more robust than the existing methods.

Karlsson and Gustafsson [33] analyze and improve the performance of particle filter using the Cram´er Rao lower bound for underwater robot system. The original Cram´er Rao lower bound rules are simplified by using map and high quality sensors. The work proposed a more realistic five state model which uses Rao-Blackwellization to decrease computational complexity.

Fox [34] describes a variation of Monte Carlo Localization (MCL) method. This method uses a mixture distribution to facilitates recovery from localization failure fast. The mixture distribution has been proved having the advantages over the normal ones used in standard MCL. An implementation of MCL to cooperative multi-robot localization of robots is also presented.

Gustafsson et al [35] develop a framework for positioning, navigation, and tracking problems using particle filters. The framework includes a class of motion models and a general non-linear position measurement equation. A general algorithm with a small number of particles is presented. Using this algorithm, a Kalman Filter estimates all position derivatives and a particle filter keeps low-dimensional. It also describes a map matching method. Using the matching method, an aircraft's elevation profile is matched

to a digital elevation map and a horizontal driven path is matched to a street map. The results from experiments show that the developed method has higher accuracy and integrity than the satellite navigation (as GPS).

Kim and Iltis [36] use a novel Mean-Value Theorem Particle Filter (MVT-PF) to build a highly nonlinear measurement model from the GPS C/A code. The performance of the particle filter and the one from a conventional Extended Kalman Filter are compared. The results from simulation demonstrate that the MVT-PF surpasses the conventional Extended Kalman Filter.

Milstein et al [37] develop an extended Monte Carlo Localization (MCL) algorithm. The algorithm can handle a special localization problem in a highly symmetrical environment, in which the conventional MCL is unable to track the multiple possible poses of the robot. Two new ideas are used in this algorithm: clustering of samples and modifying the proposal distribution based on the probability mass of those clusters. The presented experimental results show that this new extended MCL algorithm successfully solves the special localization problem from the symmetric environmental features.

Grisetti et al [38] use a particle filter to solve the simultaneous localization and mapping (SLAM) problem. Unlike the normal applications, each particle in this particle filter carries an individual environmental map. For grid map learning, the number of particles in the Rao-Blackwellized particle filter is reduced by a developed adaptive technique. Both the movement of the robot and the recent observation are considered to compute an accurate proposal distribution. The uncertainty about the robot's pose in the prediction step of the filter is drastically decreased. To minimize the particle depletion problem, a selectively-carry-out method is used in the re-sampling step. The developed algorithm is

tested by experiments in large-scale indoor and outdoor environments. The results show that the method gains the advantages over previous approaches.

2.5.3 JAUS

The Joint Architecture for Unmanned Systems (JAUS) is a unified message set. JAUS is designed to reduce development time and improve interoperability between unmanned systems. As a new application hot point, JAUS has been implemented on some ground vehicles and robotic systems.

The goal of the Joint Architecture for Unmanned Systems (JAUS) is to reduce life cycle costs, enabling fast integration of new technologies, and facilitating interoperability among the unmanned systems. In [39] the author defines the scope of the JAUS, the rationale for the scope and the definition of the requirements. In [40] the author defines the Reference Architecture (RA). “RA is the technical specification that the developer shall use to implement JAUS on the unmanned system. It is also the document that will be used by the acquirer to assess technical compliance.” In [41], the JAUS specific protocol for transmission of JAUS messages is specified. Also [41] presents the JAUS standard data types, data models, the JAUS message header and the types of JAUS messages. In [42] the JAUS message set was specified.

Evans III in his master thesis [43] investigates geospatial data modeling methods used within the unmanned systems and geodetic information systems (GIS) communities. He presents observations about current methods and recommendations for a common JAUS world modeling message set. This message set is a standard for communicating real-time and prior geospatial data between unmanned systems. His work has been submitted as the

first draft for the first generation of the world modeling knowledge storage component [44] within the JAUS.

In [45], C. Crane describes the development of an autonomous vehicle system. This system participated in the DARPA Grand Challenge in 2005. The architecture is based on version 3.0 of the JAUS. The "smart sensor" concept is introduced. Using this concept, the different sensors are given a standardized means and the processing of data presentation and integration from the different sensors are unified.

R. Touchton, et al [46] build two user-defined JAUS components: off-line and reactive path planning and world modeling. The design of these components considers four characteristics: vehicle platform independence, mission isolation, hardware independence and technology independence which are also the requirements of the JAUS systems.

J. Steven, et al [47] outline a method called vision based correction. The prior information and perceived road characteristics are considered during the navigation and localization procedure. The vehicle system is based on the JAUS.

G. Gothing et al [48] discuss the implementation of JAUS on a fully autonomous 2004 Cadillac SRX. A Potential Field Method is used for navigation. The overall system architecture is designed using the JAUS protocol. The test data and experiment results are also presented. The work demonstrates that the potential field navigation method is compatible with JAUS interoperable control system.

Chapter 3

Gate Crossing Using 2-D Laser Range Sensor

3.1 Introduction and Problem Description

Security robot patrol task often include two different phases: during the first phase a robot navigates in essentially open area with landmarks and weak limitations over its position accuracy. In the second phase it navigates near pronounced infrastructures with much tighter requirements to its position accuracy. Transition between those phases is a challenging problem. The problem of gate recognition and crossing addressed in this chapter is an example of such a problem.

During the patrol task of the mobile security robot, GPS provides information for global level navigation but also are often used for navigation at local level by some necessary coordinate systems exchanging. However, there are some drawbacks of using a GPS sensor. They could be:

1. Periodic signal blockage due to obstruction;

2. No available satellites in some areas;

3. Special tasks, such as the navigation through the gate, in which case an error in GPS location signal may be too large and have serious consequences.

These problems are particularly important if differential GPS (DGPS) cannot be used, either because it is not available in the area or it is not desired due to logistic requirements.

Specially, the problem concerned in this chapter consists of guiding a vehicle through the area delimited by two objects of known shape called posts located at the known distance from each other (entrance width). The geographical location of the whole structure (called gate) is supposed to be only roughly known. This means that the longitude/latitude of the entrance center point can be transformed to its position on the area map with few meters accuracy, but the geometry of the gate can be known with few centimeters accuracy.

The typical task to be performed consists of navigating along the path that leads to the gate area and passing through the gate. Typical entrance width is $6\,m$, typical size of the post is about $1\,m$. Experimental vehicle used for a proof of concept stage is about $2\,m$ wide and $3.5\,m$ long.

A few meters precision of basic localization using GPS is not sufficient for guiding through the gate or narrow passage between two buildings with similar typical width ($4\,m \sim 6\,m$) and required precision is about $0.3\,m \sim 0.5\,m$. Considering the above reasons, combining GPS with other navigation technologies would be very effective.

The chapter is organized as follows: in section 3.2, a brief literature review for the related research is presented; in section 3.3, the model of the environment is constructed and the notion of the gate signature is introduced; in section 3.4, gate recognition, filtering procedure, and localization method are described; section 3.5 is devoted to the design of

motion control modules. Description of obtained results and discussion of future work is given in section 3.6 and some conclusions in section 3.7.

3.2 Review

In practice, the integration of multi-type sensors was found to produce very reliable autonomous operations in proven mobile robot navigation systems [48, 50, 51, 52].

Many works have been done for door detection and/or crossing in indoor environment. Most of these are portion of the indoor navigation or pose tracking problem.

In [53], the concerned indoor pose tracking problem is solved by using a minimalistic environment model and a realistic laser scanner model. The large scale structure like the wall and door in a room are regarded as important features for the minimalistic environment model. The feature tracking (wall following, door detection) simplified the computation of localization procedure. An Extended Kalman Filter is used to extract the corresponding features from the laser scan. Several indoor experiments demonstrated the effectiveness of the algorithm.

In [54], the authors built a virtual sonar sensor to detect the door and locate the robot in the topological map. In this work, the door is regarded as a landmark. The successful recognition helps to detect the localization error. The AD control architecture is applied with the developed virtual sonar sensor.

In lots of achieved works, the door detection and/or crossing problem is addressed using vision system for indoor environment [55, 56, 57]. In [55], the detection is depended on

the color of the door which is different from the other objects. In [56], an opened or closed door is detected by sonar and infrared sensor. In [57], the features of the door image are extracted. The detection of the door depends on the match of the features. The status of the door is judged by the sonar sensor.

The gate crossing problem addressed here consists of detecting the entrance using a laser sensor and reactively navigating through this entrance. The gate crossing problem differs from range-based, wall-following problems [58, 59], since it requires the transition from open field, essentially GPS or sensor fusion-based navigation [52] to range-based only navigation. This transition itself requires a feature extraction and recognition phase usually not necessary in the environments such as underground mines or indoor operation, where range-based wall-following is sufficient [58, 59].

The extraction of robust geometrical features from the data provided by the LMS (Laser Measurement Sensor) is not an easy task. In [60, 61, 62, 63], some different methods are discussed to extract the geometric features, such as points clouds registration [61, 62], context recognition [60] and feature extraction from point clouds [63].

This proposed solution for guidance through the gate consists of several steps: environment representation, LMS data filtering, gate recognition, localization and motion control. Unlike the existing methods, a vector (signature) is used to represent the special gate-like object and the zones of viewing are defined to simplify the recognition algorithm. Moreover, a corresponding control algorithm is designed to navigate the robot crossing the gate based on the information from the recognition algorithm.

3.3 Zone Dividing and Signature Concept

Two wooden boxes (cubes of $1.20\,m$) have been used to construct the gate. The distance between the boxes – the gate entrance - is set to $6.0\,m$. The vertices are indexed for reference as shown in Figure 3.1 (the edge is referenced by delimiting vertices). Depending of the position of the sensor, the perception of the gate obtained using LMS scanner is different. To address the perception of the gate geometry by the sensor, the neighborhood of the gate is divided into proximity zones. Figure 3.2 shows the ten partitioned zones of the gate. In each zone some posts' edges are observable while the others are hidden. Figure 3.3 illustrates visible and hidden edges for the scanner located in zone V: the edges $1-2$, $2-3$, $5-6$, and $6-7$ are observable and the edges $3-4$, $4-1$, $7-8$, and $8-5$ are hidden.

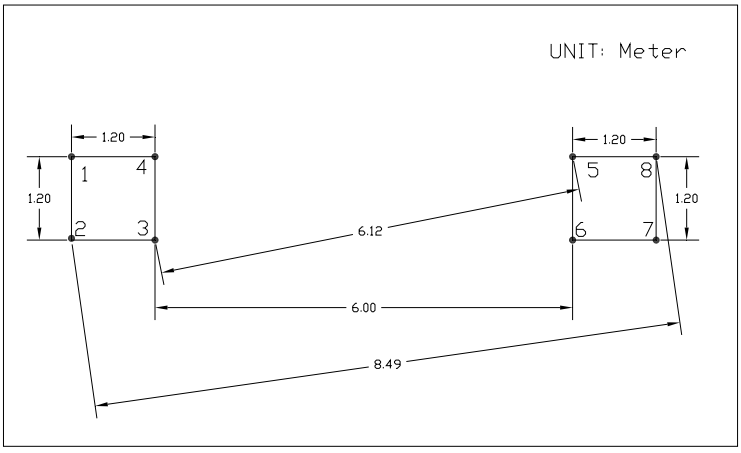


Figure 3. 1 Gate dimension and vertex points (all dimension are in meters) [5]

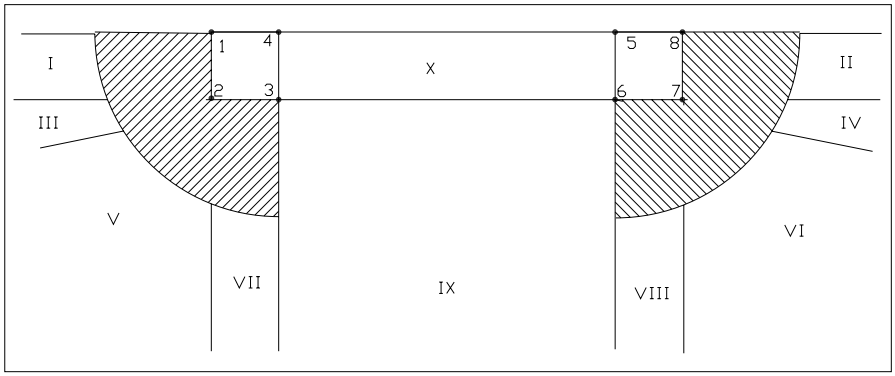


Figure 3. 2 Proximate gate zones for navigation [5]

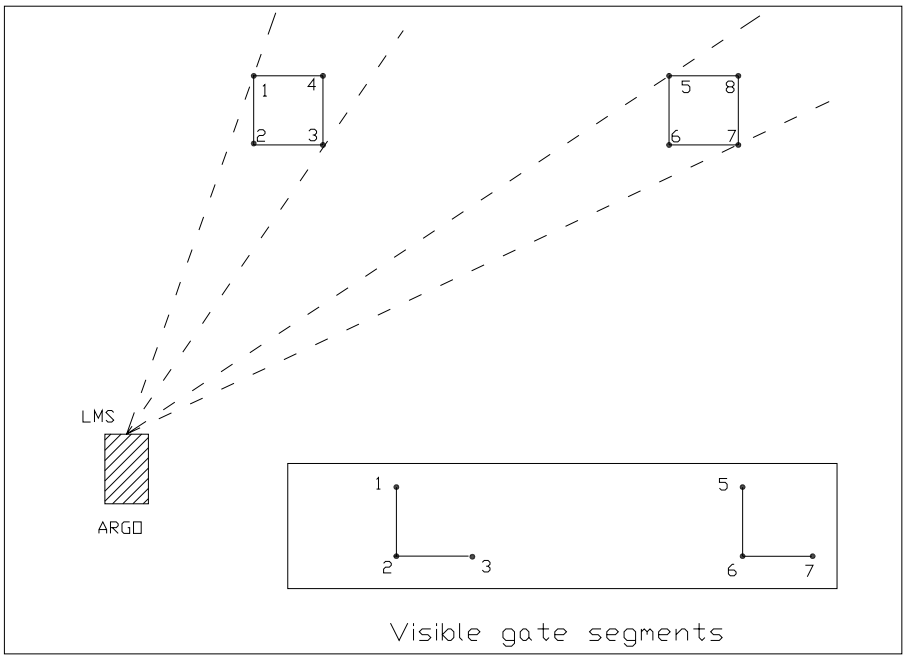


Figure 3. 3 Visibility of edges from LMS and the visible gate segments from zone V [5]

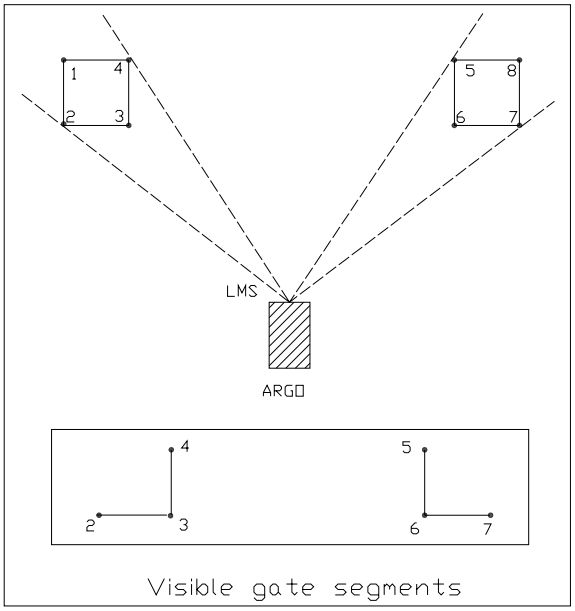


Figure 3. 4 Visibility of edges from LMS and the visible gate segments from zone IX [5]

The obtained scan (observable image of the gate) is shown in the inserted rectangle in Figure 3.3. As can be easily seen, each scan contains two portions (corresponding to two posts); each portion contains an isolated segment or two adjacent segments (a corner). Figure 3.4 is another scanning illustration from zone IX .

Scanning from zones ($I - IV$, X in Figure 3.2) gives poor representations of the gate as illustrated by icons in the table 3.1. For Zone I and II there is only one visible edge, in zone III and IV the reflection degrades since the part of the surface is illuminated at a very small angle. This control algorithm will try to avoid guiding through those zones $I - IV$, using zones $V - IX$ instead. Zone X also gives a poor representation of the gate, but this zone is inevitably visited by the vehicle after it has almost crossed the gate area. Recognition is not an issue by that time – the gate is almost crossed but the special procedure is needed to estimate the relative pose of the vehicle in this zone.

Table 3.1 Visible gate segments from the zones [5]

Zone	Gate Segments		Zone
I	⎸	⎸	II
III	⌞ ⌞	⌟ ⌟	IV
V	⌞ ⌞	⌟ ⌟	VI
VII	_ ⌞	⌟ _	VIII
IX	⌟ ⌞	⎸ ⎸	X

In order to compute the vehicle's location relative to the gate, the zones have first to be distinguished from which the scan is obtained. The concept of the "gate signature" has been proposed. The "gate signature" is a vector describing the geometry characters of a gate-like object. Signature vector is computed considering the length of each observed segment, the distance between two continuous portions of the scan (gap), and the distance between the scan start point and the scan end point (size). So, the information in the "gate signature" includes the visibility of the gate edges, the size of the gate object and the entrance of the gate object.

Two examples of signature are given in Figure 3.5: in the upper rectangle, there are four segments, ab, bc, ef, and fg; ce is a gap between two posts and ag is the total width of the scan. A vector $[ab, bc, -ce, ef, fg, ag]$ represents so called signature. The negative sign of the third component emphasizes the fact that it corresponds to the gap and not to the edge. Similarly, for the low rectangle, the signature would be $[bc, cd, -de, ef, fg, ag]$.

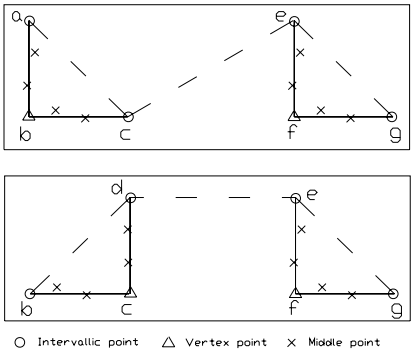
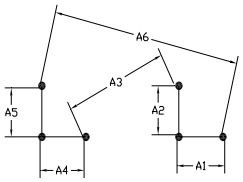
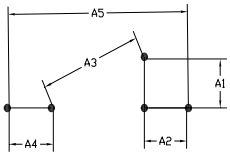
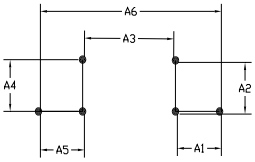


Figure 3. 5 Signature illustrations of gate segment [5]

Table 3.2 shows the possible canonical signature vectors for the scans obtained from the zones *V* , *VII* , and *IX* . One can see that even the dimension of the signature varies from zone to zone.

TABLE 3.2 Signatures of the gate scans [5]

Zones	Image	Signature	Dimension
V		[A1,A2,-A3,A4,A5,A6]	1x6
VII		[A1,A2,-A3,A4,A5]	1x5
IX		[A1,A2,-A3,A4,A5,A6]	1x6

The canonical signatures are computed for every zone. When executing the navigation the gate signature is computed on-line using collected LMS data (scans). By comparing this on-line signature against the set of canonical zone signatures, the robot location with

respect to the gate (corresponding zone) may be roughly determined and then be used to proceed the gate recognition and finally to the refinement of the localization result and guidance.

3.4 Gate Recognition Procedure

An algorithm of gate recognition is described in this section. The algorithm contains several modules: the GPS/INS/LMS data collection module, the map building module, the map filtering module, the map fitting module, the gate signature calculation module and the unexpected error check module. The flowchart of the algorithm is shown in Figure 3.6.

Data is acquired from GPS/INS/LMS firstly with GPS/INS/LMS data collection module. The map-building module computes the map of the current environment using the row data from LMS. The preliminary map includes all the visible objects within the range of the laser scanner. Some of these objects are false positives, they do not correspond to real objects but to the noisy LMS measurements (physically attributed to multi-path reflections, dust etc.). An example of the preliminary map from the raw data is given in Figure 3.7. The encircled object is a gate.

Generally, the preliminary map cannot be used directly because of too many non-gate objects. All the noisy points and non-gate objects from the preliminary map would be removed by the map filtering module. It evaluates all the detected objects by their dimension and relative position and keeps only objects corresponding to the gate posts.

The approximations for post size and entrance width are used for this computation (50% difference in size and width is tolerated).

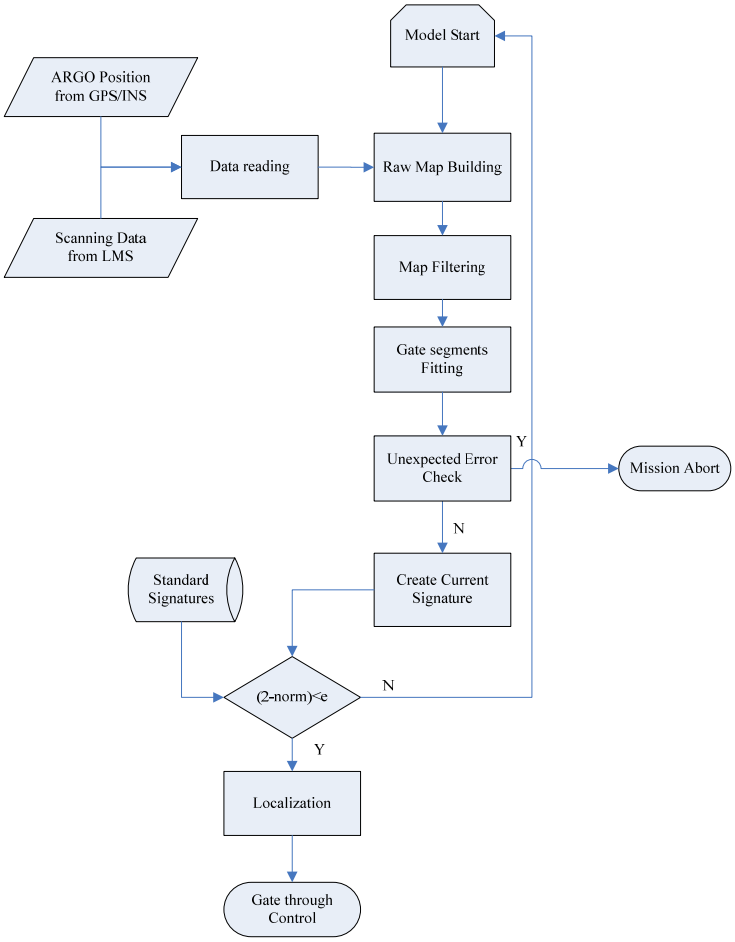


Figure 3. 6 Flowchart of gate navigation module [5]

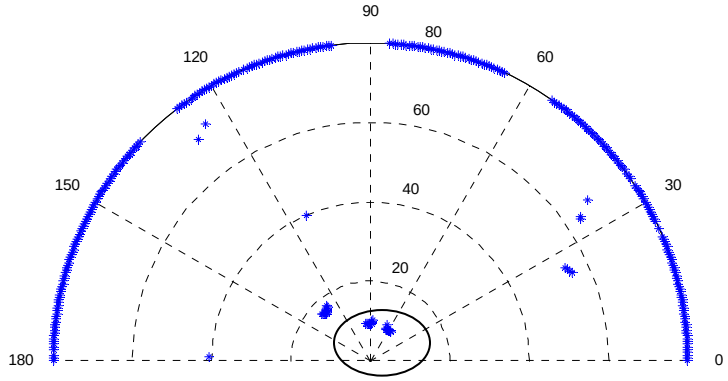


Figure 3. 7 Map of the raw scanning data [5]

The map-filtering algorithm is described in details as:

Step 1: Set the effective range to $\leq 30\,m$ (software limit) while the maximum range of LMS sensor is set to $80\,m$ (hardware limit). Thus, all points with ranges higher than $30\,m$ are deleted;

Step 2: Delete all the objects consisting of only one isolated point (usually the single point corresponding to noise);

Step 3: Determine the dimension of each object and delete those objects that, based on their dimensions obviously do not correspond to the gate (50% tolerance);

Step 4: Calculate the distance between two consecutive objects (scanning from left to right) and delete those which, based on their relative distance are clearly not the gate posts (50% tolerance).

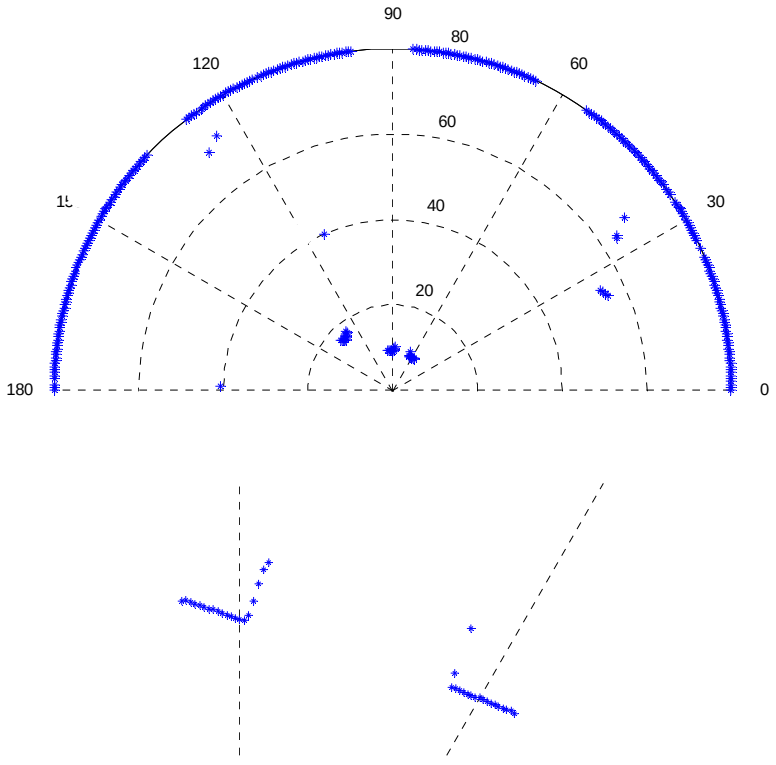


Figure 3. 8 Filtered map and the enlarged gate object [5]

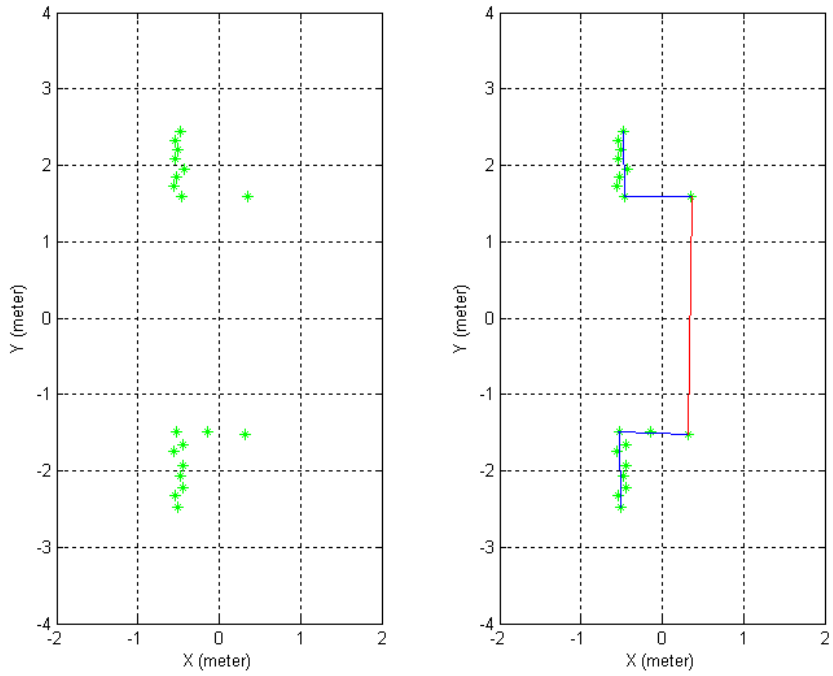
Figure 3.8 shows the filtered map corresponding to a preliminary map from the Figure 3.7.

Give : *scanning* : $P^0 = \{p_1, p_2, ... p_{361}\}$
gate size tolerance : $T_{gate} = 1.5$, *gap tolerance* : $T_{gap} = 1.5$;
gate size : D_{gate}, *gap size* : D_{gap};
for $i = 1:361$
{ *if* $Range(p_i) > 30$
{ *delete* p_i *from* P^0 } }
%P^0 changes to $\{p_1, p_2, ... p_n\}$, n<=361;
%P^0 includes N_0 clusters $\{A_1, A_2, ... A_{N_0}\}$;
for $i = 1 : N_0$
{ *if* (*number of points in* $A_i = 1$)
{ *delete* A_i; } }
%P^0 changes to $\{p_1, p_2, ... p_m\}$, m<=361;
%P^0 includes N_1 clusters $\{A_1, A_2, ... A_{N_1}\}$;
While($N_1 > 2$ & $T_{gate} > 1$ & $T_{gap} > 1$)
{ *for* $i = 1 : N_1$
{ *if* $(Dimension(A_i) > T_{gate} * D_{gate} \mid Dimension(A_i) < D_{gate})$
{ *delete* A_i; } }
%N_2 *clusters remained*;
$N_1 = N2$;
for $i = 1 : N_1$
{ *considering* A_i *and* A_{i+1};
if $(\text{Distance}(A_i \text{ to } A_{i+1}) > T_{gap} * D_{gap} \mid \text{Distance}(A_i \text{ to } A_{i+1}) < D_{gap})$
{ *delete* A_i; } }
%N_2 *clusters remained*;
$N_1 = N2$;
$T_{gate} = T_{gate} - \Delta T_{gate}$;
$T_{gap} = T_{gap} - \Delta T_{gap}$;
}
If $(N_1 = 2)$ *gate detected*;
else no gate in the sensing range.

Figure 3.9 Map filtering algorithm

After the above steps, if the filtered map still contains more than two objects, the tolerance conditions for the dimension and distance are tightened and the filtering algorithm is re-applied. The algorithm is shown by pseudo code in Figure 3.9.

The points obtained from the scan are discrete, but represent continuous edge lines. Each edge is delimited by vertex points and can be identified by those points. The vertex points are located on the border of the point cloud (have neighbors only on the right or only on the left hand side) and thus correspond to the corner of the object. The estimated gate image is obtained by connecting the vertex point one by one.



(a) (b)

Figure 3. 10 The image of gate [5]

The detail of the segment fitting algorithm may be described as the following steps:

Initial step: Determine which points belong to the gate. In the module, the effect from environmental disturbances is not considered and the assumption that the only object in the filtered map is the gate is used. In an example shown in Figure 3.10, the gate includes

all the points marked as shown in the lower rectangle of Figure 3.5 from point (b) to point (g).

Step 1: Scanning from left to right, find the last point of the left hand post, point (d), and the first point of the right hand post, point (e). This is done by comparing the distance between each two consecutive points and the canonical size of the post and the canonical size of the entrance;

Step 2: To find the "corners" (points (c) and (f) in Figure 3.5 notation), if they exist, the points which have the largest distance to the line connecting the first and the last points of each post ((b), (d) and ((e), (g)) are found. For the case illustrated in Figure 3.10, two corners for the two posts exist.

When all the vertexes/corners are identified, the current signature for the scanned gate is constructed as explained in section 3.

For example in Figure 3.5, the current signature is a vector of dimension 6 (full): $[bc, cd, -de, ef, fg, ag]$.

Comparing this current signature with the canonical signatures, the gate is recognized if a canonical signature matches the current signature. For matching decision, the conventional Euclidian norm of the difference between the current and canonical signature is applied.

Let the canonical signature be:

$$A_0 = [a_{01}, a_{02}, -a_{03}, a_{04}, a_{05}, a_{06}]$$

and the current signature be:

$$A = [a_1, a_2, -a_3, a_4, a_5, a_6]$$

The criterion norm of $A - A_0$ is:

$$e = [(a_1 - a_{01})^2 + (a_2 - a_{02})^2 + (-a_3 + a_{03})^2 + (a_4 - a_{04})^2 + (a_5 - a_{05})^2 + (a_6 - a_{06})^2]^{\frac{1}{2}}$$

For matching the value e is compared against the threshold $e_{\max}$. If, $e < e_{\max}$ two vectors are matched otherwise they are not.). The positive matching result ($e < e_{\max}$) means the gate is recognized. The next map fitting step consists of establishing correspondence between observed points and the gate segments (edges of the posts).

Computing relative position of the vehicle with respect to the gate is a crucial step for being capable of designing an appropriate controller for guiding the vehicle through the gate. The localization module outputs the data to the gate crossing control module. Here the final step of processing localization data is described. Defining a coordinate frame in this way, {*OXY*} attached to a vehicle with the original point *O* – being a point of the platform where LMS is located, *OY* – being a longitudinal axe of the vehicle aligned with 90° LMS beam. Figure 3.11 illustrates the layout. The gate recognition procedure is based on the extraction of points $\{1,2,3,4,5,6,7,8\}$. Using this information the middle point C of the entrance and the direction of the straight line containing points {2, 3, 6, 7} (from the left post to right post) can be estimated. More precisely, the direction of the beam pointing from the left to the right post is estimated – called entrance direction, P_{ENTER}. Some other geometrical features needed for the control module (described in the next subsection) are also extracted.

An angle ψ between P_{ENTER} and OX axe is the first parameter to be inputted to the controller being designed. A target point T initially coincides with C then is shifted

from C along the direction orthogonal to P_{ENTER} , called the gate direction, P_{GATE} . A distance ρ from point O to point T is another parameter to the control modules.

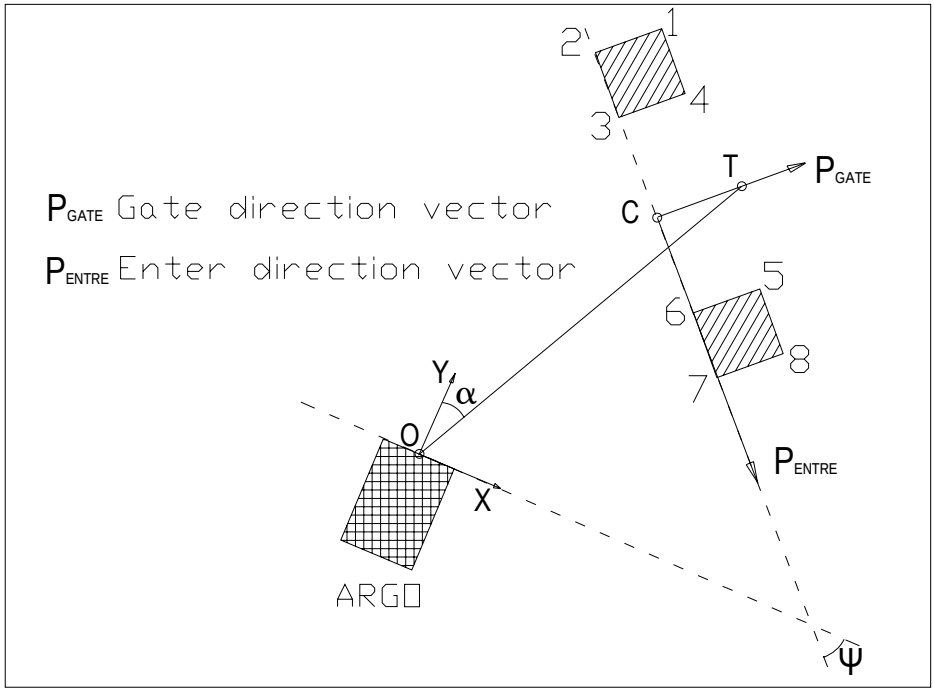


Figure 3. 11 The layout of localization and control method [5]

Another parameter is an angle, α , between OT and OY . As can be easily seen for the vehicle perfectly aligned with the gate, both angles α and ψ are zero.

3.5 Control System

In order to guide the vehicle through the gate, one may apply either planning/correction approach or nonlinear control approach. The latter has been chosen, exploiting the similarity of the problem of guiding a vehicle towards and through the gate with the parking problem addressed in [64]. In both cases, the vehicle is driven to a desired pose

(position/orientation). For the "gate crossing" problem the desired position is a middle point of the entrance segment and orientation – orthogonal to this segment. In contrast with original settings described in [64], the target point in this case is not known in advance but has to be continuously estimated on-line. This constitutes a remarkable difference between gating and parking problems. Another difference is that the speed of the vehicle is not constrained to zero at the target point, but rather set to the desired value (usually lower than the speed in the open space).

In order to address these differences, (1) the entrance middle point C needs to be estimated, (2) a target point T is defined as initially coinciding with C and then moving from the entrance middle point along the gate direction (see definition above). This procedure allows to keep the vehicle offset from the target point (singular point in [64]), thus ensuring a stable motion across the gate.

The kinematic model of the robot is defined as follows:

$$\begin{aligned} d\theta / dt &= \omega \\ dx / dt &= V \cos\theta \\ dy / dt &= V \sin\theta \end{aligned} \tag{3-1}$$

or in discrete setting:

$$\begin{aligned} \theta_k &= \theta_{k-1} + \omega_k \Delta t \\ x_k &= x_{k-1} + v_k \Delta t \cos\theta_k \\ y_k &= y_{k-1} + v_k \Delta t \sin\theta_k \end{aligned} \tag{3-2}$$

The main concern of gate crossing control is the steering control, although the linear speed V is also regulated. Control is designed in two steps. First, an algorithm called Astolfi-controller initially proposed in [64] for achieving a desired pose (parking problem)

is adapted for gate navigation. Second the robust version taking into account particular platform kinematics (skid-steering) is designed.

The control design is based on the information about the vehicle position relative to the gate extracted by the gate recognition model. As defined in the previous section OXY is the vehicle coordinate system (O - center of the vehicle, OY - oriented along the longitudinal axe of the vehicle), C is a middle point of the gate, P_{ENTER} is entrance direction, T is a target point shifted from C along the gate direction P_{GATE} (orthogonal to P_{ENTER}). Distance to the gate ρ and two angles ψ - between the gate entrance P_{ENTER} and OX axis, and α - between OT and OY have been defined above (see Figure 3.11) .

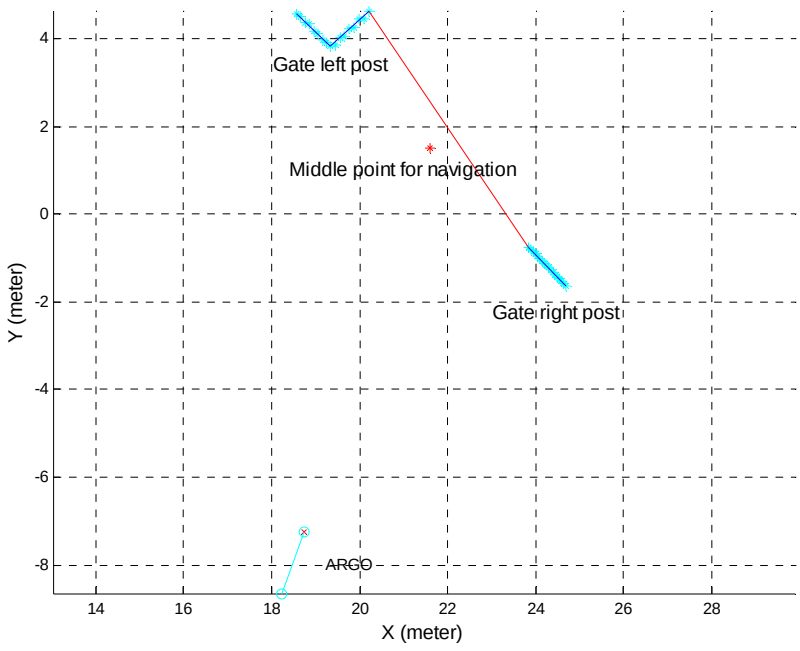


Figure 3. 12 Final gate image and the middle guide point [5]

Figure 3.12 shows the extraction of the middle point and entrance direction from the real LMS image.

In the original paper [64], it was shown that by changing the variable from (x, y, θ) to (ρ, α, ψ), the kinematical model (Equation 3-1) can be rewritten in the form of:

$$\begin{aligned} d\rho / dt &= -V \cos\alpha \\ d\alpha / dt &= V \sin\alpha / \rho - \omega \\ d\psi / dt &= -\omega \end{aligned} \tag{3-3}$$

A stabilizing controller for this system is then chosen in the following form:

$$\begin{aligned} V &= K_\rho \rho \\ \omega &= K_1 \alpha + K_2 \psi \end{aligned} \tag{3-4}$$

In linear settings, the coefficients K_ρ , K_1 , K_2 have to be chosen by pole placement technique for a matrix from the system equation (3-3) and (3-4):

$$\begin{bmatrix} -K_\rho & 0 & 0 \\ 0 & -K_1 + K_\rho & -K_2 \\ 0 & -K_1 & -K_2 \end{bmatrix} \tag{3-5}$$

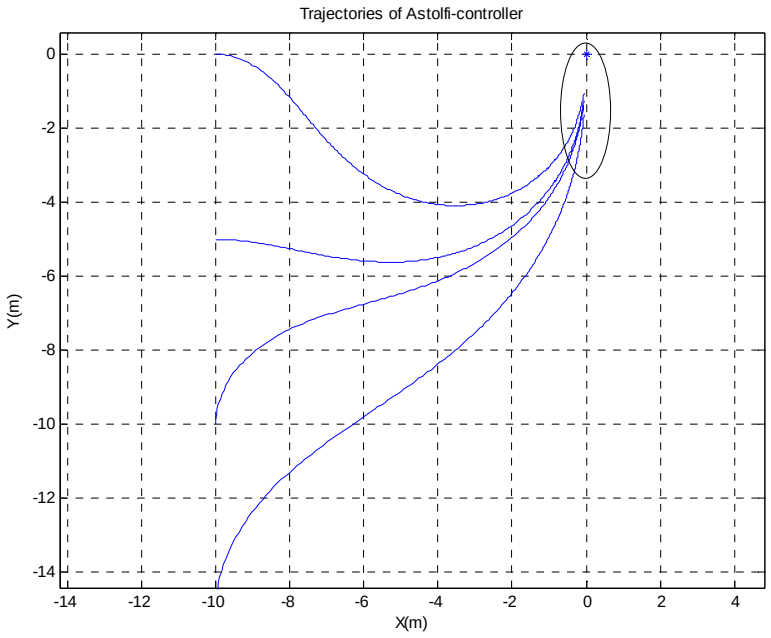


Figure 3.13 Trajectories of Astolfi-controller

In the initial coordinates (x, y, θ) described by equations (3.1), the control low is:

$$V = K_\rho \sqrt{x^2 + y^2}$$
$$\omega = K_1(\arctan(\frac{y}{x}) - \theta) + K_2(\frac{\pi}{2} - \theta) \tag{3.6}$$

Global exponential stability in nonlinear settings (convergence to $[0,0,0]$) is also proven in [64]. Figure 3.13 illustrates the global convergence of the trajectories obtained from Astolfi-controller.

For the gate navigation, the most interest is in the aligning property observed along the last portion of the trajectory (inside of the ellipse). This property is of utmost importance for navigating through the gate.

Inspired by the results presented in [64], the control algorithm for gate navigation is developed by extending the controller described in [64]. The whole motion is divided in three phases. First phase is a path following phase with data collection and analysis targeting the potential gate recognition. "Gating controller" is not activated in this phase. When the gate is recognized, the data collection is continued until the distance to the gate is decreased to the first threshold ρ_1. From here, the second phase starts and "gating controller" in the original form proposed in [64] is activated and kept until the second threshold ρ_2 ($\rho_2 < \rho_1$) is reached. Parameters of the controller have been experimentally chosen in the following ranges: $\rho_1 = 20(m)$, $K_\rho = 0.1\,(s^{-1})$, $K_1 = 0.9(\mathrm{s}^{-1})$, $K_2 = -0.4(\mathrm{s}^{-1})$. Such K_ρ, K_1, K_2 result in the negative eigenvalues at -0.1, -0.2, -0.2. Exponential stability demonstrated in [64] ensures that ρ is decaying and therefore eventually reaches ρ_2. From here, the third phase starts. "Gating controller" with a moving target point is activated. The following paragraphs describe its design in more detail.

Performing geometrical analysis similar to these given in [64] and leading to Equation (3.4), a moving target point T is defined as staring from the point C at the moment when $\rho = \rho_2$ and moving along the direction P_{GATE} with speed equal to the projection of the vehicle speed to P_{GATE} direction. The speed of the vehicle relative to this moving target becomes V_{ENTER}, which is parallel to P_{ENTER}, since the projected component on P_{GATE} is canceled out by the motion of the target T.

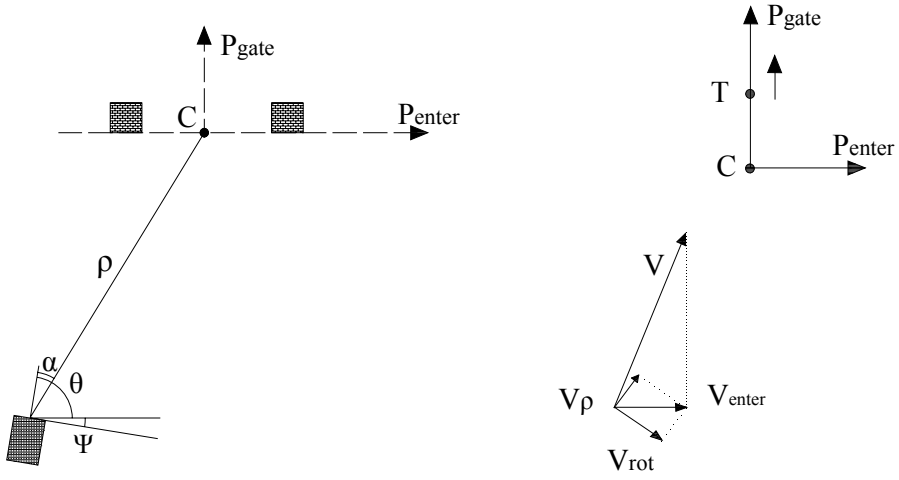


Figure 3.14 Angles definition and velocity projections

For this relative velocity V_{ENTER}, its projection onto direction to the target is V_ρ and its projection onto the normal (to ρ) direction is V_{rot} (as shown in the right of Figure 3.14). These velocity components can be calculated as:

$$\begin{aligned} V_{ENTER} &= V\cos\theta \\ V_\rho &= V_{ENTER}\cos(\theta-\alpha) \\ V_{rot} &= V_{ENTER}\sin(\theta-\alpha) \end{aligned} \tag{3-7}$$

In this phase, Equations (3-3) has to be modified as follows

$$\begin{aligned} d\rho / dt &= -V_\rho \\ d\alpha / dt &= V_{rot} / \rho - \omega \\ d\psi / dt &= -\omega \end{aligned} \tag{3-8}$$

According to Equation (3-7) and $\psi = \pi/2 - \theta$, Equation (3-8) can be rewritten as:

$$\begin{aligned} d\rho / dt &= -V \sin\psi \sin(\alpha + \psi) \\ d\alpha / dt &= V \sin\psi \cos(\alpha + \psi) / \rho - \omega \\ d\psi / dt &= -\omega \end{aligned} \tag{3-9}$$

Linearization of Equation (3-9) shows that first equation results in $\rho = const$ and therefore $V = const$ according to Equation (3-4). The closed-loop system for α and ψ is presented by:

$$\begin{aligned} d\alpha / dt &= -K_1\alpha - K_2\psi + K_\rho\psi \\ d\psi / dt &= -K_1\alpha - K_2\psi \end{aligned} \tag{3-10}$$

The matrix describing the system equation (3-10) is $\begin{bmatrix} -K_1 & -K_2 + K_\rho \\ -K_1 & -K_2 \end{bmatrix}$.With the same gains as in the second phase ($K_\rho = 0.1\,(s^{-1})$, $K_1 = 0.9(s^{-1})$, $K_2 = -0.4(s^{-1})$), the eigenvalues have negative real parts (also slightly shifted from –0.2 set for phase 2).

An important aspect of the application (contrary to an initial "parking" problem) is the online estimation of the relative pose (extracted from the laser profile of the gate). This results in inevitable errors that need to be addressed. In order to compute the final control output, a particular steering design of the ARGO vehicle (skid-steered) is taken into account. In order to make the vehicle turn consistently and avoid unnecessary reactions on the small variations in α and ψ that can even lead to unstable behavior, a dead zone $(-u_{\min}, u_{\min})$ for ω is introduced and final control output is computed as follows:

$$U = N(sign(\omega - u_{\min}) + sign(\omega + u_{\min}))/2 \quad (3\text{-}11)$$

Here N stands for a value to be applied for initiating a vehicle turn.

As can be easily seen if $(-u_{\min} < \omega < u_{\min})$, the resulting U is zero, alternatively it is N.

For speed control, the original terminal speed setting in [64] is required to be 0. The proposed controller forces the target point *T* move with the predefined speed and set the vehicle speed proportional to the distance to the target point *T*. This results in following the target within the gate area. After leaving the gate area, the control algorithm will be switched to 'open-space' GPS/Ins/Odometry based path following.

3.6 Discussion of the Results

Using simulation, all the developed gate-recognition and guiding-crossing algorithms are tested first with simulated data. In addition, the filtering and signature-based recognition modules have been verified with LMS data collected along the manual driving through the gate. Finally, the algorithm has been fully integrated on-board and tested in the field.

Figure 3.15 shows the whole trajectory of ARGO from GPS data during the experiment. The area in the dashed circle is gate recognition and crossing area.

The map filtering module continuously monitors the environment based on the available LMS data. If the gate is actually farther than a soft-upper-limit range, the filtered map is empty. When the gate posts become observable they completely appear in the filtered map after relatively short transition period corresponding to the partial visibility of the

posts, noisy data etc. The experiments show that this period is usually about 1 second and never exceeds 3 seconds.

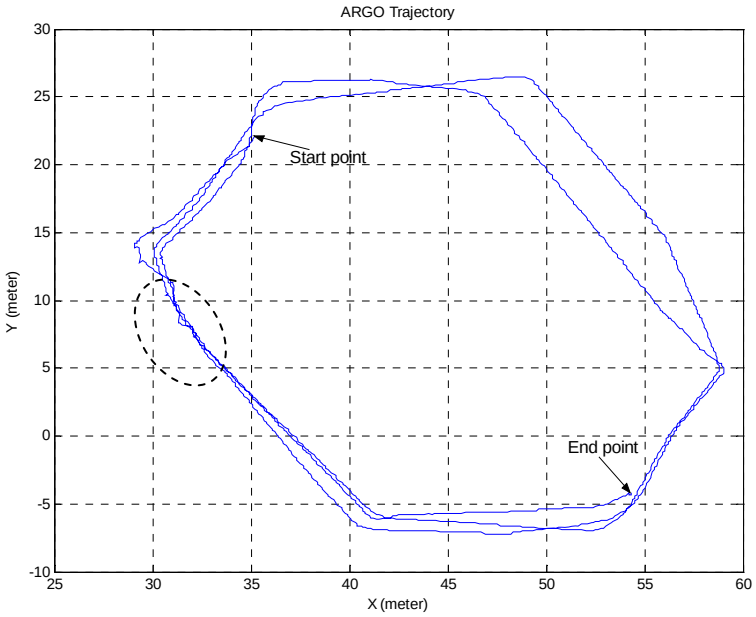


Figure 3. 15 GPS trajectory of experiment [5]

As soon as the gate becomes consistently visible, the recognition module computes the gate signature, searches for a closest canonical signature, and estimates the vehicle pose relative to the gate. The localization module computes the gate middle point C and gate direction P_{GATE} then sends them to the motion control module, which calculates the control outputs U and send it to the hardware (low level controller).

Figure 3.16 illustrated motion through the gate. Small circles correspond to the vehicle trajectory; stars correspond to few consecutive vehicle locations with gate images acquired from those locations.

One can see the uncertainty (of about $2\,m$) in absolute (GPS/INS/Odometry-based) position of the gate. A corresponding map is shown in Figure 4.18 which is from a map

component addressed in chapter 4. In spite of this uncertainty, the LMS-based position of the vehicle relative to the gate is precise enough ($0.1\,m$, $3°$) and the vehicle is successfully navigated through the gate using the algorithm described above.

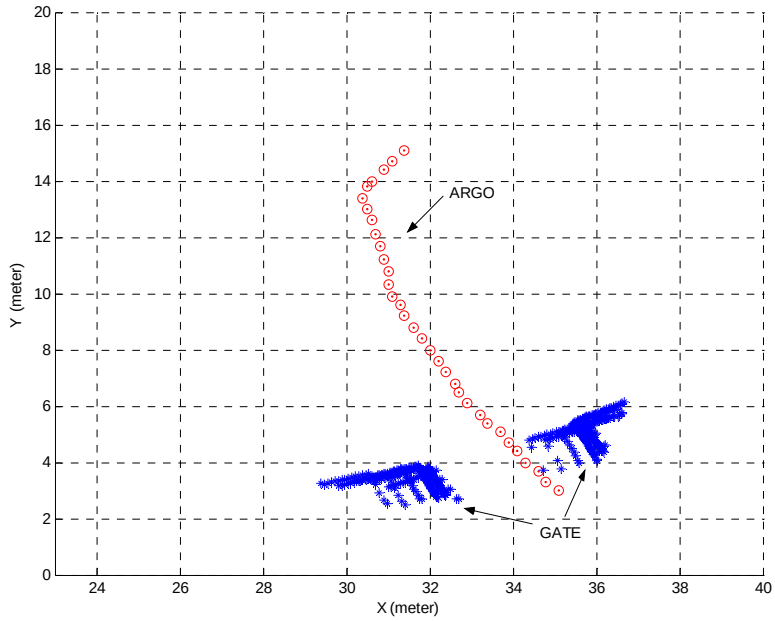


Figure 3. 16 Illustration trajectory of 'gate crossing' [5]

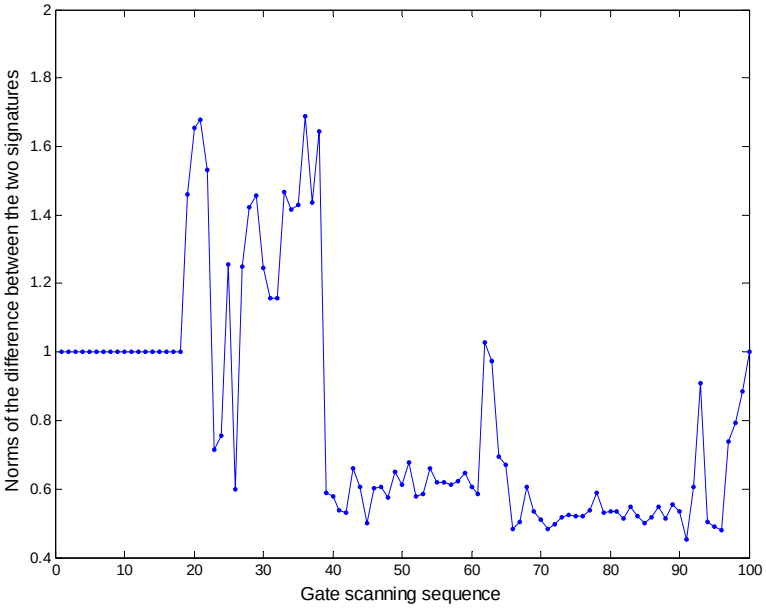


Figure 3. 17 Recognition error [5]

Stable gate recognition is crucial for subsequent steps of the algorithm. Figure 3.17 shows the recording of the recognition results while the vehicle is autonomously entering the gate. The offset (2-norm of the difference) is between the current signature and a canonical signature (taken from the data base according to the zone where the vehicle is located). It may be seen that the error is reduced significantly in accordance with the distance to the approaching gate. The soft-upper- limit range has been set based on theoretical and experimental analysis. If the posts are located too far apart although below the hard-range limit, the recognition results deteriorate. This can be roughly explained as follows: observing a small object (a singular point in the limiting case) does not give enough information for positioning – lateral and orientation offsets are coupled. In practice the computation remains possible but measurement errors are amplified and results become unusable.

The developed concept of gate signature provides an effective method to estimate the relative position between the vehicle and the gate. The motion control algorithm based on the nonlinear transformation to polar coordinates proposed by [64] coupled with on-line estimation of the vehicle pose and enhanced by the moving target point for avoiding singularities ensures the stable gate crossing with acceptable lateral errors of about $0.3\,m$.

In current work the more general fusion of range measurements with GPS/INS/Odometry data have not been addressed. Using methods proposed in [65, 66], the absolute positioning of the vehicle along the "gate crossing" portion of the path can be improved. Such a "tightly coupled" absolute and relative localization is expected to improve the reliability of the navigation system by providing better global position estimates along the whole path and smoothing the phase of approaching the gate.

The numerous field experiments performed with retrofitted ARGO platform have proven that the signature concept, recognition algorithm, and gating controller work very well and provide an effective mean for carrying out the gate navigation tasks. Figure 3.18 illustrates the vehicle passing through the gate in an experimental field.

Figure 3. 18 The 'gate crossing' procedure experimental verification [5]

3.7 Conclusion

Navigation of autonomous ground vehicles through the gate and/or around similar characteristic structures or the environment's features was proved to be difficult. The performed experiments, demonstrated validity and usefulness of the presented concepts for a single gate case. More experiments are needed to verify the conditions for the multi-gate case. The possible improvements of the recognition module would let the navigation algorithm recognize more than one gate and then make its decision according the

requirement from the upper level control. The requirement may consist of an entry into a particular gate or the sequence of gates in any given sequence.

Some limitations related to the hardware (laser scanner) have been encountered. Conditions for object recognition are difficult in some particular situations. Creating an environment map using multi-sensor information should be considered. In particular, multi-laser systems or a laser scanner linked with radar sensors, sonar, or video cameras ought to be considered. The system improvement with the multi-sensor and sensor fusion procedure would make the recognizing procedure more effective and the overall system more robust.

Chapter 4

Map Data Server for Mobile Robots

In chapter 3 the 'gate crossing' of mobile robots has been discussed and the concept of object signature has been described. The developed algorithm guides the robot to cross the gate-like objects successfully. The implementation is based on the given signatures, which include the segments featured from these objects. It works well for the special application, gate recognition and crossing.

In this and the following chapters, the more general problem in which the mobile robot has to handle many different tasks, not just gate crossing problem, is considered. To achieve this goal, first a map has to be built, then the robot may extract the necessary information from the map to assist the required tasks.

This chapter introduces the map applied in mobile robot systems and map data processing. A new user defined JAUS component – Map Data Server - is built. The Map Data Server component is required to provide the source map data of the robot's working space. Two types of map data are supported, namely vector map and occupancy map. For the second type, an adaptive multi-level grid map is used to reduce the computation cost and data transferring time. The component is built using GNU C/C++ under a Linux platform. Also, the future improvements for map data updates, modifications, and the interface with a GIS system are discussed.

4.1 Maps Used for Mobile Robot System

A map is the representation of the geographic environment. It is essential for mobile robot systems that the systems can access the information from the map to execute their tasks. The maps used by the mobile robot systems are various on the format and the content. It is dependent on the requirements of the applications. For some mobile robots, the maps may just include the perimeter of the working space. Others require the complex map such as GIS (Geographic Information System). The methods used to represent the world (world modeling) are also distinct for different maps. [67] and [68] summarize the approached methods of world modeling for mobile robot systems.

Three types of topographic maps are widely used in mobile robot navigation/localization applications. They are occupancy/grid/raster map, feature/vector map, and topological/graph map. In the following section, the occupancy and vector map which are applied in this work are discussed.

4.1.1 Occupancy map

An occupancy map (also called grid or raster map) consists of the grid squares which are arranged in rows and columns. The value attached to each grid square (cell) represents the probability of occupancy or the traversable condition of this cell. The probability of occupancy is usually depicted using the range of [0, 1], zero indicates a free space (not occupied), while one indicates a definitely occupied space. The space is represented as a matrix of cells as shown in Figure 4.1.

The data structure used in occupancy maps (rows and columns of cells) makes the information accessing fast and easy. The cell is the basic element. The cells usually have

the same size and shape which makes the data collection, retrieval simple. On the other hand, an occupancy map appears crude when it is viewed in larger scales. This character makes occupancy map not appropriate presenting the data which have feature finer than the size of the cell [69]. In occupancy map with large cell size, the information about the finer feature's exact location, shape, and boundary can not be retrieved.

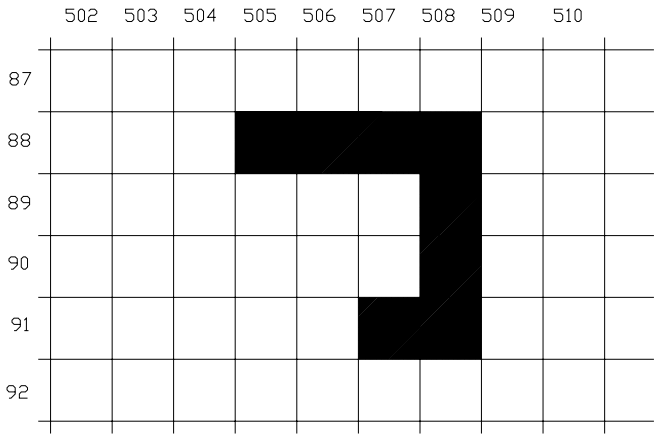


Figure 4.1 Portion of an Occupancy Map

The occupancy map could be controlled by adjusting the cell size, orientation and origin point. Assume the number of cells is fixed, a map with larger cell size covers more area but with less details. A map with smaller cell size requires a larger number of cells to represent the same area. For example, 1 million 1×1 meter cells can represent a space of $1\,km^2$. if the cell size is reduced to 0.5×0.5 meter, then the number of cells in $1\,km^2$

increases to 4 millions [69]. Considering the data storage, an appropriate cell size is required.

To reduce the number of cells while keeping a high resolution, an adaptive occupancy map structure has been proposed and implemented in the thesis. The adaptive occupancy map has a multi-level structure. Each level's cells may be used to represent the occupancy map with a different cell size (resolution). One adaptive occupancy map has more than one resolution according to the features in the map. If there are objects in some areas, the size of the cells in these areas will be reduced, in order to get a higher resolution for better feature expression. Furthermore, the empty cells will keep the maximal size. So, this kind of occupancy map gives the maximal resolution to represent the details in the essential areas and uses the minimal resolution (and therefore the smallest number of cells) for the empty or unimportant areas.

Figure 4.2 demonstrates how the adaptive occupancy map represents the object shape in details with a much smaller number of cells than a conventional occupancy map. Panels (a), (c), and (e) correspond to the adaptive occupancy maps. Panels (b), (d), and (f) correspond to the conventional occupancy maps. To represent a map feature with the same resolution, the adaptive map just needs a distinctly smaller number of cells, but the underlying structure is more complex.

In this application, an adaptive occupancy map is dynamically created from vector-based information on request. Each cell has a value to represent the occupancy status. The algorithm transferring the vector map to an adaptive occupancy map is described in the section 4.3.

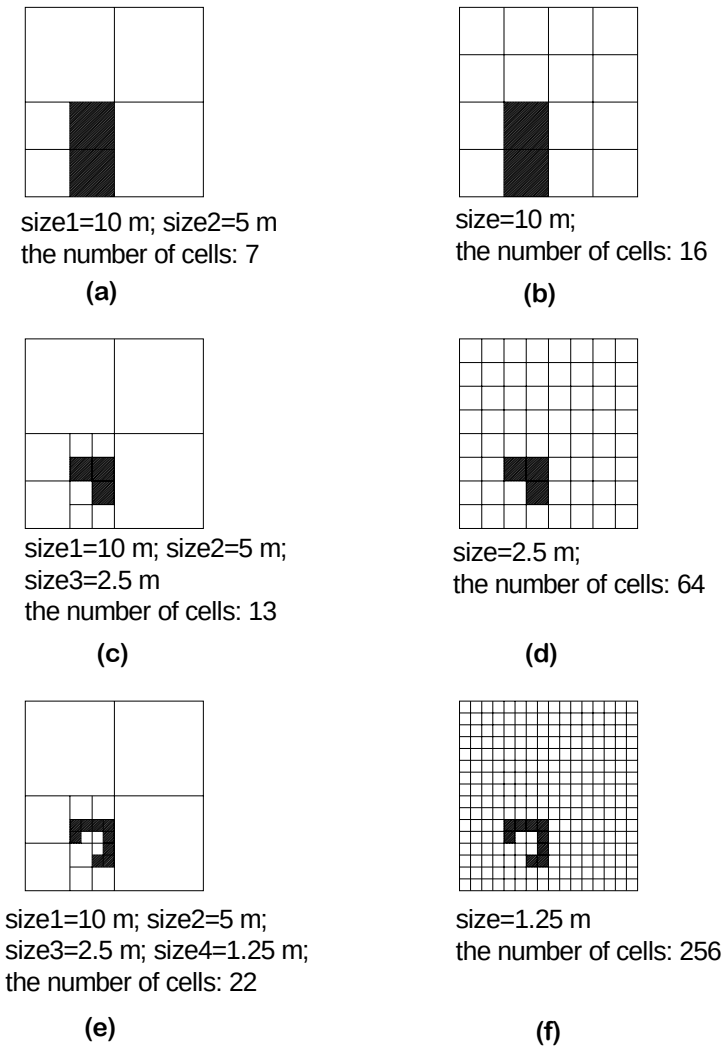


Figure 4.2 Example of adaptive occupancy map

4.1.2 Vector map

The vector map uses geometric features such as points, segments, vectors, polygons and circles to represent the environment. Considering the complex of 3D modeling, most of the vector maps are based on 2-D plane. To store more information such as elevation or other required data, some attributes which describe more detailed characteristics of the objects may be added to the 2-D geometric features (objects).

The basic element of geometry feature is the point. All the geometry features in the vector map are represented as sets of points. Figure 4.3 shows the examples of rectangle and circle which consist of points.

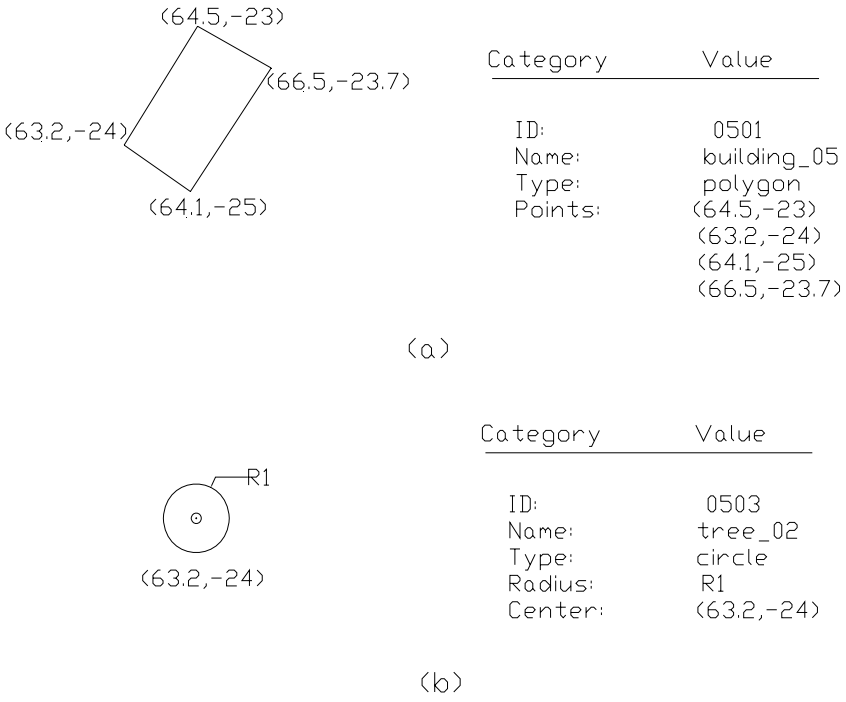


Figure 4.3 Objects in vector map

The vector data could be stored in three ways: point objects, polygon objects, and circle objects (objects containing points connected by circular arcs are not currently supported) A single point feature is represented by a single location value. Linear features are represented as one or more vectors. More complex objects are represented as a series of vectors – a polygon. A polygon is a closed set of vectors. The polygons must be defined according to special rules in order to represent the information effectively. The most common specifications are: (A) the boundary is described in clockwise order, (B) the boundary of the polygon is closed, and (C) the boundary vectors can not cross. [69]

Points, vectors, polygons and the relationships among these objects may represent complex geographic objects by adding more information such as attributes of objects. There could be many different object types (with different attributes) in one vector map, defined by means of different map layer. Figure 4.4 shows the structure of a vector map with different layers. The objects in a vector map are classified according to their attributes and put into different map layers. Each object is identified with an ID, a name, and geometry definitions.

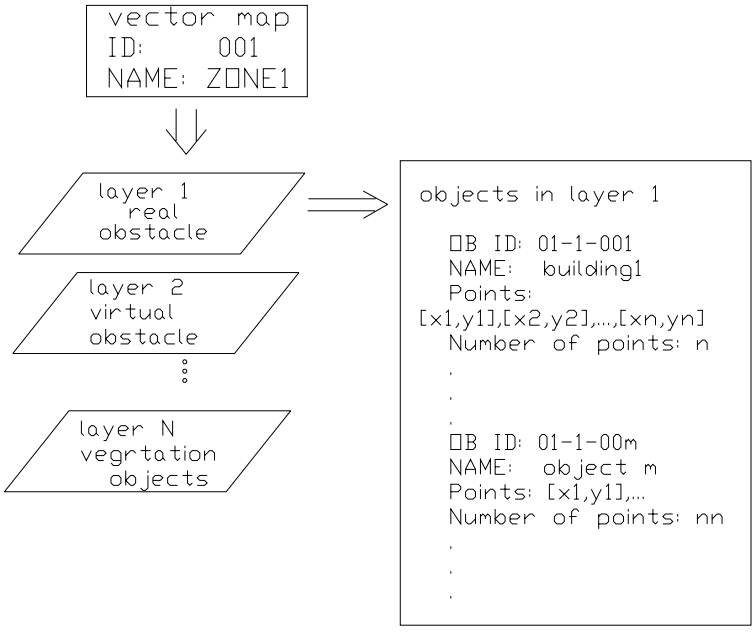


Figure 4.4 Vector map with multi-layer

In this work, for the vector map, all objects are presented by polygons that define object perimeters or extents. Typical objects are (1) real obstacles (buildings, poles, trees, walls, fences, holes, ditches, roadside curbs etc.), (2) areas covered by vegetation (lawns, forests, swamps etc.), and (3) water bodies (streams, lakes etc.). Each object is described by its polygonal outline and set of attributes. The vector map is logically organized into multi layers. Each layer overlays a common geographic area and contains objects of a

particular type. Some objects may belong to several layers. For instance, a tree may be located in the real obstacle layer and also in the vegetation layer.

4.2 Conversion between Different Coordinate Systems

In this application, the robot is considered to be running in a two dimensional planar environment. Three elements are needed to locate the pose of the robot. There are three main coordinate systems that are used in general: geographic global coordinate, planar local coordinate and robot-centered (planar) local coordinate system. The geographic global coordinate system uses latitude (degree) and longitude (degree) to represent a point. The planar local coordinate system uses x (meter) and y (meter) to represent a point. The x axis points to the north and the y axis points to the east. The origin point of the planar local coordinate system could be any point on the planar. The third coordinate system, robot-centered local coordinate system is attached to the robot pose (or the range sensor mount point for this thesis). This coordinate system always moves along with the robot. The axes are conventionally oriented Y axis along with the robot's heading and X axis pointing to robot's left.

According to the definition of the ellipsoidal model from WGS84 (World Geodetic System 1984), Figure 4.5 shows an example of a conversion from the geographical $[latitude, longitude]$ to the planar local $[North, East]$ and to the robot local coordinate system $[x, y]$. Assuming the robot location in the map is given $[lat, lon]$ in units of [degree, degree] in the spherical global coordinate system. To transfer to local planar

coordinate system, the left bottom of the map extent is considered as the new origin for the local planar coordinate system. If the coordinate of the left bottom point of the map is $[lat_0, lon_0]$, then the corresponding local coordinate is $[0,0]$. This point is called the base point for the conversion.

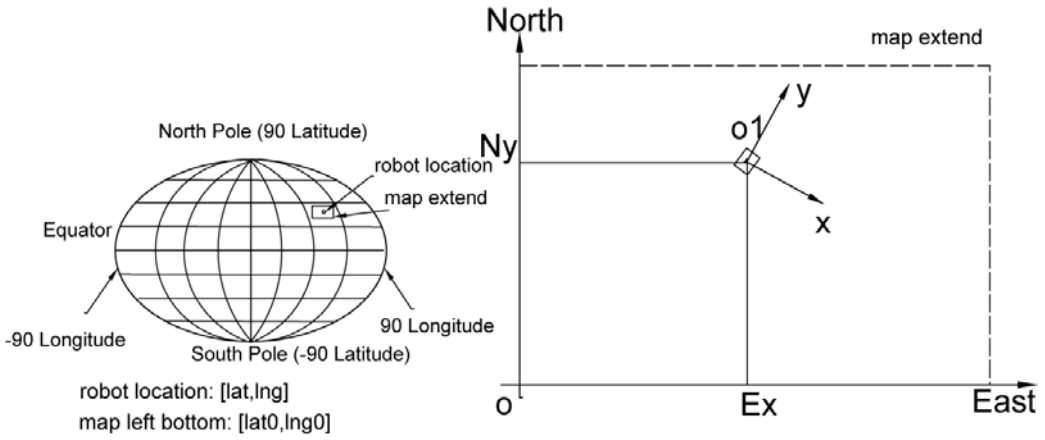


Figure 4.5 Example of coordinate system conversion

Actually, the earth model is an ellipsoidal model. So the radii from different points on the earth are variable. The following equation may be used to calculate the radius at different location.

$$R = R_0\sqrt{(1-E^2)/(1-E^2\cos^2(lat))} \tag{4-1}$$

Where, R is the earth radius at any point with lat latitude; R_0 is the radius at the Equator; E is eccentricity of the Earth. In WGS84, $E = 0.081819191$.

If the base point and the radius are known, the coordinate of point $[lat, lon]$ in the planar coordinate system could be calculated by the following equations:

$$\begin{aligned} N_y &= R(lat - lat_0) \\ E_x &= R(lon - lon_0)\cos(lat) \end{aligned} \tag{4-2}$$

Inversely, the latitude and longitude can be calculated from $[E_x, N_y]$.

$$lat = lat_0 + (\frac{N_y}{R})$$
$$lon = lon_0 - \frac{\frac{E_x}{R}}{\cos(lat)} \quad (4\text{-}3)$$

The latitudes and longitudes in the above equations are in radian.

4.3 Converting Vector map into Adaptive Occupancy Map

Various ways to transform an occupancy map into a vector map are addressed in [70]. The transformation from the vector map to the conventional occupancy map is simpler; however in this case it needs to be addressed, since the developed adaptive occupancy map has more complex data structure than the conventional one. The Map Data Server in the current form supports the occupancy map with adaptive resolution and the vector map. The vector map is stored in the system as pre-assigned data. The occupancy map is created from the vector map by the Map Data Server component when the occupancy information is queried by other components. Considering that the application will typically be used in large scale outdoor environment, three level resolutions are adopted: $10\,m$, $1\,m$, and $0.1\,m$. The lower resolution (bigger cell size) is used for global path planning and other high level tasks. The higher resolution (smaller cell size) is used for such tasks as local path planning, obstacle negotiation and environment monitoring. The components querying the occupancy map must specify which resolution level they need. The procedure of converting the vector map into the adaptive occupancy map is described below.

Step 1: when the adaptive occupancy map is requested, all vectors, which are in the queried extents and meet the queried attributes, are extracted from the vector map data base;

Step 2: an occupancy map with the queried extents is initialized using the lowest resolution (10 x 10 meter cell in the application);

Step 3: all the cells intersecting with the vectors are marked as occupied;

Step 4: for all the occupied cells, if the resolution is not the highest, the steps 2 and 3 are repeated with higher resolution;

This procedure uses recursive algorithm to produce the different level cells (see Figure4.2).

4.4 Map Clipping Algorithm

Usually, the map used for outdoor mobile robots covers a wider area. It includes a large number of objects in several types. For navigation at a particular time period and in particular location, a detailed local map is needed. Extracting necessary information from the global map is computationally expensive. To keep the information extraction computationally effectively, a map clipping algorithm has to be used.

The map clipping for the vector map is addressed below. It is rather trivial for an occupancy map (due to its row/column underlying structure). According to the definition of a vector map, all objects are defined as polygons with a closed sequence of vectors. The polygons could be concave or convex, but no self-intersecting or overlapped edges.

There are several existing polygon clipping algorithms, such as Sutherland-Hodgman, Laing-Barsky, and Weiler. Each of them has its strengths and weaknesses. The Sutherland-Hodgman algorithm is relatively simple and also very efficient when the polygon is completely inside or outside of the clip window. The Laing-Barsky algorithm is faster than the Suther-Hodgman algorithm when many polygons' lines intersect with the clip window, but it's sensitive to the complexity of the polygon. The Weiler algorithm is more complicated, but it could be used for non-rectangular clip windows. [71]

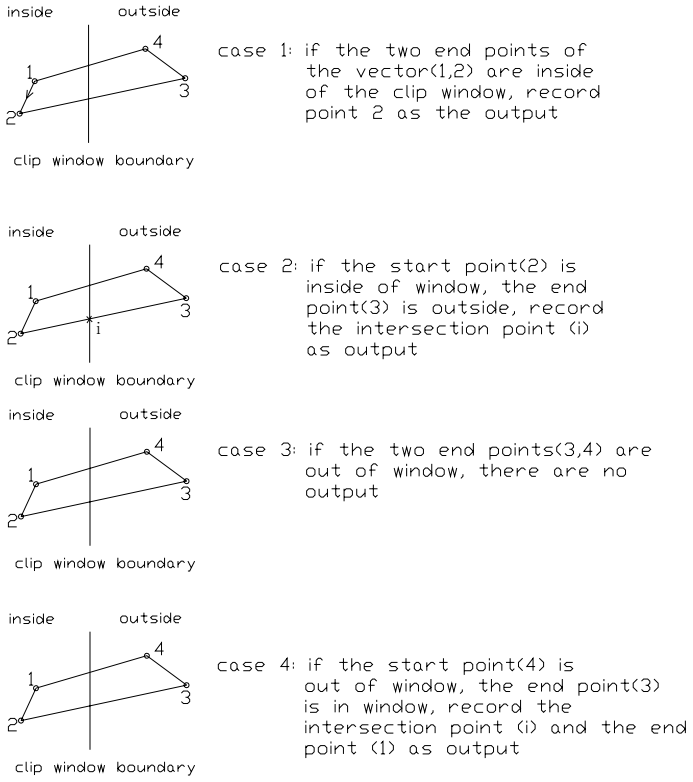


Figure 4.6 Sutherland-Hodgman polygon clipping

Considering the needs for the application (randomness of the object shape, rectangle clipping window), the Sutherland-Hodgman algorithm is chosen. It uses a divide-and-conquer strategy to solve the problem [71]. It clips the polygon by each clipping boundary in order. During the whole procedure, there are four cases that need to be

analyzed when clipping a polygon to a single edge of the clipping window. Figure 4.6 illustrates the possible cases. In Sutherland-Hodgman clipping, the four clipping steps are independent to each other. The output from the previous clipping is the input to the next one [72].

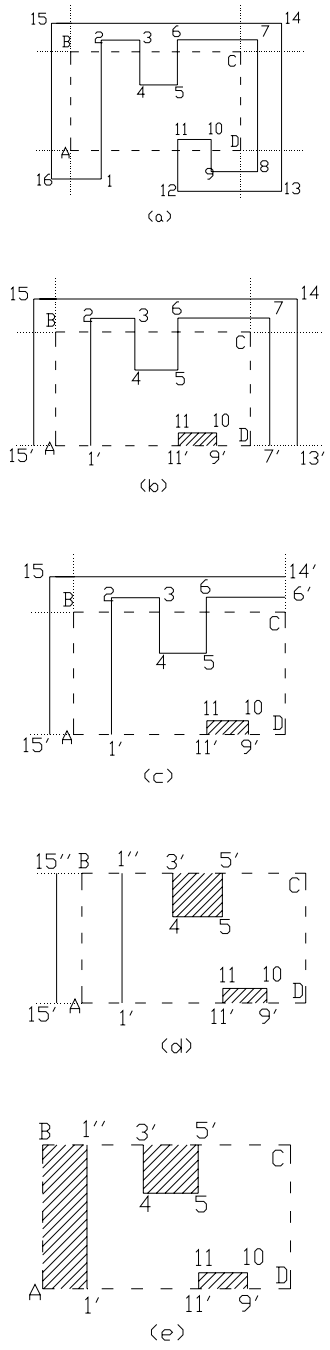


Figure 4.7 Steps of the improved clipping

The weakness of this algorithm is that it always returns one set of vertices. If the actual result is single polygon after the clipping (it is true for convex polygon), it is easily to pick the single polygon. However, if the clipped polygon is concave, there may have multiple polygons created after the clipping. The algorithm does not have a mechanism to redistribute these vertices over different polygons. In this case, a single polygon with overlapped edges will be created. For mobile robot navigation, it may lead to an error, which results the blockage of a possible proper path.

Due to this issue, some adjustment for Sutherland-Hodgman algorithm is made to address how to find the corresponding vertices for the new polygon(s) after the clipping.

An example of using the improved algorithm is shown in Figure 4.7 ((a) – (e)).

In Figure 4.7, the original polygon has 16 vertices (16 vectors), which are marked by digits 1 to 16. The clipping window is a rectangle with the vertices marked by A, B, C, D.

The improved clipping algorithm includes the following steps:

Step 0: extending each edge of the clipping windows to infinity (Figure 4.7(a));

Step 1: clipping the polygon with the straight line from the extension of the bottom edge of clipping window in step 0 (Figure 4.7(b)). This includes connecting each two sequential points if all the points between them are inside of the original polygon. It also means computing the new polygon if the previous connecting action goes back to the first point. Now there are two new polygons. Let's name them as polygon a and b. Polygon a has 12 vertices (in clockwise, $1'$, $15'$, 15, 14, $13'$, $7'$, 7, 6, 5, 4, 3, 2). Polygon b has 4 vertices (in clockwise, $9'$, 10, 11, $11'$). These two polygons will be considered separately during the following clipping steps.

Step 2: clipping the two polygons with the straight line from the extension of the right edge of the clip window in step 0 (Figure 4.7(c)). There are no intersections between polygon b and the right edge. Polygon b is in the left of the clipping line. So, polygon b does not change. The connecting action described in step 1 will be done also. For polygon a, the vertices are changed to $6'$, $14'$, 15, $15'$, $1'$, 2, 3, 4, 5, 6 after clipping.

Step 3: clipping the two polygons with the straight line from the extension of the up edge of clip window in step 0 (Figure 4.7(d)). Polygon b is still unchanged. The previous polygon a is divided into two new polygons. Let's call them polygon c and polygon d. Polygon c has four vertices 5, $5'$, $3'$, 4. Polygon d also has four vertices $1'$, $1''$, $15''$, $15'$.

Step 4: clipping the three polygons b, c, d with the straight line from the extension of the left edge of clip window in step 0. Polygon b and polygon c remain unchanged and polygon d is changed to $1'$, $1''$, B, A. Figure 4.7(e) is the final result of the clipping. There are three new polygons created from the original polygon.

To demonstrate the changing on the vertices of the clipped polygon(s), Figure 4.8 shows the lists of vertices in the above steps and the connections among these vertices. In Figure 4.8, the sign $\otimes$ stands for the case when there are no legal connections between two vertices. The sign $\rightarrow$ stands for the case when two vertices can be connected. The adjusted Sutherland-Hodgman algorithm is represented using pseudocode in Figure 4.9.

The adjusted clipping algorithm uses the rules of the traditional Sutherland-Hodgman algorithm to judge the intersection points when clipping against the four edges of clipping window. After each clipping, the resulted vertices are regrouped to one or more

new polygon(s) before they continue to the next clipping. The algorithm has be coded in C and tested on Linux platform.

$$\textit{Vertices of polygon}: 1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9 \to 10 \to 11 \to 12 \to 13 \to 14 \to 15 \to 16;$$

$$\textit{Vertices of clipping window}: A \to B \to C \to D;$$

$$\textit{Vertices from the first clipping step (clipping against } AD): 1' \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 7' \otimes 9' \to 10 \to 11 \to 11' \otimes 13' \to 14 \to 15 \to 15'$$

$$\textit{new polygon\_1}: 1' \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 7' \to 13' \to 14 \to 15 \to 15'$$

$$\textit{new polygon\_2}: 9' \to 10 \to 11 \to 11'$$

$$\textit{Vertices from the first clipping step (clipping against } DC): \textit{new polygon\_2}: 9' \to 10 \to 11 \to 11'$$

$$\textit{new polygon\_1}: 1' \to 2 \to 3 \to 4 \to 5 \to 6 \to 6' \to 14' \to 15 \to 15'$$

$$\textit{Vertices from the first clipping step (clipping against } CB): \textit{new polygon\_2}: 9' \to 10 \to 11 \to 11'$$

$$\textit{new polygon\_1}: 1' \to 1'' \otimes 3' \to 4 \to 5 \to 5' \otimes 15'' \to 15'$$

$$\textit{new polygon\_1 divided to}: \textit{new polygon\_1\_1}: 1' \to 1'' \to 15'' \to 15'$$

$$\textit{new polygon\_1\_2}: 3' \to 4 \to 5 \to 5'$$

$$\textit{Vertices from the first clipping step (clipping against } BA): \textit{new polygon\_2}: 9' \to 10 \to 11 \to 11'$$

$$\textit{new polygon\_1\_1}: 1' \to 1'' \to B \to A$$

$$\textit{new polygon\_1\_2}: 3' \to 4 \to 5 \to 5'$$

Figure 4.8 Demonstration of vertices connection

```
define : VERTEX = {id, point, *Next};
         Polygon = {number of point, *head};
Given : clipped polygon : P_o{N_0, *V_1^0};
         output_polygon[N_max];
         output_count = 0;
%after clipping against bottom, the list of returned vertices
V_i^1, i = 1,...N_1;
let  t1 = 1; mark = 1;
%mark the broke chain
 for i = 1 : N_1 − 1
{   if V_i^1, V_{i+1}^1 ∈ P_0 | V_i^1 ∈ P_0 | V_{i+1}^1 ∈ P_0
         V_i^1 → Next = V_{i+1}^1;
         mark++;
    else
         V_i^1 → Next = Null;
         temp_list[t1] = {V_{i−mark}^1, V_{i−mark+1}^1, ...V_i^1};
         mark = 1; t1++;
    end }
let  t2 = t1;
% find the existed closed chain
 for i = 1 : t2
{   if (line{temp_list(i, first), temp_list(i, last)} ⊂ P_0)
    {   output_count++;
         output_polygon(output_count) = temp_list(i,:);
         delete temp_list(i) from temp_list;
         t1−−; } }
%repair the broke chain
 for i = 1 : t1 − 1
{   if (line{temp_list(i, last), temp_list(i + 1, first)} ⊂ P_0)
    {   temp_list(i, last) − Next = temp_list(i + 1, first) } }
t2 = t1;
% find the repaired closed chain
 for i = 1 : t2
{   if (line{temp_list(i, first), temp_list(i, last)} ⊂ P_0)
    {   output_count++;
         output_polygon(output_count) = temp_list(i,:);
         delete temp_list(i) from temp_list;
         t1−−; } }
if (t1 ≠ 0)  failure;
else output_polygon;
```

Figure 4.9 Algorithm of polygon clipping

4.5 Map Data Server Component and Its Messages

The Map Data Server is the source of map data used for navigation, path planning, and localization components. The basic requirements for the Map Data Server component are:

- To provide the ability for JAUS components within a single node to access map data in a bounded time;
- To store the map data and provide efficient access to multiple types of map data;
- To present map data to a client based on map extents, data type, and representation type. Two types of representation are supported: occupancy and vector map;

Figure 4.10 shows the processing flow of the Map Data Server component.

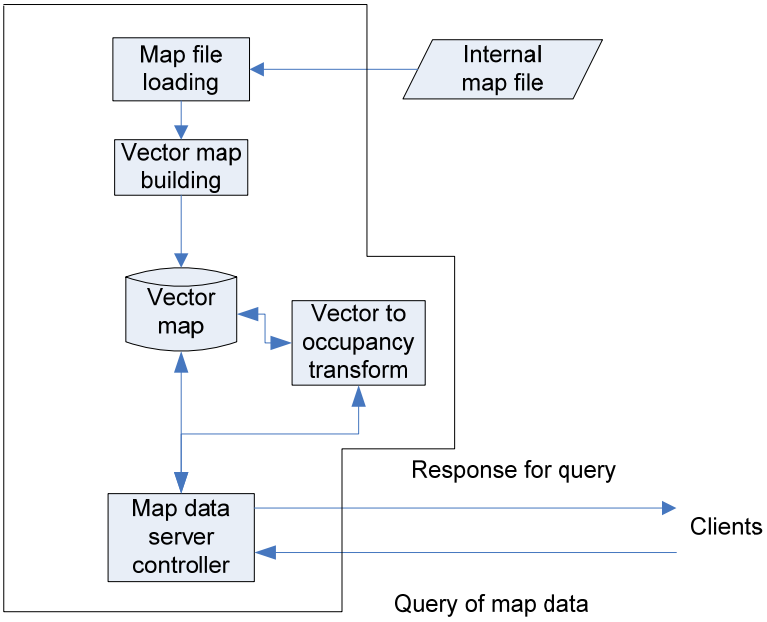


Figure 4.10 Flow chart of Map Data Server component

The Map Data Server component uses the standard JAUS message header structure. In addition to the existing JAUS core input and output message sets, (such as shutdown,

standby, reset, etc.) the following messages for the map data query and the corresponding responses are currently supported:

Query map capability message;

Query vector map data message;

Query occupancy map data message;

Response for map capability query message;

Response for vector map data query message;

Response for occupancy map data query message;

All the contents of these messages are listed in Appendix C.

Before any other component can query map data from the Map Data Server, they will send a "query capability" message to know what kind of map data is available from the Map Data Server and what the extents of the map are. Then, according to their needs, they send a query with defined extents and attributes of the objects. The Map Data Server component will search the map database to extract what they need and then send the required information as the response message. The structure of all messages is defined according to the JAUS system requirements.

For each map data query from any other component, the Map Data Server will check the validity of the query. If the query is invalid, for instance, if the query extent is beyond the map boundary or the query area is out of the map, an empty response or the data in a changed extent will be sent back. For some cases, if the message is longer than the limit of a JAUS message (4096 bytes), it will be regarded as a large data set. Large data set will be sent as multi-packed data with sequence numbers included.

4.6 Simulation and experiments

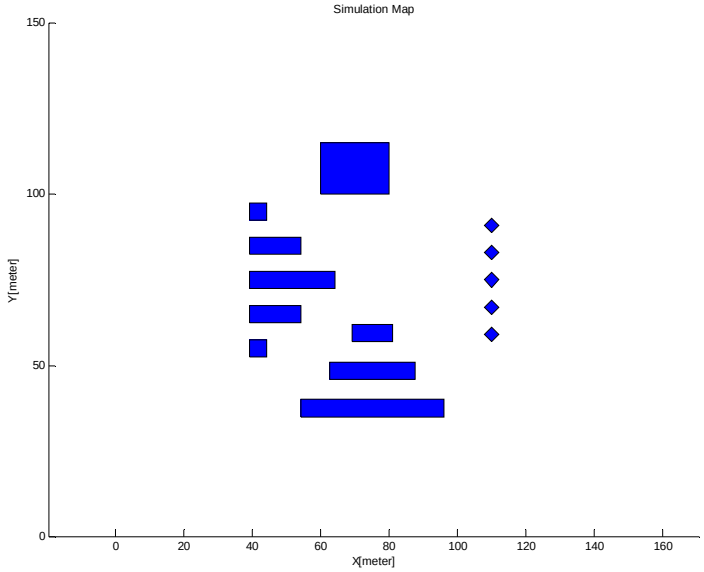


Figure 4.11 Geometry layout of the simulation

The objective of the simulation presented in this section is to test the adaptive occupancy map method and the polygon clipping algorithm for a geometry (vector) map.

The map simulates a popular outdoor warehouse layout, which is about $150m \times 150m$ in size and has 14 piles of freight as the objects. Figure 4.11 demonstrates the geometric format of the simulation map.

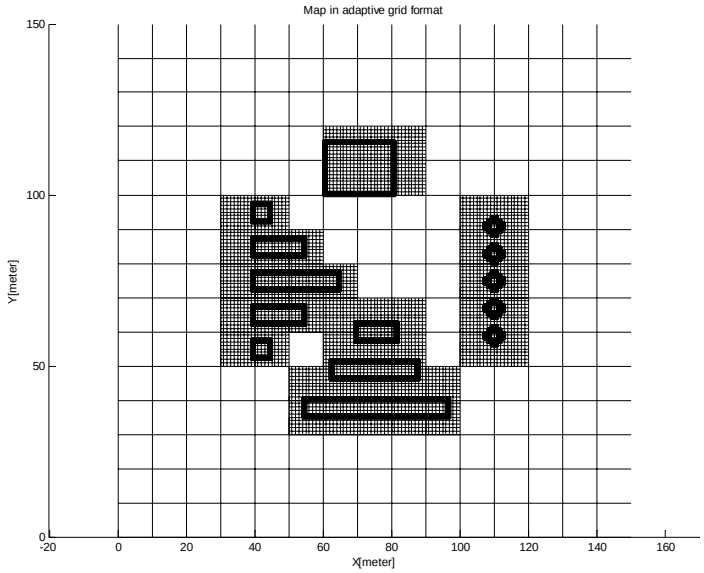


Figure 4.12 Adaptive occupancy map of the simulation geometry

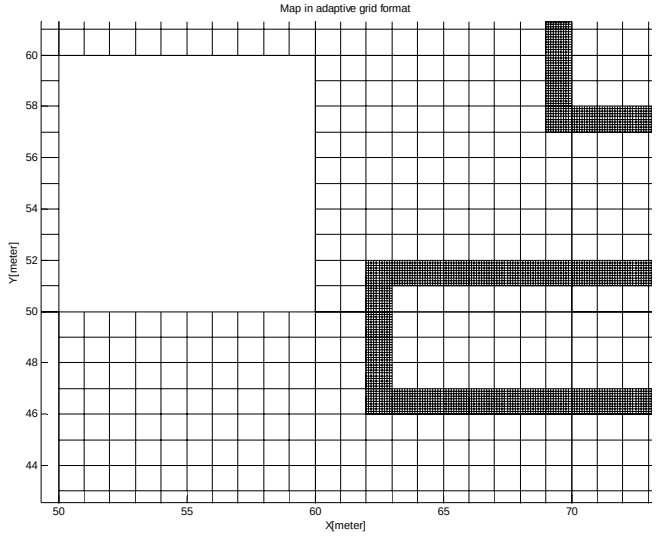


Figure 4.13 Enlarged portion of the adaptive grids (1)

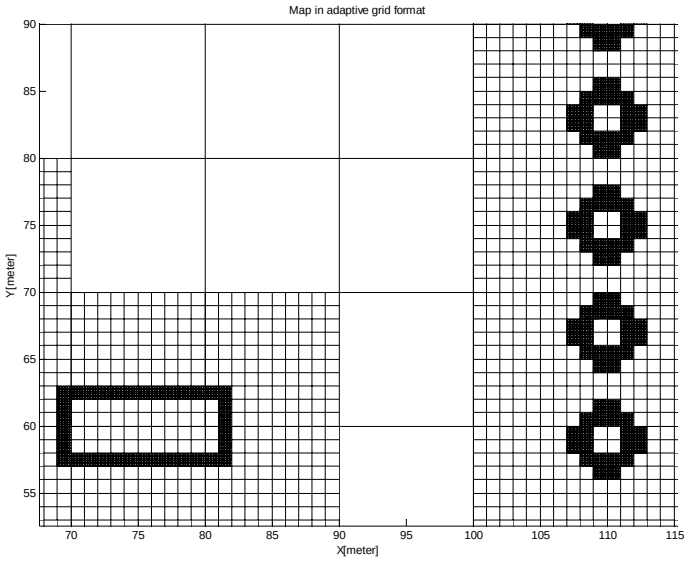


Figure 4.14 Enlarged portion of the adaptive grids (2)

Applying the adaptive occupancy map algorithm described previously, the corresponding adaptive occupancy map may be built from the vector map. Three resolutions are included. The basic resolution uses $10m \times 10m$ cell. It could be changed to $1m \times 1m$ cell or $0.1m \times 0.1m$ cell, according the occupancy situation (Figure 4.12 to 4.14).

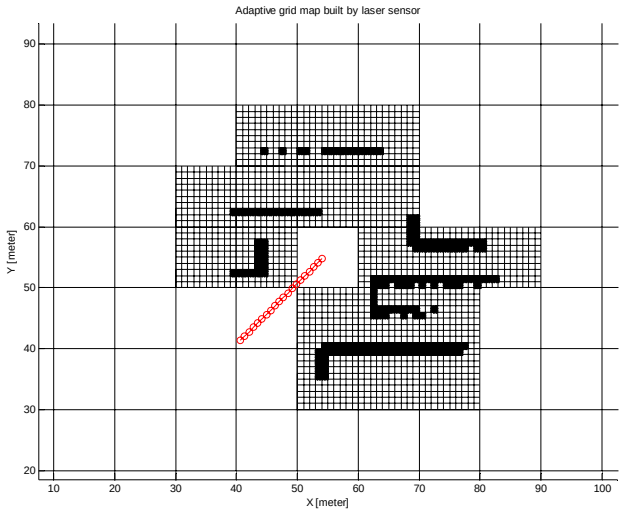


Figure 4.15 Adaptive occupancy map built by the laser sensor

The adaptive occupancy map algorithm also can be used for the map building process by the obstacle detection sensors. In this case, the laser sensor scans the space during the movement of the robot and builds the adaptive occupancy map (see Figure 4.15).

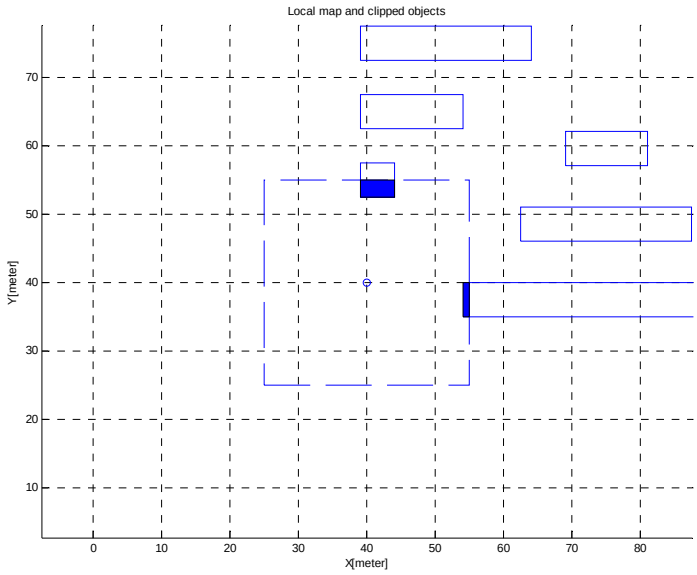


Figure 4.16 Simulation of the polygon clipping (1)

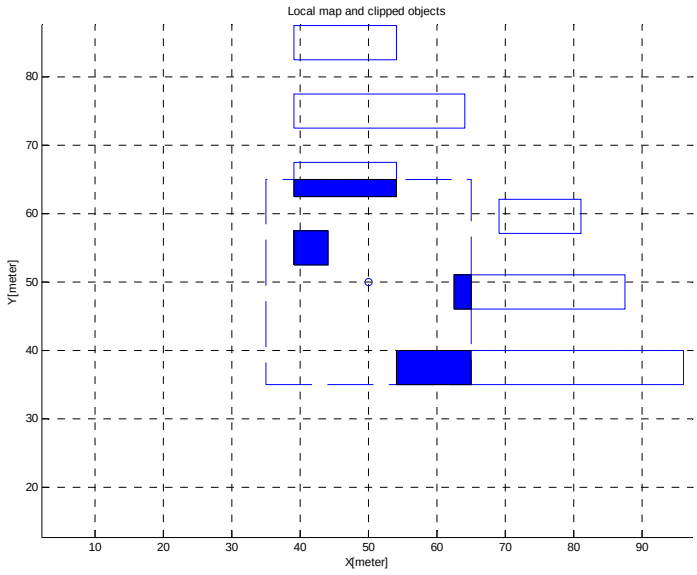


Figure 4.17 Simulation of the polygon clipping (2)

The clipping algorithm is also tested using the simulation method. A local map which is smaller than the whole map and localized around the robot is defined. In the simulation and the later experiments, the local map is a $30m \times 30m$ rectangle with a fixed direction, up being north, and right being east. The local map is moving with the robot and is always centered at the robot's location (Figure 4.16 and 4.17).

The $30m \times 30m$ local map is used for the localization improvement and obstacle detection (discussed in the following chapters). The objects from the whole map are clipped by the local map boundary (rectangle). The clipped polygons which are inside of the local map are the objects for local consideration. Figure 4.16 and 4.17 demonstrate the clipping results from the simulation. The shadowed portions of the objects are the local objects when the simulated robot is running in the space.

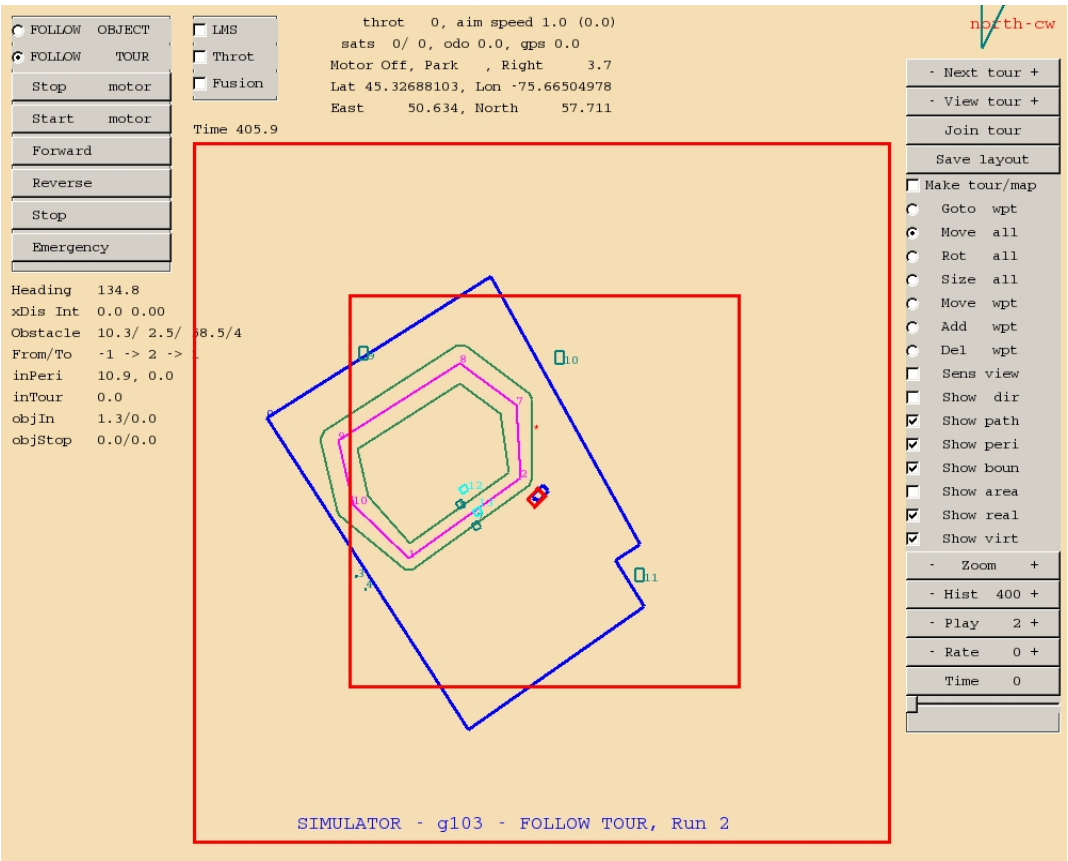


Figure 4.18 Screenshot (1) of the Map Data Server component GUI for the experiments

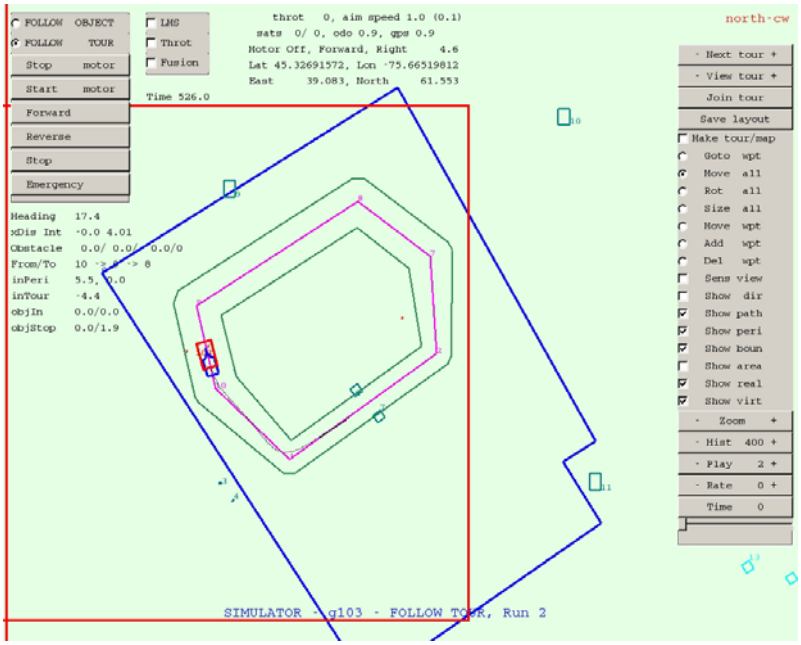


Figure 4.19 Screenshot (2) of the Map Data Server component GUI for the experiments

Figure 4.20 Mobile robot carries out the patrol task using Map Data Server component

The experiments were done in the environment that included objects with various physical sizes. Figure 4.18 and 4.19 are screenshots of the system GUI during the experiments. They show the robot pose and the actual trajectory. The desired path (in

magenta), the objects around the robot (small green rectangles), extents (in red) and the perimeters (in blue) are provided by the Map Data Server component.

Figure 4.20 is a photo taken during the experiments. The robot is carrying out the patrol task using the map data provided by the Map Data Server component. The red box in Figure 4.20 corresponds to small green rectangles on the left and right side of the robot path in Figure 4.18 and 4.19.

4.7 Conclusion and Discussion

The Map Data Server component has been developed to provide map information for other components in JAUS-type architecture. This component is a necessary part of an autonomous robotic system dedicated to the tasks, such as navigation, localization, and path planning. The component is built using GNU C/C++ under a Linux platform. It has been tested and is currently being integrated into a mobile robotic system under development in a company. The simulations and experiments show that the developed algorithms work well and the component delivers the map information correctly.

Currently, the Map Data Server component supports only the following functionalities: (1) initially storing the map and (2) providing the map data by query. All of the map data is permanent, so it cannot be deleted or changed. More functions are needed and will be added after the current version is extensively tested and proved to be consistent.

The future work will be focused on two issues, the interface with an existing GIS system and the map data modification. For this version of the Map Data Server component, all

map data is stored in the internal map files and loaded when the component starts. An interface between this component and external GIS data will provide the system with wider adaptability to different environments.

For map modification, some other problems have to be considered before that ability is added to the Map Data Server component. One of the problems is how to define the authority for modification of existing map data. Map modification can be initiated in two ways. First, the system/subsystem commander components may modify the existing map data. These components which have a high level authority (so the modifications have a high reliability) may send messages to the Map Data Server to modify the map. This way is clear and easy to control. The second situation arises if the changes of the map are detected by other components (as e.g. ranger sensor) according to the sensing data. Then the components who detected the changes may send the message requiring the changes in the database. The potential problem here is that the reliabilities of required modifications are variable. Some modification alarm may be from the false detections due to noisy sensor data, wrong position estimates, or poor observation conditions. If the map server always accepts these modifications, the map would become disordered and unauthentic.

Due to the above consideration, some new components are under development related to the tasks of map data modification. They are: the Map Data Manager Component, the Localization component, and the Recognition component. Any queries about map data modifications would be checked by the Map Data Manager before they are applied. The Map Data Manager also can get assistance from the Localization and the Recognition component to verify the reliabilities of these messages.

As one of the components for localization functionalities, the Map Data Server supplies map information to the whole mobile robot system. The map information may be presented in vector/geometry or occupancy /grid formats.

The vector/geometry map may be used for data association of laser sensor scanning, which will be addressed in the next chapter. The occupancy map may be used for path planning, which is outside of the scope of this thesis. The algorithm of the adaptive grid may be used to build the traversability map more effectively.

Chapter 5

Data Association of Scanning from Laser Range Sensor

5.1 Introduction

In chapter 3, an algorithm has been developed to navigate the mobile vehicle throughout the gate using the signature-based approach for matching. In chapter 4, a JAUS component, the Map Data Server, is described that is responsible to supply the necessary map information for robot navigation/localization. The concern in this chapter is how to generalize the special signature-based matching in order to make it applicable for matching generic objects (polygons for this work). Moreover, the prior map should be used to replace the canonical signatures and some matching algorithms should be created to replace the signature comparison.

The work described in this chapter addresses data association algorithms which can be applied to improve the localization performance.

As a mobile security agent, the robot is never allowed to cross over the perimeter of the given area. All the objects involved in this development are assumed to be line segments or polygons - no curved or circle shape objects (actually this is true for most outdoor environments). There is a prior map of the environment and the robot may load the whole map or part of the map. All real (physically existing) map objects and some virtual

objects (like the boundaries, the different task zones, etc.) are represented by polygons. Each polygon is composed of serials segments (straight lines). All the map objects are closed polygons. Not all of the real map objects are visible to the sensors (currently laser range finders). Some real map objects (like pools, ditches) cannot be detected by the sensors. So the robot has to navigate and localize itself with both sensors and map.

The elementary localization functions are available from equipped GPS/INS/Odometry on the robot platform. Considering the uncertainty of GPS sensor and the rough surface of the outdoor environment, more precise pose estimation which using the equipped 2-D laser range finder will enhance the existing localization performance.

The following contents are proposed in this chapter. In section 5.2, literatures about line extraction and sensor data matching are discussed. Section 5.3 describes the range sensor data processing. Section 5.4 introduces the proposed map scanning concept. Section 5.5 shows the matching method and how to apply the result to pose estimation. In section 5.6 the experimental results are discussed and in section 5.7 some conclusion are formulated.

5.2 Literature Review

The data association method discussed here can be divided into two main steps: data processing and feature matching. The data from laser range scanner (points) is regrouped by the different objects and the line (segment) feature is extracted from each point group in data processing step. Then, the achieved features from scanner are paired to map

features or previous scanned features in feature matching step. The related literature is discussed in the following sections.

5.2.1 Line feature extraction methods

Originally, line/segment extraction is studied in computer vision processing. It becomes an essential step for feature extraction in data processing for mobile robot localization using range sensor. Paper [73] summarizes six distinguishable line/segment extraction algorithms from mobile robotics and computer vision fields. They are Split and merge, Line regression, Incremental, Random sample consensus, Hough transform and Expectation maximization.

The comparison of these six algorithms with simulation is done in [73] and the comparison result suggests that split and merge is the best line extraction method for mobile robots localization, especially when a prior map is used. More details about the comparison of these algorithms may be found in [73].

5.2.2 Scanning feature matching methods

Basically, two approaches are used for data association in mobile robot localization. One is feature matching. The pose of the mobile robot is estimated by matching scanning points or geometric primitives from the scan and from the map or another scan [74, 75, 76, 77]. Another approach is a probabilistic method. The pose of the mobile robot is estimated by computing recursively probabilistic distribution over the state space of the robot's position [78].

For the first matching approach, there are six representative works: Complete Line Segment (CLS) [75], Cox [25], Iterative Dual Correspondence (IDC) [76], Combined Scan Matcher (CSM) [78], Anchor Point Relation (APR) [77], and Kalman methods [79].

Cox matching method [25]

The Cox matching method assumes that both translation and rotation from the two sets are small. This matching method does not require a feature extraction stage. The reference set uses the prior map segments and the current set uses directly the range data points from the laser sensor. It assigns scan points to line segments. The algorithm is described in [25]:

1. A set of scanning points is from the scanning and another set of segments is from the map. For each point in the scanning points, find the corresponding segment from the map segments which is closest to this point;
2. Find a congruence that minimizes the sum of squared distance between the scanning points and their corresponding segments;
3. Translate and rotate the original scanning points according to the congruence from (2);
4. Repeat steps 2-3 until the found congruence is converged. The sum of the components of the congruence from the beginning is the offset.

One restriction of the Cox method is the small error assumption. Also, the convergence is very slow in some cases. Furthermore, the covariance matrix used in step 2 has a high computation cost.

Iterative Dual Correspondence method [76]

Iterative Dual Correspondence (IDC) is a point to point matching algorithm. Two sets of scanning points (current scan and reference scan) are matched using an iterative

algorithm. Each point from the reference scan is used to find a corresponding point from the current scan. After the points pairs matched, the pair of points which are corresponding points are connected (pair of points to segment). By comparison of the length of these segments, IDC can judge if the pairs of points are correct or are good matching. Considering the existence of noisy points, IDC ignores the points from the current scanning which have the segments longer than a defined threshold. Only the remaining points are used for matching.

Complete Line Segment [75]

The Complete Line Segment (CLS) method processes the matching of reference and current data. The algorithm procedure includes line segment extraction, complete line segment finding, complete line segment matching, and an estimate of the matching result. To find the best matching pair, the estimate step in CLS has to check all the possible matching pair to get the best pairs. Considering the visibilities of reference scan, some line segments from the current scan have to be defined as sub-complete line segments in order to find the matching from the reference scan.

Anchor Point Relation [77]

The Anchor Point Relation (APR) matches the current laser scan to a set of reference scans. A certain number of outputs corresponding to the reference scans consist of the weighted hypotheses. The basic idea of the APR algorithm is to find the relation between two sets of characteristic 2D coordinates which are defined as anchor points. There are three types of anchor points: jump edge, angle, and virtual edge. The APR selects the best match by comparing fully connected graphs built by the sets of anchor points. Then using

the pair of matched sets, an alignment step finds a coordinate transformation that aligns one of the reference scans with the current scan.

The APR works well for most cases. However, if the number of anchor points is large, the alignment step is expensive in terms of calculation cost.

Combined Scan Matcher [78]

The Combined Scan Matcher (CSM) is a combination of the Cox matching approach and the IDC matching method. There are two matching algorithms in CSM. One is points-to-segments assignment from a modified Cox method. CSM extracts line segments from the reference scan and uses them as #1 prior model [78]. The second is points-to-points assignment from the IDC method. CSM examines the line segments in the current scan. If there are enough scan points lying on line segments, then the extended Cox algorithm is used; otherwise the extended IDC method is used.

Kalman method [79]

The Kalman method generates good pose estimation by using the information from the scanning data obtained while detecting an object [79].

Let $X = [x, y, \theta]_k^T$ represent the pose of the mobile robot at time k. It's the state vector of the robot. Let z_k represent the measurements from reference sensors (could be odometry, INS, GPS). It's a deterministic input. Then the prediction of the pose at time $k+1$ is:

$$X_{k+1} = X_k + z_k + w_k \tag{5-1}$$

where, w_k is the process noise.

The scan data from the laser sensor is processed by data filtering and creates a series of segments. Then an estimated pose is calculated.

All the data association methods introduced above has been applied for indoor environments. Outdoor application has to deal with highly irregular and non-static environments. For example, scans of plants result in unexpected reading because of the irregular edges. Also, the appearances of dynamic or unknown objects lead to significant changes in the corresponding scans.

5.3 Scan Data Processing

For laser sensor data, there are two features of interest: break points detection and line segmentation. These features give full information of the environment. The whole procedure of laser sensor data processing is divided into three steps: scanning data pre-processing, detection of break points, and extraction of line segments.

5.3.1 Scanning data pre-processing

The sensor data after the ground and vegetation filtering may still include contact points from objects outside of the map boundary, when the robot is running close to the map boundary. At first, all the points beyond of the map boundary will be removed from the scanning points list. This is done easily by determining whether a point is inside the map boundary polygon.

Now, all the points left in the points list are from the real objects (visible to LMS). Some of them belong to map objects, while other belongs to non map objects.

5.3.2 Detection of break points

Break points are scan points responding to discontinuities in a whole scanning. These discontinuities may results from two different factors:

- New objects were contacted in the direction of laser ray;
- The angle of the scanned surface changed due to the appearance of a new surface.

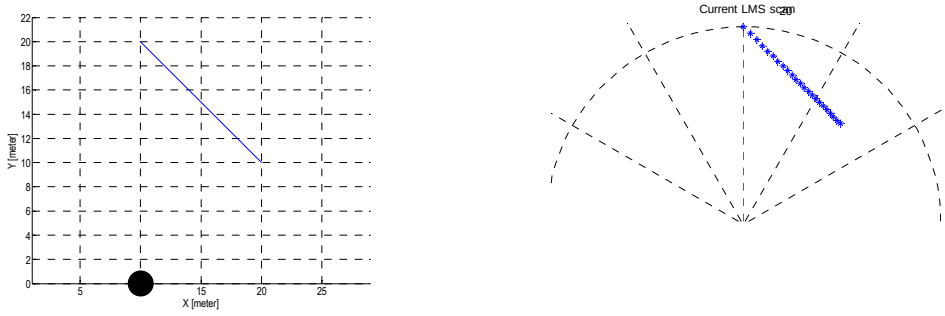


Figure 5.1 Scanning points from a single smooth surface [107]

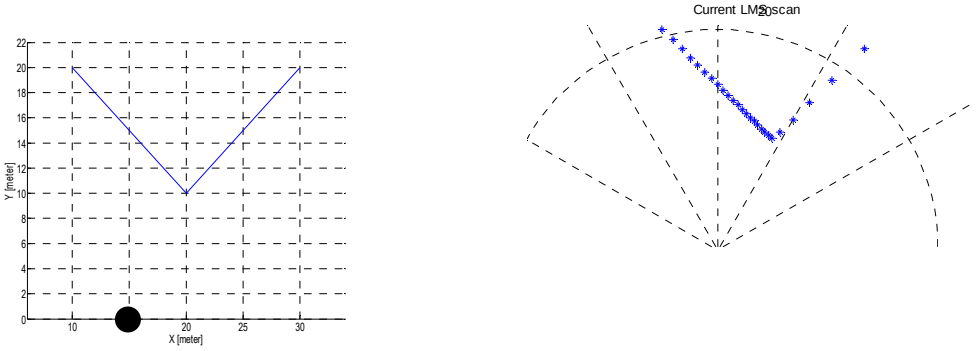


Figure 5.2 Scanning points on two smooth surfaces [107]

According to the characteristics of the laser range sensor and the setting parameters, an algorithm has been developed to classify all the points by the objects to which they belong. The adjacent points are from adjacent laser rays. The angle between the two rays is 0.5° by the configuration. The distance between two adjacent points is dependent on different factors. If the contact surface is a continual smooth surface, the distance is

dependent on the contact range and the angle between the laser rays and the contact surface (see Figure 5.1). If the adjacent points are on different surfaces from the same object or on different objects, the distance is dependent on the ranges and the angles between the laser rays and the surfaces (see Figure 5.2 and Figure 5.3).

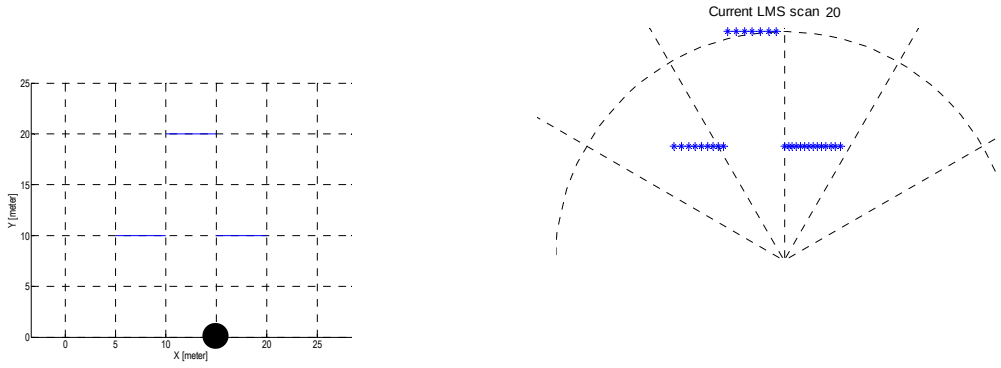


Figure 5.3 Scanning points on different surfaces [107]

The scanning points Figure 5.1 to Figure 5.3 are from ideal cases. For real application, the points from one smooth surface always have departure from the surface. To extract segments from these points, three steps have to be applied: separating points by objects, separating points by segments, and fitting points to segments.

The break point detection could be done by choosing all points which have a small distance between the adjacent points. If the distance between two adjacent points is greater than a given threshold – those points are called break points. All the points before the break point are known as belonging to one object. This procedure is repeated until all points are split to different objects. This is the simplest break points detection method [80]. The break point condition may be represented as:

$$\left\| p_n - p_{n-1} \right\| = \sqrt{(x_n - x_{n-1})^2 + (y_n - y_{n-1})^2} > D_{\max} \tag{5-2}$$

Where, $\|p_n - p_{n-1}\|$ is the Euclidean distance between point $p_{n-1}(x_{n-1}, y_{n-1})$ and point $p_n(x_n, y_n)$. $D_{\max}$ is the constant threshold.

However, such a processing method will result in a situation where the points of one object are divided into two different objects (demonstrated as Figure 5.4). This could happen when the laser scanning ray is almost parallel to the scanned surface. An adaptive threshold may be used to avoid this situation.

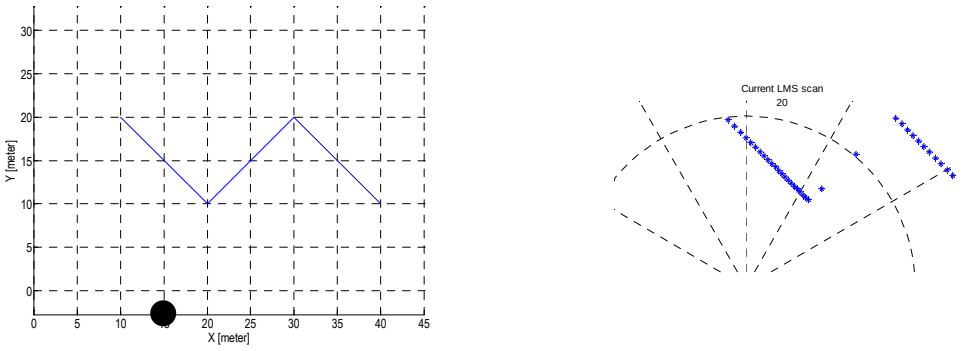


Figure 5.4 Failed points collection from one object [107]

For finding an adaptive threshold, two factors have to be considered - angular step and range. For the laser range finder, the angular step is constant. So, the threshold will just relate to the range. Here, a methodology in [81] is used to determine $D_{\max}$ according to the range. Figure 5.5 is the demonstration of this methodology. The following description about this methodology is from [81]:

To determine if there is a break between point $p_{n-1}(x_{n-1}, y_{n-1})$ and point $p_n(x_n, y_n)$, a virtual line is defined. This line passes through the point $p_{n-1}(x_{n-1}, y_{n-1})$ and has an angle ψ with the scanning direction of point p_{n-1}. It represents that a line may be reliably

detected under an uttermost case. The aim of this line is to find the farthest range point p_n [81].

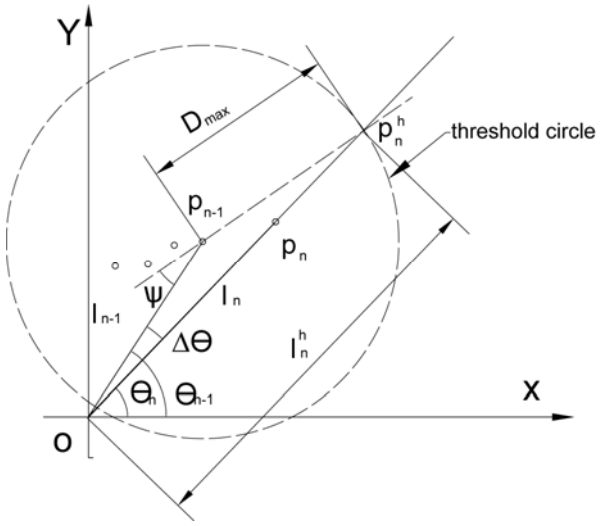


Figure 5.5 Demonstration of algorithm for break point detection [81]

Assuming there is a hypothetical point p_n^h on this virtual line. The range of this hypothetical point is l_n^h. The following equation represents the relation between l_n^h and l_{n-1}:

$$l_{n-1}\sin(\psi) = l_n^h \sin(\psi - \Delta\theta) \tag{5-3}$$

Here, $\Delta\theta$ is the angular step of the laser sensor. l_{n-1} is the range of point p_{n-1}.

For the arbitrary triangle $OP_{n-1}P_n^h$ in Figure 5.5, using the trigonometry equation:

$$\left\| p_n^h - p_{n-1} \right\| = l_{n-1} \frac{\sin(\Delta\theta)}{\sin(\psi - \Delta\theta)} \tag{5-4}$$

Then, $\left\| p_n^h - p_{n-1} \right\|$ may be used as the threshold for point p_{n-1} and point p_n. Furthermore, using the 3σ rule to add the sensor noise to the threshold estimate and Equation 5-4 is changed to:

$$D_{\max} = \left\| p_n^h - p_{n-1} \right\| + 3\sigma_r = l_{n-1} \frac{\sin(\Delta\theta)}{\sin(\psi - \Delta\theta)} + 3\sigma_r \quad (5\text{-}5)$$

where, σ_r is the standard deviation of Gaussian noise in range measurements and $D_{\max}$ is the threshold of break point detection.

If the point p_n is outside of the circle in Figure 5.5 (with the radius equal to $D_{\max}$), point p_n and point p_{n-1} both are regarded as break points. The parameter σ_r is included in the manufacture's manual of the laser sensor. The angle ψ may be estimated by experiment or experience.

5.3.3 Line extraction

After the detection of the break points, all the scanning points are divided into different groups. Points in one group belong to the same object. The points in one group will be transferred to lines in this step. Line segments are supported by sets of points which are ordered consecutively according to the acquisition scanning process.

Line extraction step handles three problems: find all line segments in a set of points, divide the points to different sub-group by line segment, fit the points in one sub-group to a line [81, 82].

The line extraction used here is adapted from the Split and Merge algorithm. The differences from the classical Split and merge method are in two sub steps:

- The time of the regression calculation is reduced by using intuitive line instead of Least Squares Line Fitting before fitting the final lines.
- The corner/turning points are processed in a special way.

The algorithm is explained below. First, a simple intuitive line is created by connecting the two end points from a group of points which are supposed to be from the same object. The distances from each point to the created line segment are compared with each other. If the largest distance is bigger than a given value (it could be estimated according to the environment), the point which has the largest distance to the line is regarded as a turning point. All points after this point (excluding this point) are marked as a new group and stored into the buffer. A new line segment is created using all points before this point (also excluding this point). The procedure is repeated until there is no turning point left. Then, take the newest point group from the buffer and process these points using the same procedure until the buffer is empty. Figure 5.6 shows the recursive algorithm flowchart.

After all turning points have been found, all points have been divided into groups by potential segments. The Least Squares Line Fitting (LSLF) method is used to fit line segments to the points. LSLF is also called regression line fitting.

Given the n data points $(x_1, y_1),\ (x_2, y_2),\ \cdots,\ (x_n, y_n)$, LSLF determines a best (in least square sense) line $y = ax + b$ that fits the points. The coefficients a and b are given by:

$$a = \frac{(\sum_{k=1}^{n} x_k)\sum_{k=1}^{n} y_k - n\sum_{k=1}^{n} x_k y_k}{(\sum_{k=1}^{n} x_k)^2 - n\sum_{k=1}^{n} x_k^2} \tag{5-6}$$

$$b = \frac{(\sum_{k=1}^{n} x_k)\sum_{k=1}^{n} x_k y_k - \sum_{k=1}^{n} x_k^2 \sum_{k=1}^{n} y_k}{(\sum_{k=1}^{n} x_k)^2 - n\sum_{k=1}^{n} x_k^2} \tag{5-7}$$

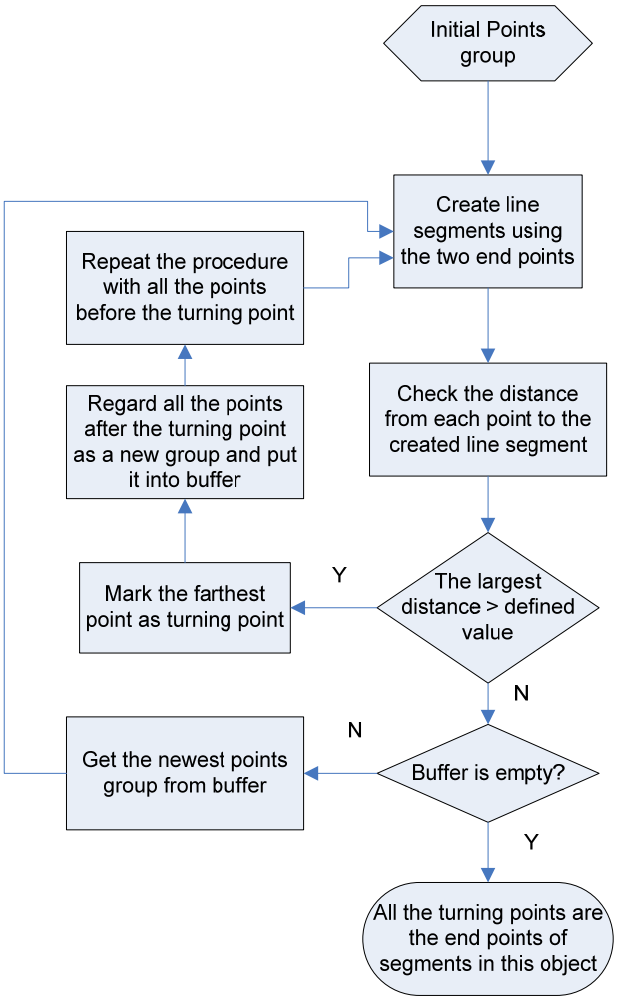


Figure 5.6 Diagram of recursive segments turning detection [107]

In this implementation, the points groups for one line segment must contain at least 2 points. The turning points are different from the normal points. All the turning points are supposed to be the intersection of two edges. However, for laser scanning, the ray may miss the corner of the objects. So, the turning point may belong to only one edge instead of two edges. Figure 5.7 shows the case. There are 12 points from the two perpendicular edges A and B. Using the algorithm in step 2, point 7 is chosen as the turning point. The point group for edge A has 7 points (point $1,2,3,4,5,6,7$) and the point group for edge

B has 6 points (points $7,8,9,10,11,12$). Obviously, point 7 would bring a big error to the fitting algorithm. Considering this, the ending points (turning points) from the previous step will not be counted as valid points.

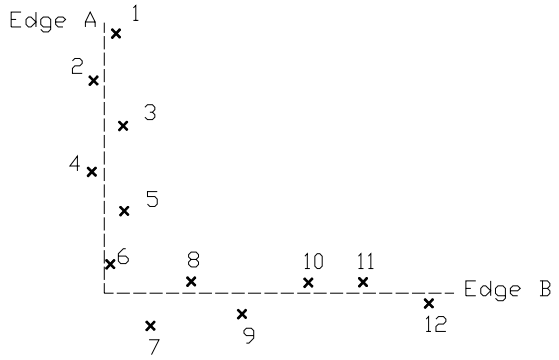


Figure 5.7 Laser ray missed the corner

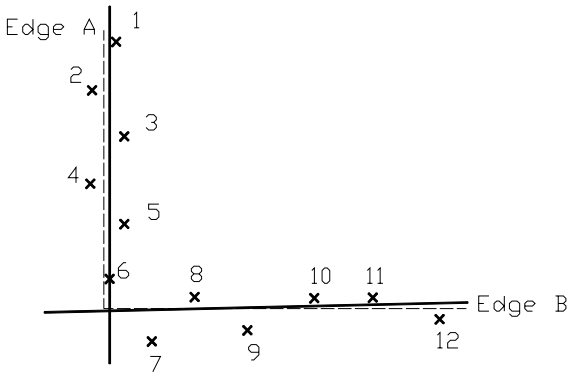


Figure 5.8 Line segment from fitting

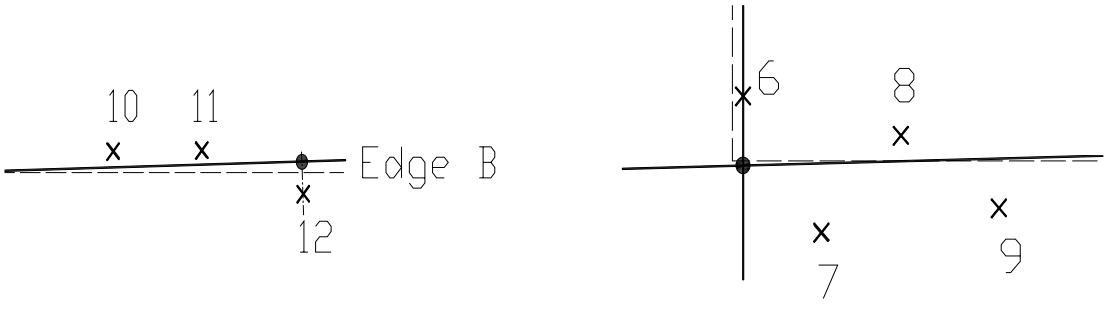


Figure 5.9 Estimate of ending points

```
Given : a set of point s which has N_0 point s, P_0 : {p_0, p_1, ... p_{N_0}},
Define : Hanoi = 0, Hanoi_index[max_number_segments][2], seg_count = 0;
         seg_index[max_number_segments][2], start = 1, end = N_0;
Call function ITERATIVE_SPLIT(P_0, start, end, seg_count)
{
   do{
      for i = start : end
     {   P_temp = {p_start, ..., p_end};}
      if (turning point existed at p_m from p_start to p_end)
      % has turning point, process current one and store the second into stack
      {   Hanoi_index[Hanoi][] = {start + m, end};
          Hanoi++;
          call function ITERATIVE_SPLIT(P_0, start, start + m, seg_count);}
       else %no turning point, take the newest stored portion from stack
       {   Hanoi--;
          if (seg_count = 0)
          {   seg_index[Hanoi][1] = start;}
          else
          {   seg_index[Hanoi][1] = start + 1;}
          if (Hanoi < 0)
          {   seg_index[Hanoi][2] = end;  }
          else
          {   seg_index[Hanoi][2] = end - 1;}
          if (seg_index[seg_count][2] - seg_index[seg_count][2] >= 2)
          % at least 3 point s for one segment;
          {   seg_count++;  }
          if (Hanoi >= 0)
          {  call function ITERATIVE_SPLIT(P_0, Hanoi_index[Hanoi][1], ...
                                           Hanoi_index[Hanoi][1], seg_count);}
      }
   }while(Hanoi > 0);
}
```

Figure 5.10 Pseudocode of iterative segment extraction algorithm

The created lines are straight lines without ending points (Figure 5.8). The new ending points need to be calculated again to get line segments. If the ending point is the first point of the first segment of one object, or the ending point is the last point of the last segment of one object, the new ending point is the intersection of the fitting line and a line perpendicular to the fitting line and through the old ending point (Figure 5.9, left). Otherwise, the turning points will be replaced with the intersection of adjacent fitting lines (Figure 5.9, right). This algorithm is represented with pseudocode in Figure 5.10. The Matlab code of the algorithm is listed in Appendix F.

5.4 Map data extraction using laser sensor simulator

There are many implementations that use a prior map to assist navigation for indoor mobile robot systems. Most of these methods detect the current sensor data then compare it with the whole map to carry out localization, mapping, or SLAM tasks [25, 83, 75, 76, 78, 77, 79]. For outdoor applications, the map may be huge and complex. If the whole map has to be loaded and the calculation is against all the objects in a map, the time consumption and memory cost may be not affordable for real-time robot systems.

If the knowledge about how to choose a partial map and some objects related to the robot's current position is available, a smaller map and fewer map objects are enough to achieve the necessary calculations related to localization and mapping tasks.

For this work, an elementary localization component which uses the GPS, odometry and INS sensors is supposed to be available and provides the information about the robot's

pose. An environmental map is also built and loaded into the system. To overcome the uncertainty and the potential lack of GPS signal, a second localization method is needed to enhance the pose estimation and be a backup localization unit.

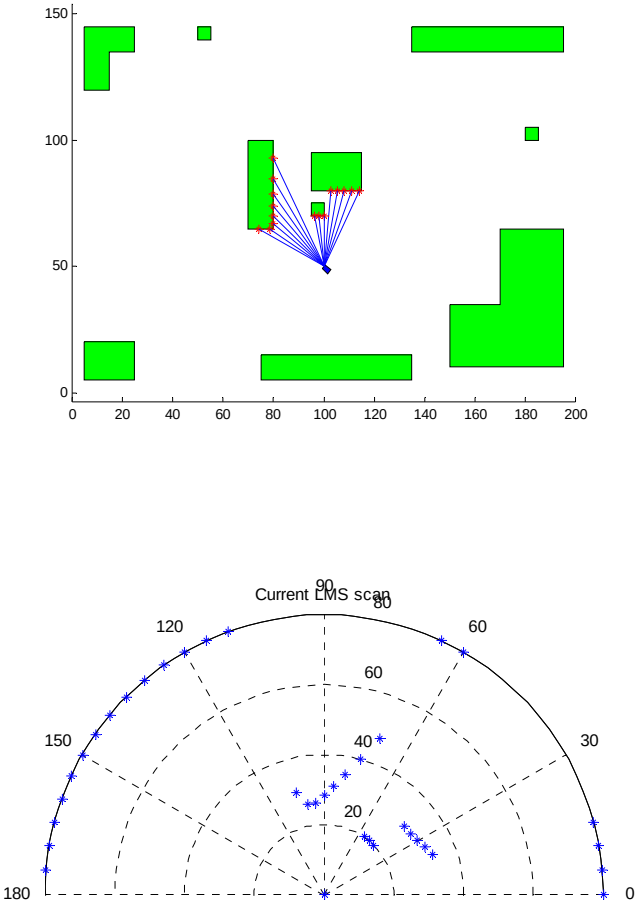


Figure 5.11 Laser sensor simulator (unit: meter and degree) [107]

The laser range finder is the main sensor used for this 'localization enhancement'. A simulator of the laser range finder is developed to help in choosing necessary map objects and partial of map. The simulator imitates the functions of a real laser range sensor. The parameters, like resolution, maximum range, and noise level, could be adjusted to

simulate the configuration of a real laser sensor. According to the robot's pose which could be from GPS/INS, robot station, or previous localization step, the simulator clips the whole map with a window and does the scanning against map objects inside of the clipping window. All segments from the simulator are returned. Then, these segments are used as a reference map for localization and mapping tasks. Figure 5.11 shows the simulation result.

The simulator is very helpful for the matching next described next. However, the computation cost for the simulator itself is high. Considering the map in Figure 5.11 (upper), there are 10 map objects with 44 segments totally. If the resolution of scanning is $0.5°$, there are 361 simulated laser rays. To find all the contact points, each laser ray line needs to do the intersection detection with all the segment lines.

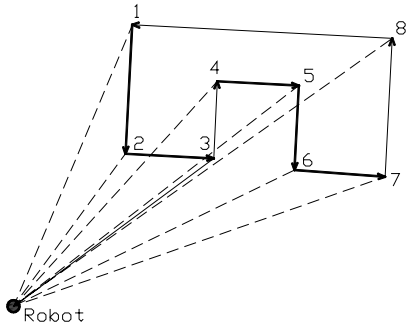


Figure 5.12 Visibility of polygon [107]

To reduce the calculation cost, some simplifications are introduced. First, as the description above, not all the map objects are necessary for the current check. A small rectangle is used to clip the whole map. Only the objects which are inside of the rectangle or have intersection with the rectangle will be checked. Then, visibility check is applied to the segments. All the map polygons consist of segments with direction (clockwise definition in here). In an example shown in Figure 5.12, the polygon has 8

segments $1 \to 2$, $2 \to 3$, $3 \to 4$, $4 \to 5$, $5 \to 6$, $6 \to 7$, $7 \to 8$, $8 \to 1$. A triangle is created by the robot point, the start of the segment, and the end of the segment. If the direction of the triangle edges is clockwise, the segment is visible from this robot point. Otherwise, the segment is invisible from this point. For the case in Figure 5.12, there are 5 segments $1 \to 2$, $2 \to 3$, $4 \to 5$, $5 \to 6$, $6 \to 7$ which are visible from the robot. The third simplification is on the scanning rays. A full scan's angle range is from 0° to 180° with 361 laser rays (under 0.5° resolution). Let α_1 be the angle of the line including the robot point and the start of the segment. Let α_2 be the angle of the line including the robot point and the end of the segment. These laser rays with angles which are within $[\alpha_1, \alpha_2]$ need to be checked.

After the simplifications above, the calculation cost is reduced obviously. The estimate in Table 5.1 shows the difference from the original simulation calculation cost and the ones after applying the simplifications for the scan against the objects in Figure 5.11. The Matlab code of a laser sensor simulator is listed in Appendix F.

Table 5.1 The difference of simulation calculation times [107]

	Number of segments	Number of used laser lines	Number of calculation times
No simplifications	44	361	15884
Using map clip(cont.)	12	361	4332
Using visibility check (cont.)	4	361	1444
Using angle limit	4	50(average)	200

5.5 Scan Data Matching and Precise Localization

This section describes the data matching step. The term, data matching in our context means determining the relative translation and rotation between two robot poses by comparing the two sets of sensing data which are collected from the two poses [83]. After the breakpoint detection, line extraction and map scan steps described in previous sections, two set of scan data represented by segments have been created. The two sets of scan data, actually the segments, are from map scan (the reference scan) and the current pose scanning (real scan). The data matching algorithm estimates the offset (translation and rotation) between the two sets and represents it as offset in position (dx, dy) and offset in orientation $(d\theta)$. Then after applying the translation and rotation of $(dx, dy, d\theta)$, the new map scan data from the updated pose $(x + dx, y + dy, \theta + d\theta)$ should best overlap the current scan data set. Finally, $(dx, dy, d\theta)$ could be used to update the mobile robot's position.

In this application, the reference scan to be used for the data matching process is taken from the simulated scanning against the prior map as discussed in the previous section. The pose used for the simulated scanning is taken from the elementary localization component (which fuses the data from GPS, odometry, and INS sensors). The current scan data is taken by a laser range finder sensor at an unknown robot's pose. The goal is to find the precise current pose using data matching algorithm.

5.5.1 Data matching used in this experiment

From the discussion in previous sections, the points from the scanning have been processed and converted to segments with object ID. A similar procedure was also applied to map objects. However, the map scan data does not need the break points detection and line extraction steps. The line segments from map can be determined directly from the simulated laser scanner. Then, two sets of segments (map segments set and scanning segments set) will be compared.

For the real application, there are unknown objects appeared to laser sensors. Also, the sensor data may still hold some noisy points which could be converted to segments. Another possible situation is that the sensor simulator scans the objects different from those scanned by the real sensor due to the error of the robot's pose. So there is no guarantee that two sets of segments always have the same number of segments.

The data matching method used in this application is a segments-to-segment matching method. As an example, Figure 5.13 and Figure 5.14 show the raw scanning points from the real laser scan data and the simulated scanning of the map objects. To watch the same style data, the map scan also uses the points instead of the segments in Figure 5.13.

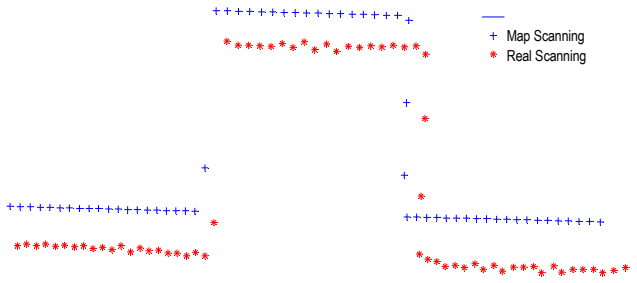


Figure 5.13 Scanning points from real and simulation [107]

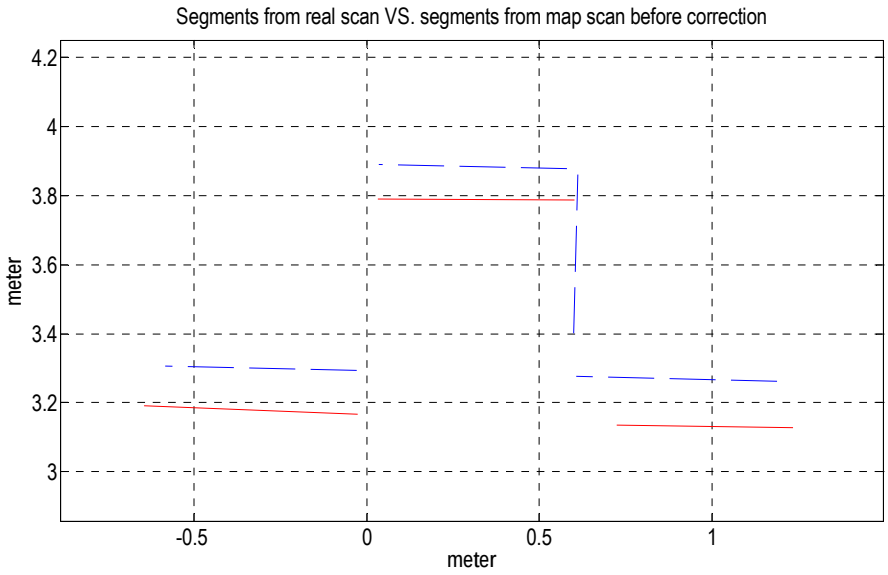


Figure 5.14 Segments from real and simulation scanning [107]

The matching procedure is described as follow:

Considering the possible maximum error in translation for the application, a value may be defined which presents the maximum error between the real pose and the estimated pose. Each segment from the map scanning creates a rectangle around itself by using this value as a buffer. In Figure 5.15, there are 4 line segments from map scanning. Four rectangles are created from these segments.

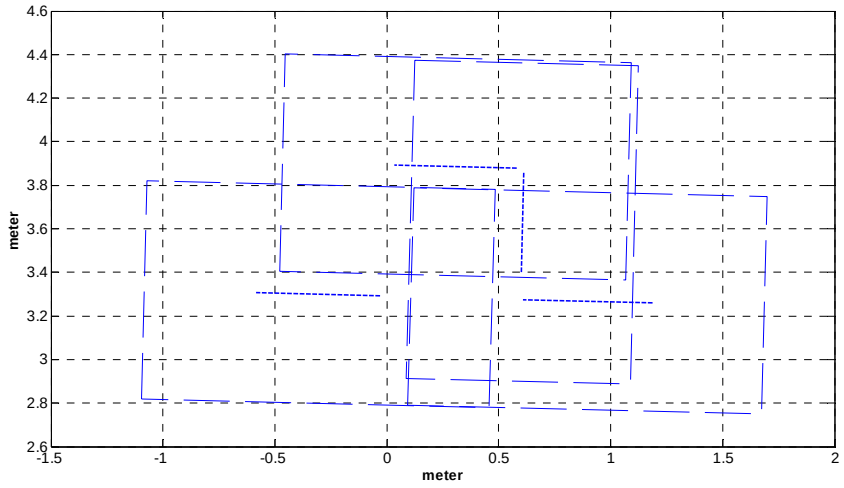


Figure 5.15 Buffering rectangles from the map scanning segments [107]

For each segment from the real scan, two conditions are checked:

1. Is the segment inside of one rectangle?

2. Is the slope difference from this segment and the map scanning segment which creates this rectangle smaller than a threshold?

If the two conditions are met, the two segments are considered as a pair of segments. Figure 5.16 shows this step.

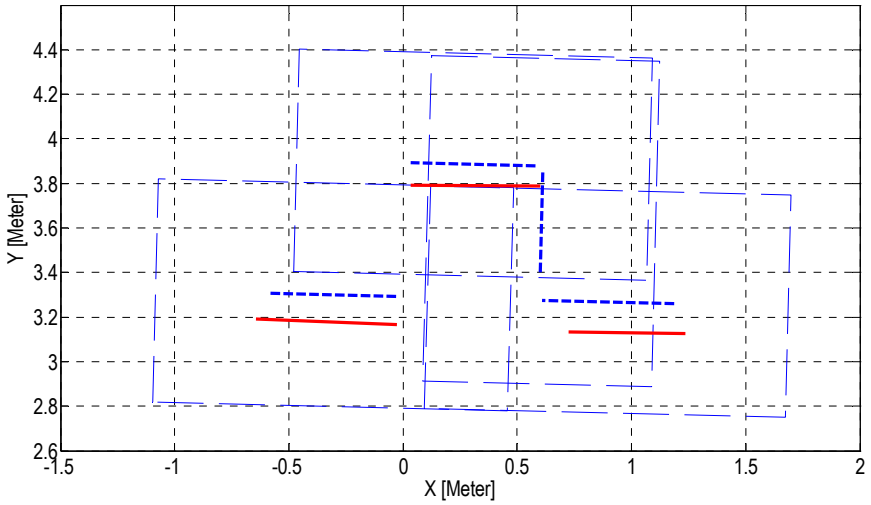


Figure 5.16 Segments pairs [107]

For each pair of segments, the slope difference is:

$$\Delta\theta_i = \theta_i - \theta_i^{'} \tag{5-8}$$

where, θ_i is the slope of the i^{th} segment from real scanning. $\theta_i^{'}$ is the slope of the corresponding segment in the pair.

So, the average slope difference for all the pairs is:

$$\Delta\overline{\theta} = \frac{\sum_{i=1}^{N}(\theta_i - \theta_i^{'})}{N} \tag{5-9}$$

where, N is the number of segments pairs.

At first, all the segments from map scanning are rotated with $\Delta\bar{\theta}$. The result is a new set of map scanning segments. For each segments in this new set, the displacement from the center of the map scanning segment to the corresponding real scanning segment is calculated as:

$$\begin{aligned}\Delta x_i &= x_i - (x_i^{'}\cos(\Delta\bar{\theta}) - y_i^{'}\sin(\Delta\bar{\theta})) \\ \Delta y_i &= y_i - (x_i^{'}\sin(\Delta\bar{\theta}) + y_i^{'}\cos(\Delta\bar{\theta}))\end{aligned} \tag{5-10}$$

Where, (x_i, y_i) is the center of the i^{th} real scanning segment. $(x_i^{'}, y_i^{'})$ is the center of the corresponding segment from map scanning.

Similarly, the average displacement for all the pairs is:

$$\Delta x = \frac{\sum_{i=1}^{N}\Delta x_i}{N}, \Delta y = \frac{\sum_{i=1}^{N}\Delta y_i}{N} \tag{5-11}$$

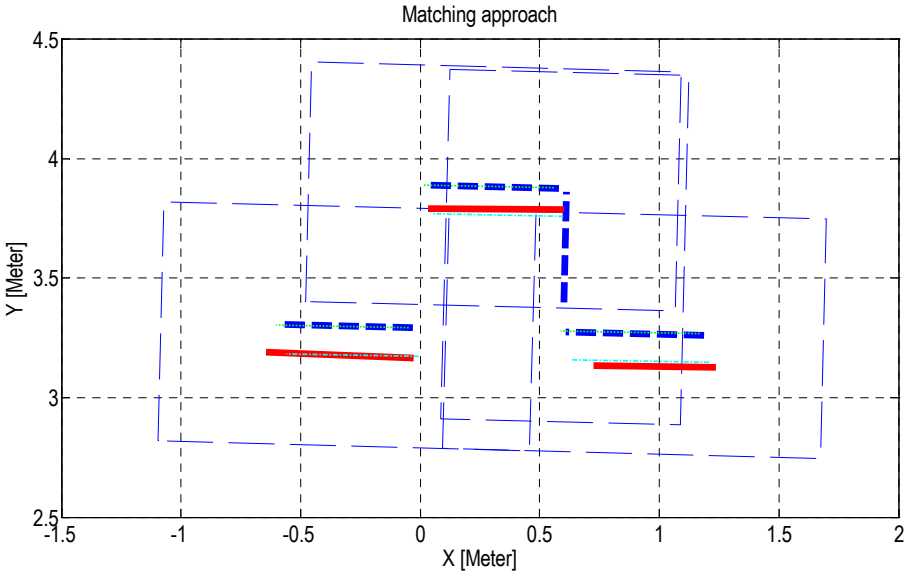


Figure 5.17 Matching approach [107]

The above matching approach steps is represented in Figure 5.17. At first, the calculated average rotation angle $\Delta\bar{\theta}$ is applied to all the segments from the map scanning (dotted

line). Then, the calculated average translation $(\Delta x, \Delta y)$ is applied (dot-dash line). The calculated $\Delta\overline{\theta}$ and $(\Delta x, \Delta y)$ are the estimated pose errors.

5.5.2 Result evaluation

After the matching step, the new set of segments from map scanning is assumed to be matching the real scan segments. At least, the errors on rotation and translation are expected to be much smaller than the original one.

To verify and estimate the quality of matching, the simulation of the map scanning function is applied again. The calculated rotation error $\Delta\overline{\theta}$ and the translation error $(\Delta x, \Delta y)$ are converted to the global coordinate system (described in the previous section). The original vehicle pose $[x, y, \theta]$ is updated to $[\hat{x}, \hat{y}, \hat{\theta}]$ by adding a global pose error $[\Delta x, \Delta y, \Delta\theta]$. The simulator scans the map at the new estimated pose. Then similar steps are applied to the new map scanning and the real scanning (it is the same scan as the previous section). To validate the matching result, two conditions have to be satisfied:

1. The error on angle from the evaluation step should be smaller than the error on angle from the matching step. It also should be smaller than an predefined angle error threshold;
2. The error on displacement from the evaluation step should be smaller than the error on displacement from the matching step. And it also should be smaller than a predefined displacement threshold.

5.5.3 Improvement of elementary localization

If the result of scan data matching meets the evaluation requirements, the estimated pose error is trustable. Then it will be applied to the current pose estimation $[x, y, \theta]$ which is calculated by the elementary localization component. The improved pose estimation becomes:

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\theta} \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \overline{\theta} \end{bmatrix} \tag{5-12}$$

In outdoor mobile robot applications, the absolutely correct pose estimation can not be obtained. All the existing effort is to reduce the error between the pose estimation and the true pose. Also, the judgment how accurate is the pose estimation is hard to be regulated. The goal of the localization enhancement method is to meet the localization and navigation requirement from the applications. Experiments described in the following section show that this method improves the accuracy of pose estimation to the level of $\sim 1.5\,cm$ on position and $\sim 1°$ on orientation. That is accurate enough to most outdoor mobile robots applications.

In this mobile robot system, the elementary localization component runs at a high frequency. The localization enhancement method works at a much lower frequency. This ensures that the system always has continued pose estimation from GPS/INS and the corrections of these estimations are added in a defined period. If GPS signal is not available or the elementary localization component fails, the enhanced localization function can supply the pose estimation by using the previous pose estimation.

The procedure of data matching method is represented with pseudocode in Figure 5.18. The Matlab code of matching procedure is listed in Appendix F.

Give : *a set of segments from real scan data*, $S:\{s_1, s_2, ...s_{n1}\}$;
a set of segments from real scan data, $M:\{m_1, m_2, ...m_{n2}\}$;
define : *angle threshold* λ; *buffer size a and b*;
segment _ pairs P; *number of pairs* $n3 = 0$;

create buffer box for all m_i *in* $M, B:\{b_1, b_2, ...b_{n2}\}$;
% *find the pair from S and M*
for $i = 1:n2$
{ *for* $j = 1:n1$
{ *if* (s_j *inside of* b_i & *angle*$(s_j, m_i) < \lambda$)
{ *add* (s_j, m_i) *to P*;
$n3++$;
remove s_j *from S*;
remove m_i *from M*;
remove b_i *from B* } } }
% *calculate the average heading offset*
$\Delta\bar{\theta} = \sum_{i=1}^{n3} (angle(s_i) - angle(m_i)) / n3;$
% *apply the angle offset to*
rotate $\{m_1, m_2, ...m_{n3}\}$ *with* $\Delta\bar{\theta} \rightarrow \{m_1', m_2', ...m_{n3}'\}$;
for $i = 1:n3$
{ *calculate* $(\Delta x_i, \Delta y_i)$ *at center of* $m_i', s_i\}$
% *calculate the average translation between* m_i' *and* s_i
$\Delta x = (\sum_{i=1}^{n3} \Delta x_i) / n3;$
$\Delta y = (\sum_{i=1}^{n3} \Delta y_i) / n3;$
output : $(\Delta x, \Delta y, \Delta\bar{\theta})$

Figure 5.18 Pseudocode of data matching algorithm

5.6 Experimental Results and Discussion

The developed localization enhancement method has been used to build two JAUS components which will be discussed in the next chapter. The mobile robot system running these components based on the developed method works well on all its outdoor tasks.

The experiment is based on the mobile robot GRUNT (See Figure 2.1) equipped with a 2-D laser scanner SICK221. The mobile robot moves around the outdoor test field and collects range data. The laser sensor scans with a frequency at $5\,Hz$. There is a pre-built map with 7 line segments. The test field has a rough surface which is covered by snow. There are some moving objects in the field (other vehicle, human, etc.).

To test the localization algorithm independently, all other sensors except the laser range sensor and GPS/INS are switched off or ignored. The robot started the motion from a roughly known pose. Figure 5.19 shows the map with a 'closed door like' object and the initial pose of the vehicle (one of the testing cases).

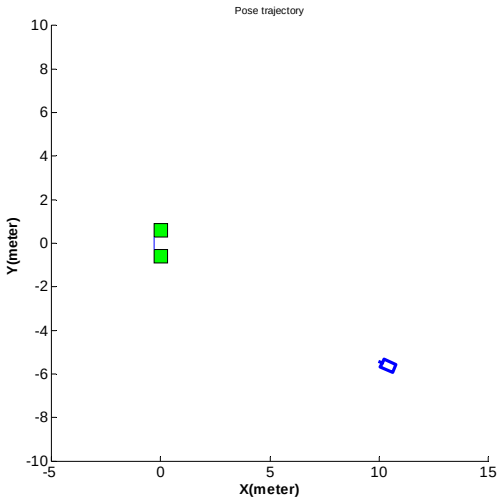


Figure 5.19 Map and the initial pose of the robot [107]

Figure 5.20 is a sample from the experimental data sequence. It presents the original scanning data from the real scanning (star mark) and the map scanning (plus mark).

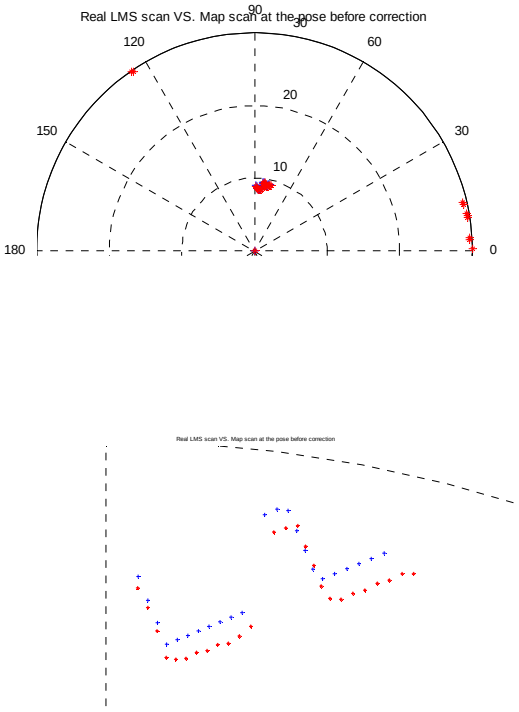


Figure 5.20 Scanning point before the pose correction (from experiment) [107]

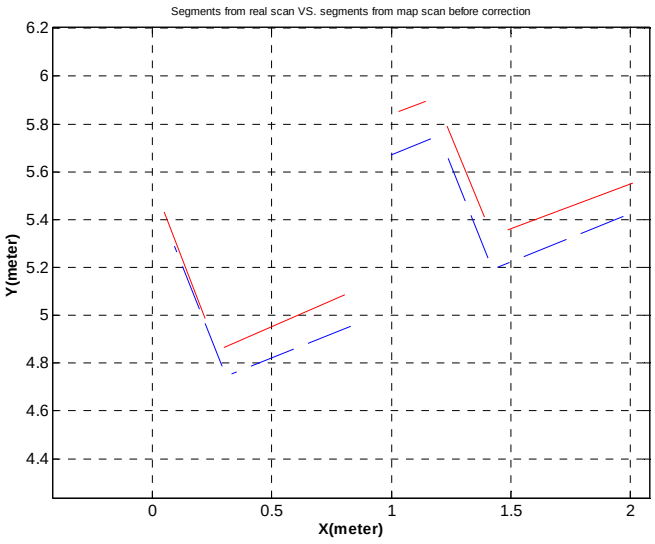


Figure 5.21 Segmentation of scanning points before the pose correction [107]

Figure 5.21 gives the result of segmentation of the original points. Solid lines are segments from real scanning and the dash lines are segments from map scanning.

After the matching step, the calculated $\Delta\overline{\theta}$ and $(\Delta x, \Delta y)$ are converted to the global coordinate system and applied to correct the robot's pose. Figure 5.22 shows the new map scanning points and the same real scanning points.

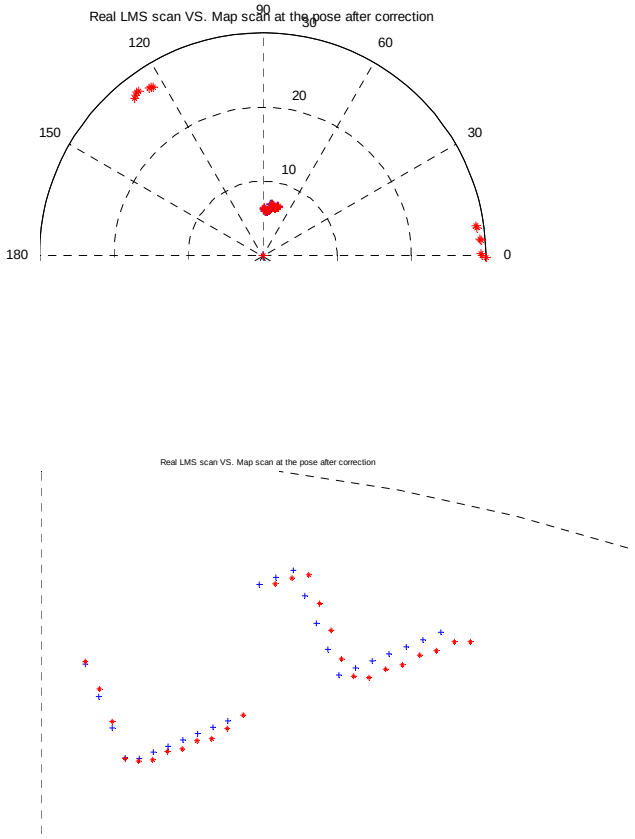


Figure 5.22 Scanning points after the pose correction [107]

Figure 5.23 gives the results of segmentation from the original scanning points and the renewed map scanning points. Solid lines are segments from real scanning and the dash lines are segments from the new map scanning.

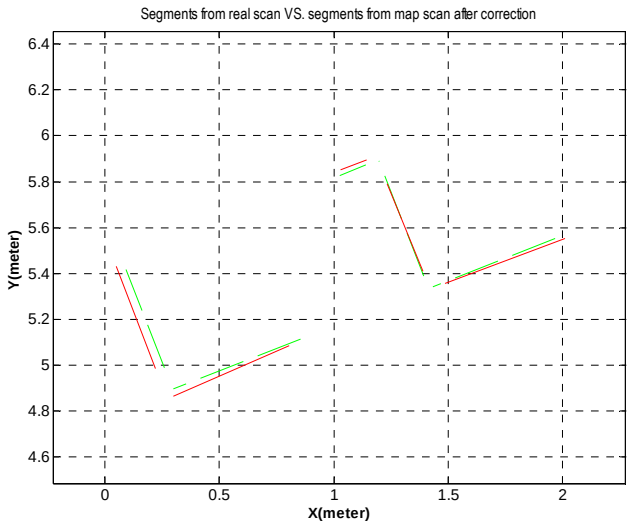


Figure 5.23 Segmentation of the points after the pose correction [107]

The error between the two sets still exists, but it's reduced significantly. The analysis of the whole data sequence shows that the errors in direction and translation of the vehicle have been reduced to around $1°$ and $1.5\ cm$.

5.7 Conclusion

In this chapter a geometry based data association method for outdoor mobile robot localization enhancement by using 2D laser range sensor is discussed. With a pre-built global map, a simulated laser scanner is developed to supply a reference scanning according to the reported pose of the vehicle. To reduce the computation complexity and time, three simplifications - clipping against the map, visibility check, and angle limitation - are used in the map simulated scanning. A set of algorithms is created to process the raw sensor data from the outdoor dynamic environment. The raw scanning

points are converted to line segments. Using a segment-to-segment matching procedure, the errors of the vehicle's heading and location are calculated from the pairs of matched segments. The validity of the calculated correction is also checked by the simulated scanning. When searching for the matched segment pairs, a buffering rectangle is applied to reduce the computation cost and eliminate the chance of wrong matching. The method is tested with simulation and experiments. Both of them prove that this method is very promising for outdoor application. The errors of localization in a large outdoor field could be limited to less than $1°$ in heading and $1.5\,cm$ in location.

The developed algorithm can be used for not only the localization enhancement but also for the detection of non-map objects and object tracking which will be addressed in the next chapter. A user defined JAUS component Range Based Pose (RBP) is designed using this algorithm to correct the error in pose estimation. Moreover, RBP can detect the non-map objects (obstacles). The design of RBP and the mechanism of obstacle detection are addressed in the next chapter.

Chapter 6

Development of Obstacle Detection and Intruder Tracking JAUS Components

6.1 Introduction

A world model JAUS component – Map Data Server has been developed and the algorithm for laser range data processing has also been created in the previous chapters. In this chapter, the discussion will be focused on the development of two JAUS components, which apply the developed algorithm of laser data association.

The mobile security robot needs to know where it is and what is around it. In some special tasks, the robot also needs to find suspicious objects and carry out some actions with respect to these objects. Two JAUS non-standard components, called Range Based Pose and Object Tracking, have been developed to handle these tasks.

The Range Based Pose (RBP) is a component which finds the error of the current pose estimation using laser sensing, detects non-map obstacles, and recognizes special non-map objects by size. The mechanism of finding pose estimation error has been addressed in chapter 5. In this chapter, the functionality of obstacle detection in RBP is described.

The Object Tracking (OT) is a component which tracks the special objects according the alarm from the RBP or from a higher level command. The strategy of how to use range sensor to detect and track the special objects are described in the sections that follow.

6.2 The strategy of obstacle detection

According to the requirements for this outdoor security application, the robot's tasks may be divided into two different cases:

Case_1: The robots patrol along a defined trajectory and ignore all objects which are not in their way. There is no intruder detection requirement.

Case_2: The robots patrol along a defined trajectory and detect possible intruders in their working space.

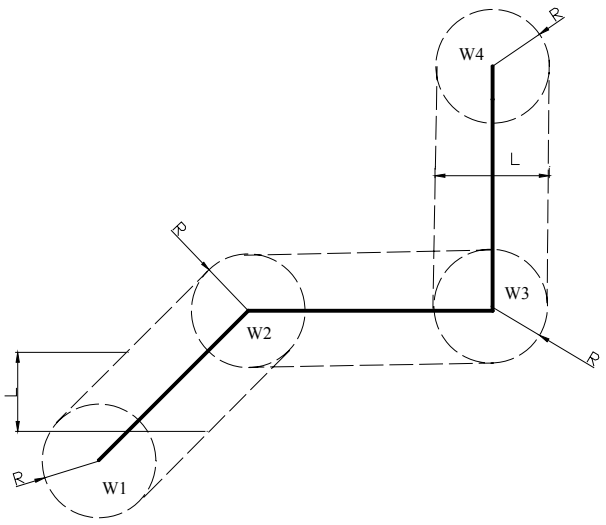


Figure 6.1 Path buffering definition in DARPA

6.2.1 Obstacle check

For both cases, if there are unknown objects in the robot's way, the robot needs to stop and query the path planning components for a new path. Some buffers are added to the path line (segment). If any objects intersect or are inside of these path buffers, these objects become obstacles. A new path planning is needed as soon as the obstacles appear.

Research shows varied definitions of path buffers. Figure 6.1 shows how the buffers are defined in DARPA [45]. For each waypoint, a buffering circle with radius R is defined. For each line, two lines parallel to the path and with a given distance L are defined. The buffer for the path is the area created by the circles and lines.

For the DARPA buffer definition, the judgment of intersection of the objects needs to consider the arc edges and line edges. With both line and arc edges, the buffering area is not a normal polygon.

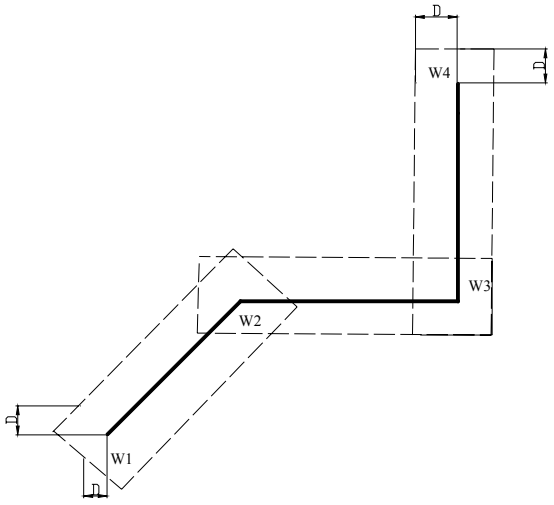


Figure 6.2 Path buffering definition in this thesis

To simplify the buffering area and intersection calculations, a path buffering which is different from the one from DARPA is defined (see Figure 6.2). A buffer rectangle is

added to each path section. The polygons intersection algorithm can be applied to decide if there is an obstacle in the way.

6.2.2 The strategy of obstacle detection

For obstacle detection purposes, the pose estimation, the scanning data from the laser range finding sensor and the prior map - which has been loaded into the vehicle - are required. The laser sensor simulator creates another set of scanning data, which is the ideal scanning against the prior map at the vehicle's estimated location. The scanning data are processed according to the algorithm described in chapter 5. As per the previous discussion, two sets of object segments are compared to calculate the error on pose estimation. On the other hand, the data association procedure also supplies information about these non-map objects. These non-map objects are checked against the planned trajectories and the obstacle list is created, if there are any objects in the path.

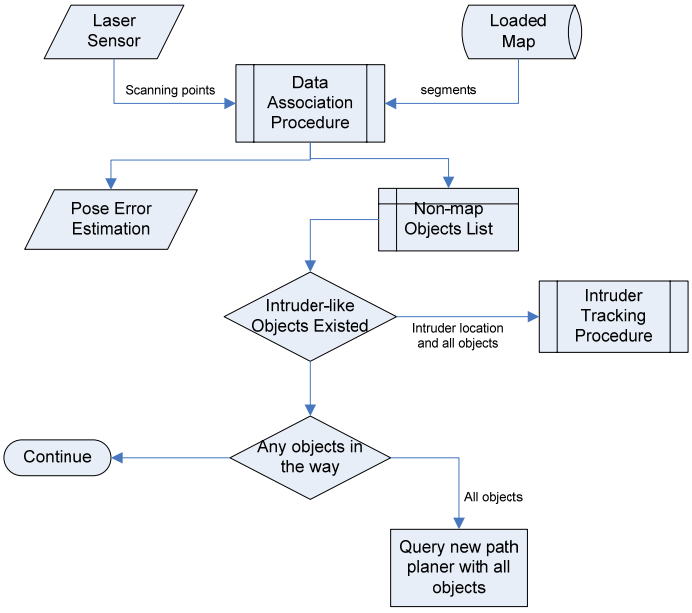


Figure 6.3 Strategy for obstacle detection

Currently, the intruder objects are similar in size to human beings.

The obstacle detection strategy is demonstrated in Figure 6.3.

6.3 Obstacle detection component and related messages

6.3.1 Component overview

The obstacle detection component in this system is called Range Based Pose (RBP). The RBP is one of the localization function components in the whole JAUS system. The RBP transfers the laser range sensor data to segments using some algorithms developed early, like adaptive breaking points detection and regression line creation (described in chapter 5). Any object with a single point will be ignored.

A laser range sensor simulator is used to simulate the physical sensor against the given map at a robot's current pose. The simulator is coded according to the principle of real sensor scanning. With the given pose and map, it scans from the given pose and returns a set of points like the real sensor. This set of points is used to refer to the real points set from the real sensor scanning.

Two sets of segments created from the real sensor and simulator are compared to find the corresponding pair. The segment-to-segment association algorithm is applied to estimate a robot's pose error.

All other segments from the sensor which could not find a corresponding pair from the map are marked as unknown objects. RBP can find the intruder-like objects from these unknown objects and select one or more intruders by the different task requirements. If

the tracking action is active, the unknown objects and intruder(s) are reported to other related components. The data association result is used to estimate the pose error and then update the pose measurement from elementary localization component.

For obstacles and intruder detection tasks, The RBP works in different procedures according to the requirements. When "following the intruder" is not required, the RBP reports all unknown objects to the path planner component if there are any unknown objects in the way. Also, The RBP notifies the possible intruder(s) to the high level component. If "following the intruder" is required, The RBP alarms the high level command component there are intruder, reports all non-map objects and sends the intruder information to object tracking component.

6.3.2 Description of the RBP component

The RBP component processes the sensor data from laser range finder, which is represented by points in latitude and longitude. RBP estimates the pose error by matching scanning data with map data. RBP detects possible intruder-like objects, if they exist.

In the RBP, the laser range data processing procedure includes the following steps:

1. All scanning points outside of the map perimeter are removed;
2. All points are divided into different groups by the different objects they belong to;
3. Segments are extracted from all the divided points according to the segments regression method.

The procedure of matching and detection processing includes the following steps:

1. Segments from the above steps are compared with the segments from the map. A pose error is estimated by the matching result;

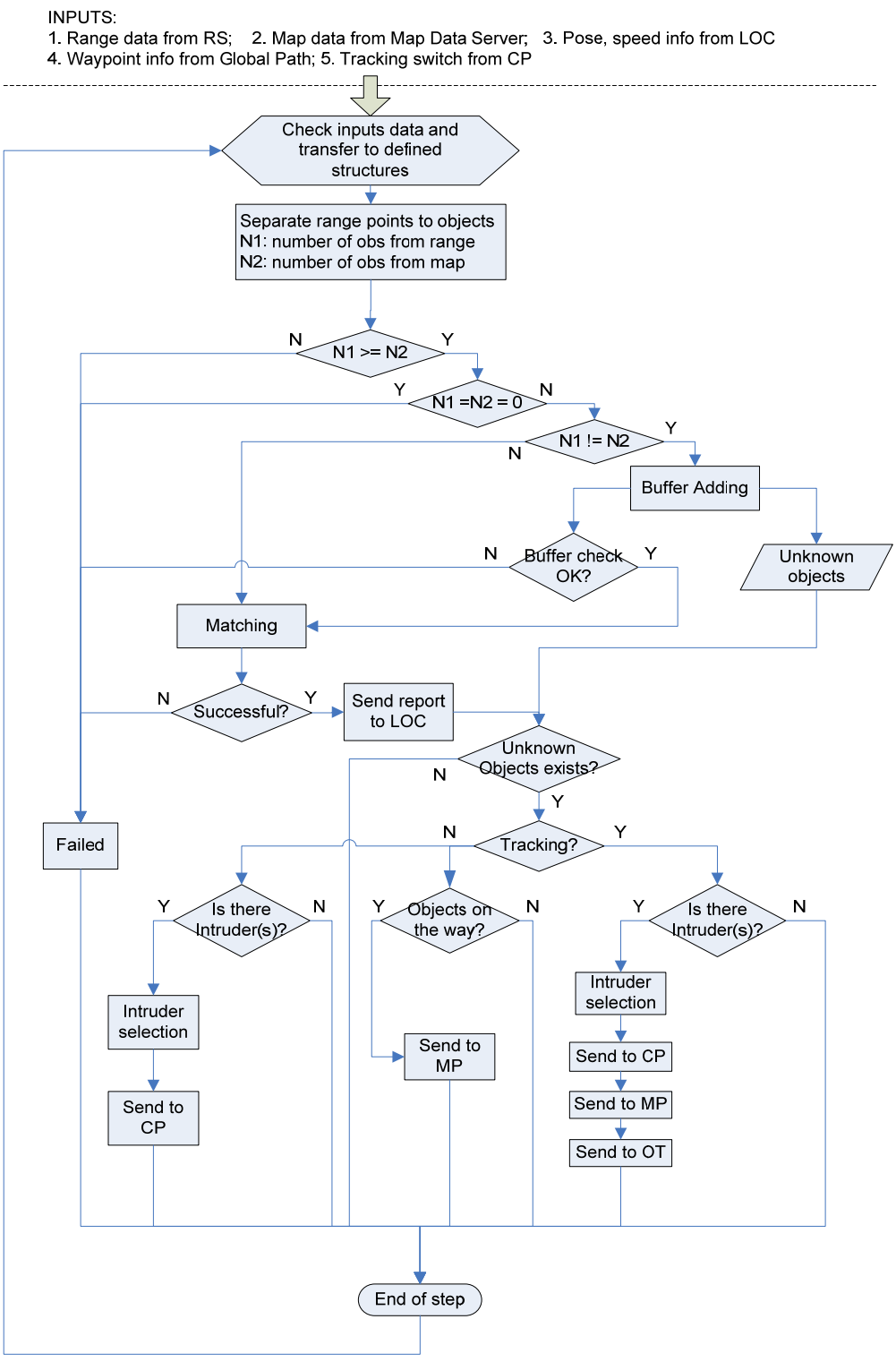


Figure 6.4 Flowchart of RBP component

2. All objects which do not belong to map objects are checked to find if they are human-like objects;
3. All non-map objects are checked to find out if they are in a robot's way (obstacles).

The responding reactions for these steps include:

- Reporting the estimated pose error to the Global Pose and Velocity Component;
- Reporting the possible intruders to the Command Process Component;
- Reporting the possible intruders to the Object Tracking Component;
- Reporting all non-map objects (including the intruders if the following behaviour is required; otherwise, excluding the intruders) to the Motion Planner Component, if there are obstacles (objects in the path).

Figure 6.4 shows the flowchart for RBP.

There are three coordinate systems used in the RBP, the global spherical system, the planar coordinate system, and the robot local coordinate system. The global coordinate system uses latitude and longitude in degree to present the location. The planar coordinate system uses north and east in meter as the x axis and y axis. The robot local coordinate system locates along with the robot. The heading direction of the robot is the direction of the y axis (See chapter 4 for details). All incoming and outgoing messages use the global coordinate system. The other two coordinate systems are used inside of the RBP.

6.3.3 Messages descriptions

The Range Based Pose (RBP) Component uses the standard JAUS message header structure. In addition to the existing JAUS core input and output message sets, (like

shutdown, standby, reset, etc.) the following messages for the map data query and the corresponding responses are currently supported:

Incoming messages:

- Report map capability;
- Report map data;
- Report global pose and velocity;
- Report range sensor;
- Set follow ability;
- Report element.

Sending messages:

- Query map capability;
- Query map data;
- Query range sensor;
- Query global pose and velocity;
- Query element;
- Set intruder found;
- Set tracking;
- Set obstacle;
- Set global pose.

The special messages of RBP are listed in Appendix D. The Matlab prototype is also listed in Appendix F.

6.4 The strategy of intruder tracking

As described in the previous section, there are two cases of object detection and tracking in the security robot application. One is “single detection” and the other is “detection and tracking”. The single detection task is carried out by the Range Based Pose component. For detection and tracking tasks, two JAUS components, the Range Based Pose and the Object Tracking (OT), are used together.

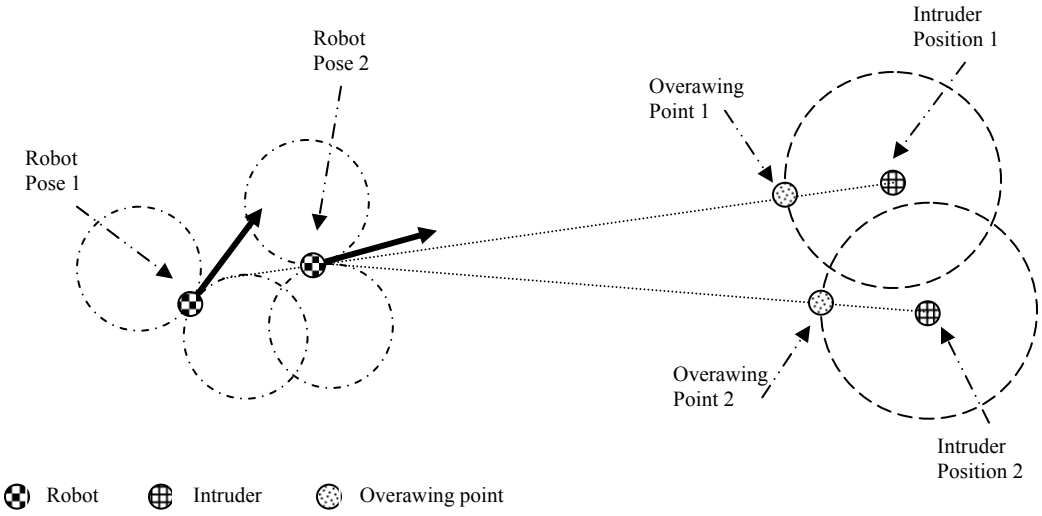


Figure 6.5 Demonstration of intruder tracking strategy

Once the RBP finds the possible intruder, the location of the intruder is sent to the OT. Then, the OT starts to search the similar objects around the provided location. If the possible intruder is found, the OT will calculate a proper position close to the possible intruder. This position point is called the “intruder overawing point”, which is close enough to the possible intruder, while still keeping a safe distance from the intruder. The coordinates of the intruder overawing point is sent to the path planner component along with all the non-map objects detected by the OT. The path planner component will create a proper path against the objects around it.

Figure 6.5 shows the strategy of the intruder overawing point calculation. When the possible intruder is found and the robot is trying to follow the intruder, two circles which have the radii equal to the minimum turning radius of the robot - are considered. A line is drawn, which passes the intruder and tangent to one of the two circles (the closer one). The intersection point of this line and the safety circle is the overawing point. The safety circle has a radius which is a property of the speed of the robot.

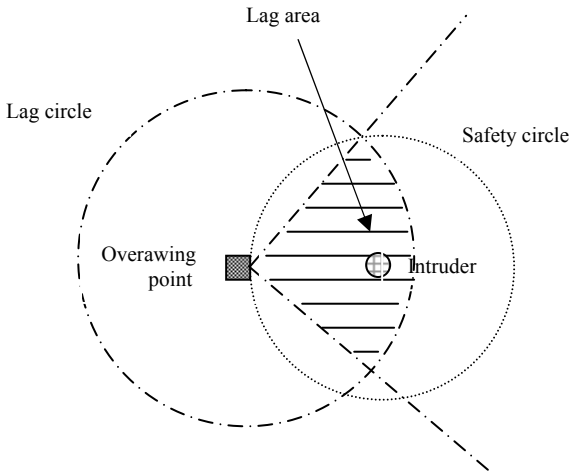


Figure 6.6 Definition of lag area for tracking

In the real applications, the intruder may keep moving, which will result in continuing calculations of the overawing point and tracking path from the OT and from the path planner component. To deal with this situation, a lag area is created to avoid the frequent calculation requirements. The lag area is a zone defined as such: if the intruder is inside of this zone, all motion from the intruder will be ignored and there are no new calculations on overawing points or necessary paths. Figure 6.6 demonstrates how the lag area is created. After the overawing point has been calculated, a circle around the point with a constant radius is drawn. This circle is called the lag circle. A sector is created from the center of the lag circle by two radii and a given angle (see Figure 6.6).

The bounded sector area is the lag area. If the intruder moves out of the lag area, OT will re-calculate the overawing point to track the intruder.

6.5 Intruder tracking component and related messages

6.5.1 Component overview

The Object Tracking (OT) transfers the range sensor data to segments using breaking points detection and regression line detection algorithms. Any object with a single point will be removed. A set of segments is created as the current objects.

The OT checks all the objects and finds those which are similar to human size. Then, according to the intruder location from Range Based Pose (reference intruder), the OT chooses an intruder-like objects that is closest to the reference intruder as the selected intruder.

If the suggested intruder is confirmed among these objects, an overawing point related to the intruder will be determined. A path query including the coordinate of the overawing point and the current objects is sending to the path planner component.

According to the location of the tracked object/intruder and the current objects, the path planner will create a set of segments (path) from the robot to the overawing point. To check the validity of the proposed path, the OT creates a rectangle around each path segment. The size of the rectangle is determined by the parameters defined in the database. The set of segments created from the sensor is checked against the created path rectangles to detect any objects that may be inside of or intersect with the path rectangle.

If there is object(s) inside of the path rectangle and the current objects set is different from the last step, the OT will set these segments as obstacles to the path planner component (with a defined frequency) for a new path.

To react on the motion of the tracked intruder properly, the overawing point will be updated if the intruder is out of the current lag zone and a query for the new tracking path will be sent to path planner component.

6.5.2 Description of OT component

The Object Tracking (OT) component works when the following/tracking mode is enabled. The OT includes three types of functions: data pre-processing, intruder finding, and obstacles detection.

Data pre-processing

1. All points which are out of the map perimeter are removed by the points clipping function.
2. Using a constant threshold, all breakpoints are marked. Then, all of the points are placed into different groups according to the marked breakpoints.
3. An improved line regression algorithm (described in chapter 5) is applied to extract line segments from the points group.
4. The OT checks the path segment list from the Report Element message. If any one has been changed, OT will create new path check rectangle.
5. All the segments (objects) from step 3 are checked against the path check rectangle. If there is any object inside of the path rectangle, OT will send the information about these objects and query for the new path to the planner component.

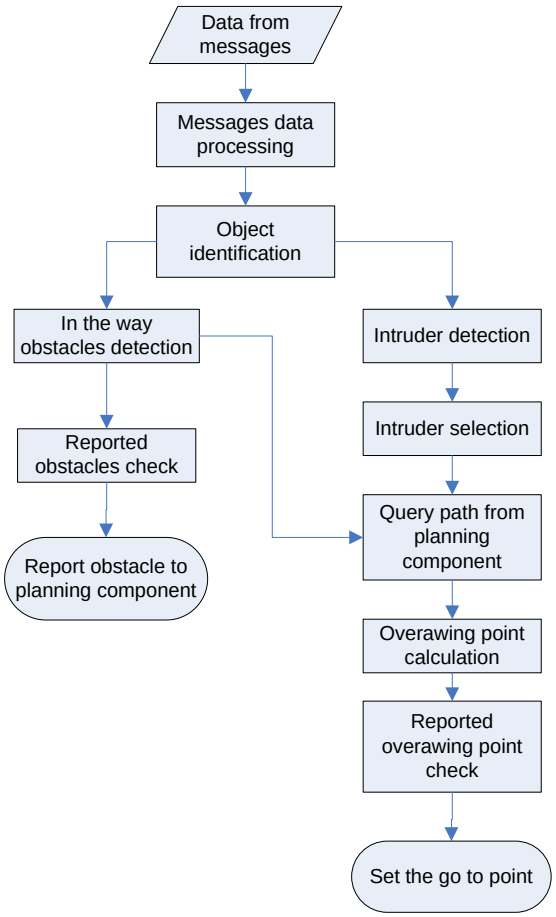


Figure 6.7 Flowchart of intruder tracking

Intruder tracking:

1. The OT detects all objects to find possible intruders by checking the size of each object. If there is more than one intruder, the one closest to the referred intruder will be chosen. The coordinate of the overawing point related to the selected intruder and the objects are sent to the path planner component.
2. After the path planner component returns a plan, the OT will monitor the validity of the path plan by check the path segment polygons against the object segments from the laser sensor data.

3. OT monitors the change on the overawing point. If the update on the overawing point is necessary, OT sends the new overawing point to the path planner component using Set Pose Message.

Obstacles detection:

The segments from step 5 in Data Pre-Processing are reported to the path planner component as the obstacles. If any of these obstacles are closer than a defined distance (called safety distance), the robot will be stopped immediately. Otherwise, a new plan to avoid these obstacles will be created.

Figure 6.7 shows the flowchart of object tracking component.

6.5.3 Messages descriptions

The Object Tracking (OT) component uses the standard JAUS message header structure. Except for the existing JAUS core input and output message sets (like shutdown, standby, reset, etc.), the following messages for the map data query and the corresponding responses are currently supported:

Incoming messages:

- Report Map Capability
- Report Map Data
- Report Global Pose and Velocity
- Report Range Sensor
- Set Tracking
- Report Element
- Report OpenSpacePathPlan

Sending messages:

- Query Map Capability
- Query Map Data
- Query Range Sensor
- Query Global Pose and Velocity
- Query Element
- Set Global Pose
- Query Open Space Path Planner
- Set Obstacle

The special messages of OT are listed in Appendix E.

6.6 Tests and Experiments

Range Based Pose (RBP) has been implemented into the whole system as a component with multi-functionalities (i.e. localization and obstacle detection). The tests and experiments of localization functionality have been discussed in the previous chapter. The results of an experiment for obstacle detection are shown below. The Object Tracking (OT) component uses the same algorithm as the one for obstacle detection and has more controlling issues involved that will not be presented here.

The experiments include two different path assignments. First, there is the regular patrolling routine (a regular rectangle). The second is an arbitrary running trajectory. The two original paths are show in Figure 6.8 and 6.9.

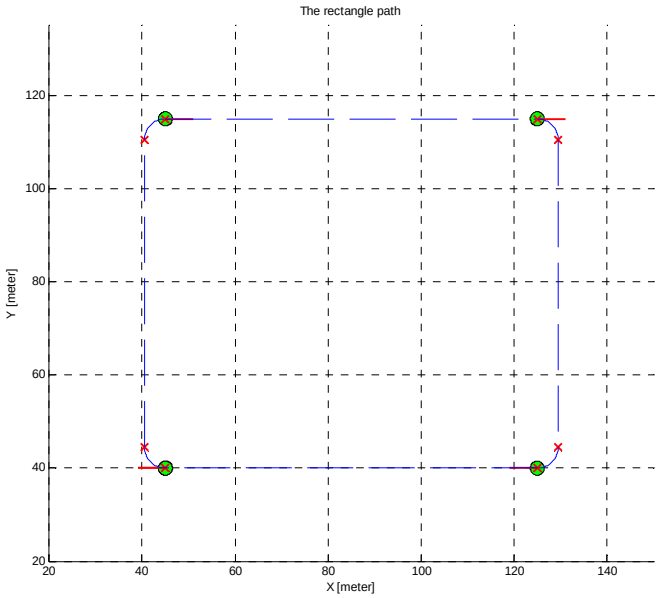


Figure 6.8 Rectangle path experiment

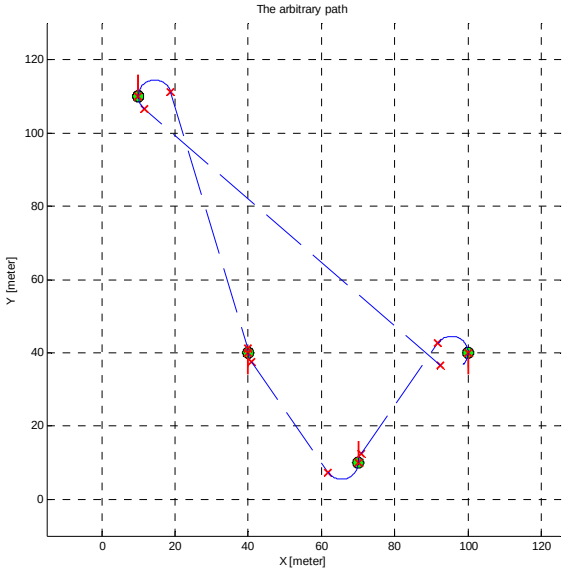


Figure 6.9 Arbitrary path plan experiment

In Figures 6.8 and 6.9, the small circles with dash direction represent the required way points which have to be reached and the small crosses show the terminal points from the

different path sections. The dashed line represents the original planed trajectory against the required way points. The tests are done without any non-map obstacles, at first. The robot ran along the planned path (see Figure 6.10 and 6.11).

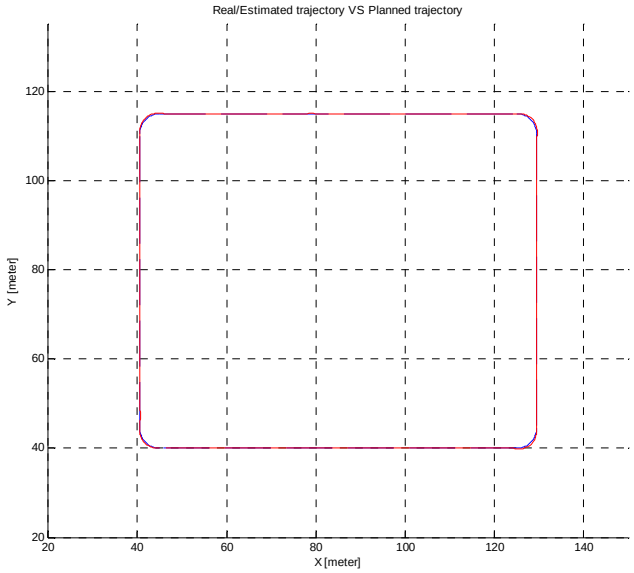


Figure 6.10 Estimated trajectory vs. planned trajectory from rectangle path test

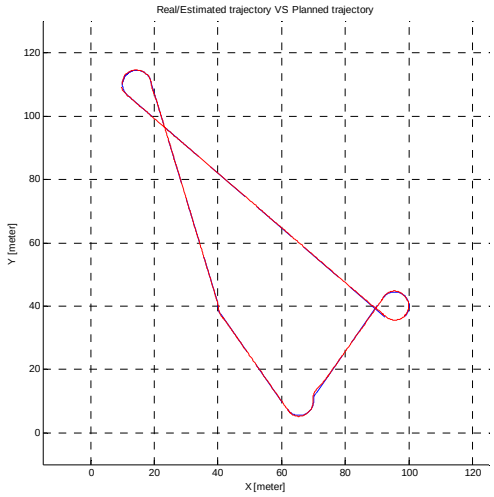


Figure 6.11 Estimated trajectory vs. planned trajectory from arbitrary path test

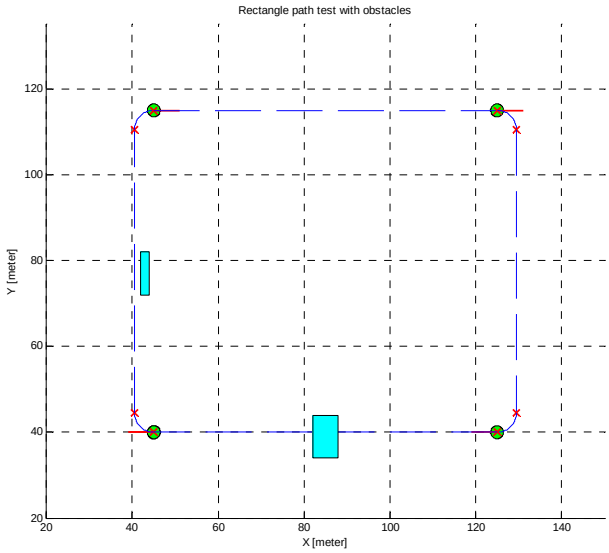


Figure 6.12 Rectangle path test with obstacles

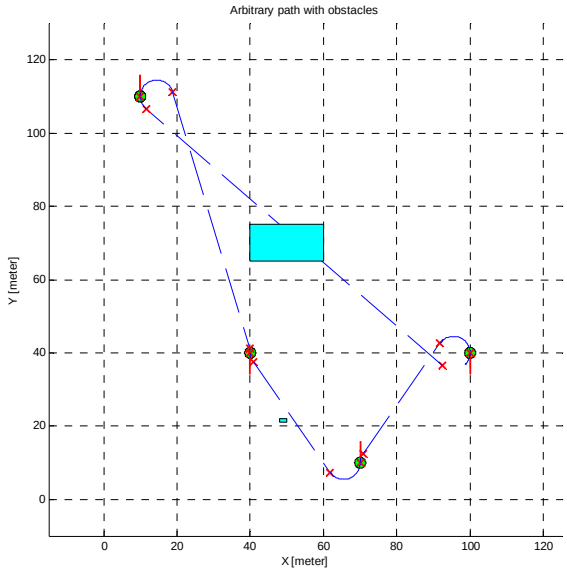


Figure 6.13 Arbitrary path test with obstacles

To prove the algorithm of objects detection, the obstacles with various sizes are placed in the robot's planned path (or very close to the path). These obstacles are not map objects which are known by the robots. Figures 6.12 and 6.13 show the locations of the obstacles related to the planned paths for both rectangle and arbitrary paths. The rectangles in

figure 6.12 and 6.13 are the obstacles which are define in the prior map and the experimental field..

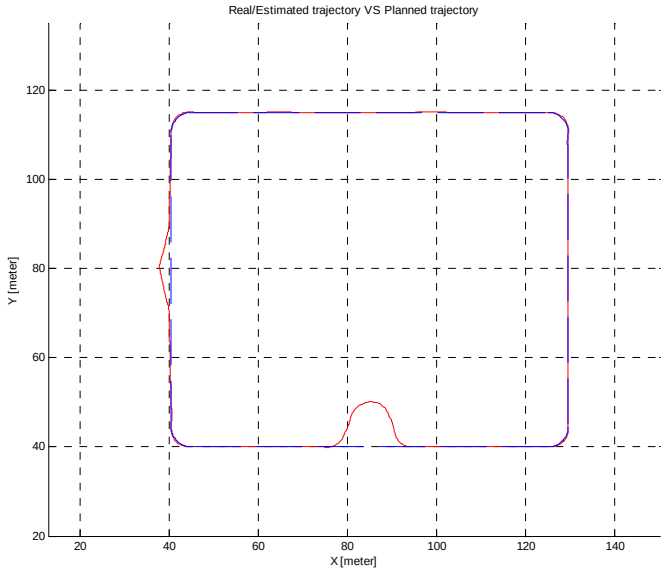


Figure 6.14 Estimated trajectory vs. planned trajectory from rectangle path test with obstacles

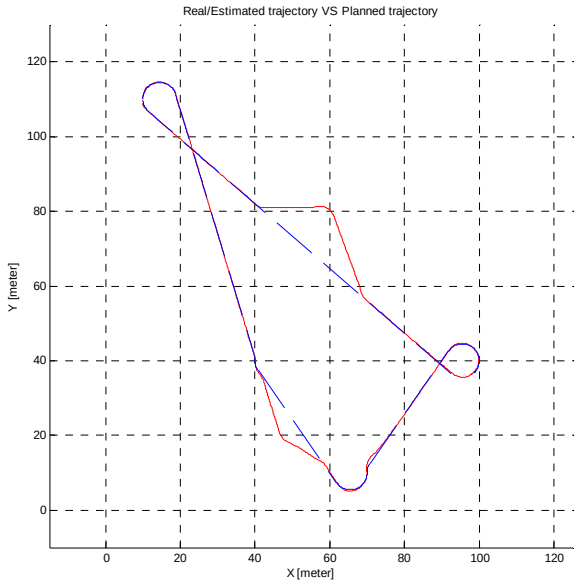


Figure 6.15 Estimated trajectory vs. planned trajectory from arbitrary path test with obstacles

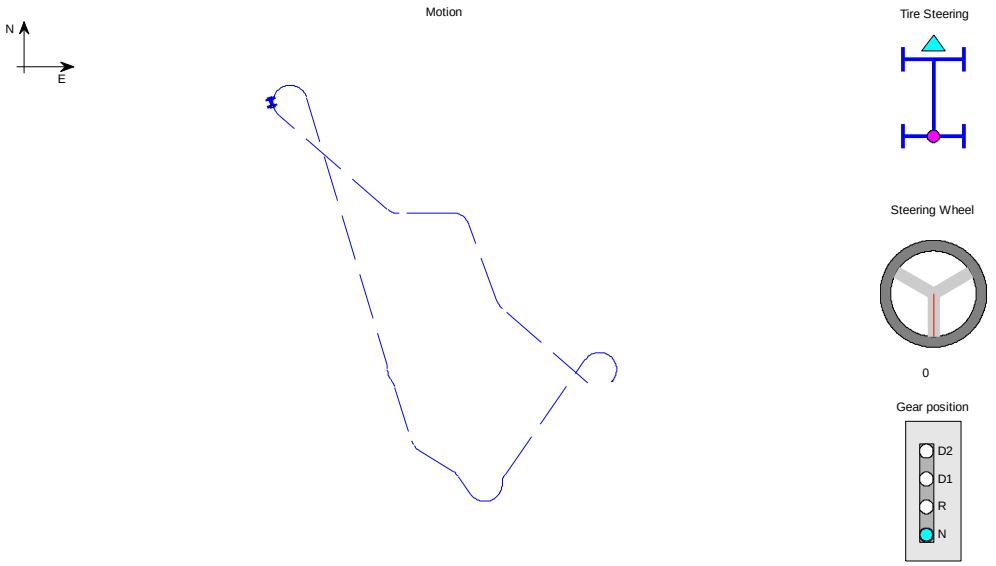


Figure 6.16 screen shot from the arbitrary path test with obstacles

At first, the robot will receive the required task as represented by a sequence of way points, and will plan its path without the consideration of non-map obstacles. When the robot accesses close to the obstacles (in the range of the sensor detection, which is $30\,m$ in this experiment), the Range Based Pose component (RBP) will find the obstacles blocking the way or being close to the path, which causes a potential collision risk for the robot. The RBP will report the locations of the obstacles to the Path Planner then the Path Planner will create a new local path around the obstacles. Figures 6.14 and 6.15 show the paths plotted from the data collection of the experiments for both rectangle and arbitrary paths.

Figure 6.16 is the screen shot from the system GUI during the arbitrary path test with obstacles, and it shows the motion trajectory of the robot according to the appearance of the non-map obstacles.

6.7 Conclusion

By applying the data association algorithm described in chapter 5, two JAUS compatible components, the Range Based Pose and the Object Tracking, were developed and implemented. Using the map information from another component, the Map Data Server (described in chapter 4), the Range Based Pose may improve the accuracy of the robot's estimated pose, recognize the map objects, find the non-map objects, and detect the objects that are on the robot's path. The Object Tracking may locate and track the object according to the commands from higher level control components or human operators.

The strategies described in the previous sections are integrated into the two components. These strategies may vary between different applications. The behavior of the robots in various environments is defined by these strategies.

The functionalities of the two components have been fully tested by simulations and real-time experiments.

Chapter 7

Localization with Particle Filtering

7.1 Introduction

The localization improvement by matching laser sensor data has been discussed in the previous chapters. In this chapter, an implementation of particle filtering is proposed.

For outdoor applications, the objects in the environment may have similar shape and characteristics. Figure 7.1 is a demonstration map of an outdoor warehouse.

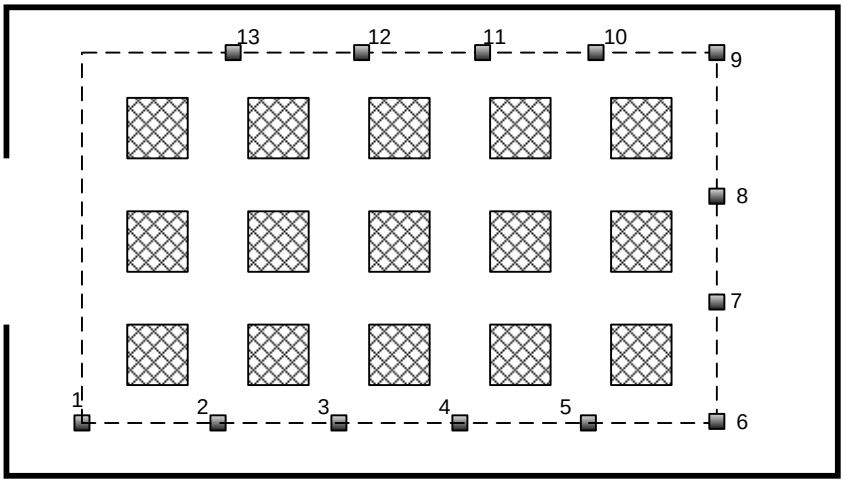


Figure 7.1 Layout of an outdoor warehouse

The cargo containers (the squares with pattern) of the same size are piled inside the fenced area. A robot is used to carry out the patrol along a given trajectory (the dot line). The robot is equipped with a laser range sensor, GPS, and Odometer sensor.

If there is no uncertainty in GPS signal and/or no noise in odometer measurements, the robot should have correct pose information and navigate along the given path.

If there is uncertainty in GPS and/or noise in odometer measurements, the laser range sensor is the only physical method to localize the robot's pose. In this case, the laser range sensor is utilized to supply the robot with more precise pose information. To achieve this, the measurements from the range sensor are processed and then are compared with the map scans using the matching algorithm described in chapter 5. Unfortunately, the matching process may not give the correct result due to the multiple pose possibilities from the repetitive environment features. For instance, in Figure 7.1, the laser sensor will get very similar scanning results when the robot is in points $\{1,2,3,4,5\}$ or $\{6,7,8\}$ or $\{9,10,11,12,13\}$. If the robot is actually at point 2 and the pose information from GPS/odometry notifies the system that the robot is at or close to point 3, the matching procedure will never identify the real pose. The scan from point 2 (real scanning) and the scan result from point 3 (map scanning) will be matched that indicates point 3 is the true pose.

In this case, if the maximum effective scan range is assumed to be equal to the interval between consecutive length of all the points and the maximum uncertainty from GPS and odometer are equal to or less than the interval between consecutive length of the points, there are at least two different possible poses every time.

For the outdoor applications, the uncertainty from GPS and odometer can not be eliminated. A proper method has to be addressed to solve this problem from the repetitive environment features, either from hardware or software. First, Differential GPS (DGPS) may be considered. DGPS enhances the normal GPS accuracy using differential corrections. A reference station is applied to test GPS signal and ignore the GPS signal from the 'unhealthy' satellites [84]. The disadvantage is the high cost of the DGPS service compared to normal GPS. The remaining problem is that the robot may still not work well if there is a blockage or lack of DGPS signal.

From the navigational point of view, technology is needed which may find multiple possibilities and keep these multiple possibilities as long as necessary.

In this chapter, an interesting method called a particle filtering technique is introduced and an implementation of a proposed algorithm for the particle filtering method is discussed. The developed algorithm may be used for localization of outdoor mobile robots, especially in those environments in which objects have similar geometry characters.

Section 7.2 discusses the application of conventional particle filter for mobile robot localization using an example adapted from Rekleitis' work [85, 86]. Comparing to the traditional method, section 7.3 shows a developed proposed algorithm of the particle filter techniques that applies the developed scan data association algorithm to solve the problem discussed in section 7.1. The simulation results and related discussion are in section 7.4.

7.2 Particle Filtering for Mobile Robot Localization: An Example

The concept of particle filter has been introduced in chapter 2. When the robot is running in its working environment, a set of hypotheses about the robot's pose or about the locations of the objects around the robot may be maintained. The input for updating these hypotheses comes from the various sensors' measurements. The different estimation algorithms may be employed to update these beliefs accurately [85, 87]. The location of the obstacles observed in the past can be updated if more data is coming from the sensors. Also, the estimation of the pose can be updated based on these data.

The Kalman Filter is a standard approach for reducing the error in measurements from different sources [13, 87, 88, 89, 90, 91]. A variation is based on Extended Kalman Filtering (EKF) in which a nonlinear model of the motion and measurement equations is used [92, 93, 94, 95].

Particle filters, based on Sequential Monte Carlo methods, are complicated estimation techniques. In mobile robotics, particle filtering has been applied successfully for a single robot and multiple robots [34, 37, 38, 105, 96, 97, 86, 98]. Among these existing works, [98] uses the laser scan matching to update the measurements of particle filter in indoor mobile robots application. The differences between [98] and the proposed algorithm in this thesis are located in: different laser scan data matching methods, different particle distribution methods, different weight calculation methods and the different purposes of using particle filter.

Both the Kalman Filter and the particle filter are kinds of derivatives of the well known Bayes filter. Bayes filtering uses a predict/update cycle to estimate the state of a dynamic system from sensor measurements [99].

The intention of this section is to show the traditional particle filtering procedure in mobile robot localization. This following example in this section is adapted from Rekleitis' work [85, 86] in which the author applies the particle filtering technique to multiple mobile robots localization. To make the procedure more clear, just the single robot localization is considered instead of the multiple robots localization in the original work. The procedure is demonstrated in three phases: prediction, update, and re-sampling.

The pose of the robot at time k is represented by $[x(k), y(k), \theta(k)]$ for 2-dimension case or $[x(k), y(k), z(k), \theta(k), \psi(k), \phi(k)]$ for 3-dimension case. $[x, y, z]$ are the coordinates and $[\theta, \psi, \phi]$ are yaw, roll and pitch angles. The 2-dimension state is used here. The system state at time k is set as $X(k) = [x(k), y(k), \theta(k)]$. Given a set of N particles $S(k) = [X^j(k), w^j(k)], j = 1...N$, each particle has a copy of the state $X^j(k)$ and a weight $w^j(k)$ that defines the contribution of this particle to the whole estimation of the state.

At time $t = k$, the effect of the operation may be modeled to obtain a prior of the PDF at time k using the previous PDF (at time $t = k - 1$). In other words, a model is adopted in order to simulate the effect of the operation. This is the prediction phase. After the new sensor data is available, the weights of all the particles are calculated and compared to the sensor data. Thus, the new weights may describe the robot's pose PDF. This is the update phase. A formal description of the particle filtering algorithm from Rekleitis is listed in Figure 7.2 [85, 86].

Given : *A set of particles for robot's pose at time* 0 :

$S(0) = [X^j(0), w^j(0) : j = 1...N]$

w is the weight and N is the number of particles.

While (*robot running*) *do*

$k = k + 1;$

if $(ESS(W) < \beta * N)$ % *particle population depleted*

$Index = \mathrm{Re}\,sample(W);$

$S(k) = S(k)(Index);$

endif

for $(j = 1\ to\ N)$ %*prediction after action m*

$X^j(k+1) = \hat{f}(X^j(k), m(k));$

end

$s = Sense()$ %*observation process*

for $(j = 1\ to\ N)$ %*update the weights*

$w^j(k+1) = w^j * W(s, X^j(k+1))$

end

for $(j = 1\ to\ N)$ %*Nomalize the weights*

$$w^j(k+1) = \frac{w^j(k+1)}{\sum_{n=1}^{N} w^n(k+1)}$$

end

end while

Figure 7.2 Particle filter algorithm[85, 86]

After the PDF of the robot's pose is known from the particles' weights, three different methods of evaluation could be used to obtain an estimation of the pose in [85, 86].

1. The weighted mean : $Pose_{estimated} = \sum_{n=1}^{N} w^n X^n$;
2. The best particle : $Pose_{estimated} = Pose_j, here, Pose_j = \max(w^k, k = 1..N)$;
3. The robust mean : the weighted mean in a small window around the best particle;

The weighted mean will fail faced with multi-modal distribution. The robust mean has an expensive computation cost.

The following paragraphs present the details of prediction, updating, and re-sampling.

Prediction

To predict the probability distribution of the robot's pose after a motion, the model of the robot plus the effect of the noise is needed. There are many different approaches to simulate the effect of the noise [86, 100, 101]. Most of them use an additive Gaussian noise. For mobile robots, any arbitrary motion $[\triangle x, \triangle y]^T$ can be divided into two steps, a rotation followed by a translation (See Figure 7.3). Assuming that the robot's initial pose is $[x, y, \theta]^T$, the robot rotates by $\delta\theta = \theta(k) - \theta, where\ \theta(k) = \arctan(\frac{\triangle y}{\triangle x})$ at first, then translates forward by distance $\rho = \sqrt{\triangle x^2 + \triangle y^2}$.

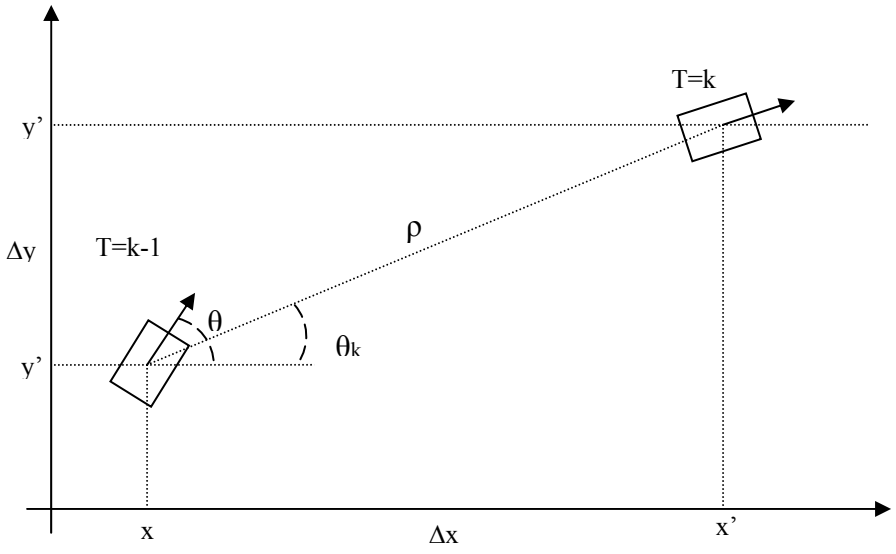


Figure 7.3 Arbitrary motion of robot [86]

The resulting pose $[x', y', \theta(k)]$ is given by

$$\begin{bmatrix} x' \\ y' \\ \theta(k) \end{bmatrix} = \begin{bmatrix} x + \rho\cos(\theta(k)) \\ y + \rho\sin(\theta(k)) \\ \theta(k) \end{bmatrix} \tag{7-1}$$

The noise models are applied separately to the rotation and the translation.

Update

After an operation (motion), new observation data is received from the sensors or other additional sources.

In Figure 7.4, there are two mobile robots, one is stationary and acts as the observation target. Another is moving and observing. The assumption is that the target robot is inside of the range of the moving robot's sensing. The moving robot's pose is $X_m = [x_m, y_m, \theta_m]^T$ and the stationary robot's pose is $X_s = [x_s, y_s, \theta_s]^T$. So, each particle has the pose vector as $X^i{}_m = [x^i{}_m, y^i{}_m, \theta^i{}_m]^T$. The sensor measurement presented by $z^i = [\rho^i, \theta^i, \phi^i]^T$ can be calculated by

$$z^i = \begin{bmatrix} \rho^i \\ \theta^i \\ \phi^i \end{bmatrix} = \begin{bmatrix} \sqrt{dx_i^{\,2} + dy_i^{\,2}} \\ \arctan 2(dy_i, dx_i) - \theta^i{}_m \\ \arctan 2(-dy_i, -dx_i) - \theta_s \end{bmatrix} \tag{7-2}$$

Where $dx^i = x_s - x^i{}_m$ and $dy^i = y_s - y^i{}_m$.

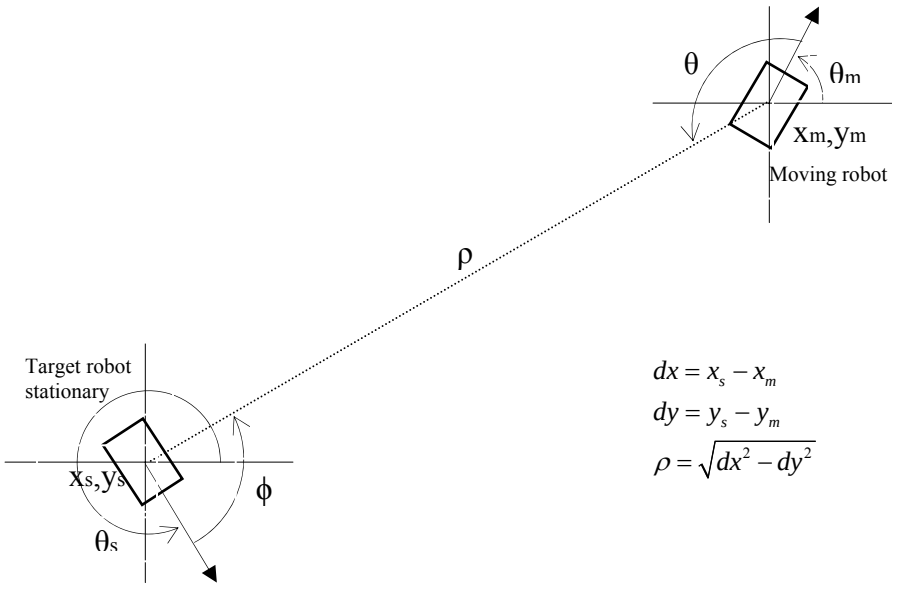


Figure 7. 4 Demonstration of sensing [86]

If the pose of the stationary robot X_s and the sensor measurement $z^i = [\rho^i, \theta^i, \phi^i]^T$ are known, the estimated pose of the moving robot may calculated with

$$X_{m-est} = \begin{bmatrix} x_{m-est} \\ y_{m-est} \\ \theta_{m-est} \end{bmatrix} = \begin{bmatrix} x_s + \rho * \cos(\phi + \theta_s) \\ y_s + \rho * \sin(\phi + \theta_s) \\ \pi + \phi + (\theta_s - \theta) \end{bmatrix} \tag{7-3}$$

According to equation 7-2, the value of ϕ^i is affected by the pose of the particle. So, given X_s and z^i, the weight for particle i is proportional to the probability of $X_m^i(k+1)$. The probability of $X_m^i(k+1)$ may be calculated as

$$P(X_m^i(k+1) \mid X_s, z^i) = \frac{1}{\sqrt{2\pi}\sigma_\rho} e^{\frac{-(\rho-\rho^i)^2}{2\sigma_\rho^2}} \frac{1}{\sqrt{2\pi}\sigma_\theta} e^{\frac{-(\theta-\theta^i)^2}{2\sigma_\theta^2}} \frac{1}{\sqrt{2\pi}\sigma_\phi} e^{\frac{-(\phi-\phi^i)^2}{2\sigma_\phi^2}} \tag{7-4}$$

Where $\sigma_\rho, \sigma_\theta, \sigma_\phi$ are the presumed standard deviations of the measurement noises. Equation 7-4 uses the polar coordinate centered at the sensor location. With Cartesian coordinate, the probability of $X_m^i(k+1)$ may be calculated by

$$P(X_m^i(k+1) \mid X_s, z^i) = \frac{1}{\sqrt{2\pi}\sigma_\rho} e^{\frac{-(dx-dx^i)^2}{2\sigma_\rho^2}} \frac{1}{\sqrt{2\pi}\sigma_\rho} e^{\frac{-(dy-dy^i)^2}{2\sigma_\rho^2}} \frac{1}{\sqrt{2\pi}\sigma_\theta} e^{\frac{-(\theta_m-\theta_m^i)^2}{2\sigma_\theta^2}} \tag{7-5}$$

As discussed in the previous paragraph, the estimated pose may be obtained using different methods [85, 86].

1. Weighted mean:

$$X_m = \sum_{i=1}^{N} X^i w^i$$

2. The best particle:

$$X_m = \max(X_m^i)$$

3. Robust mean:

$X_m = \sum X_m^i w^i : \left| X_m^i - \max(X_m^i) \right| \le \varepsilon$, ε is an infinite small value.

Given : *double array* $W[N], \sum_{i=1}^{N} W_i = 1$

$Q = cumsum(W)$; *%calculate the running totals* $Q_j = \sum_{l=0}^{j} W_l$

$t = rand(N+1)$;
$T = sort(t)$;
$T(N+1) = 1$;
$i = 1$;
$j = 1$;
while$(i \le N)$ *do*
  if $(T[i] < Q[j])$
    index$[i] = j$;
    $i = i + 1$;
  else
    $j = j + 1$;
  end if
end while
return (*index*)

Figure 7.5 Select with replacement algorithm [85]

Re-sampling

After a few iterations, most of the particles have very small weights (or may be zero after rounding) due to the drifting of motion. The contribution to the pose's PDF from these particles can be ignored. If the current set of particles is $S = \{X^i, w^i\}, i = 1...N$, a new set of particles $S' = \{X'^i, w'^i\}, i = 1...N$ is needed such that $X^i = X'^l, 1 \le i, l \le N$ and weights $w'^k = 1/N$ that represent the same PDF of the robot's pose.

Two different measures, the coefficient of variation cv_t^2 (Equation 7-6) and the effective sample size ESS_t (Equation 7-7), are presented in [86, 102]. They are used to estimate the number of weights which are near to zero.

$$cv_t^2 = \frac{\mathrm{var}(w_t^i)}{E^2(w_t^i)} = \frac{1}{N}\sum_{i=1}^{N}(Nw^i - 1)^2 \tag{7-6}$$

$$ESS_t = \frac{N}{1 + cv_t^2} \tag{7-7}$$

When the effective sample size drops below a percentage of the number of particles N, re-sampling is needed. During the re-sampling, the particles with small weights are eliminated and the particles with big weights are duplicated.

$Given: double\ array\ W[N], \sum_{i=1}^{N} W^i = 1$

$Q = cumsum(W);\ \ \%calculate\ the\ running\ totals\ Q_j = \sum_{l=0}^{j} W^l$

$t = -\log(rand(N+1));$

$T = cumsum(t);$

$TN = T / T(N+1);\quad \%normalize\ T\ to\ TN$

$i = 1;$

$j = 1;$

$while(i \le N)\ do$

  $if\ (T[i] < Q[j])$

    $index[i] = j;$

    $i = i + 1;$

  $else$

    $j = j + 1;$

  $end\ if$

$end\ while$

$return\ (index)$

Figure 7. 6 Linear time re-sampling algorithm [85]

```
Given : double array W[N], sum_{i=1}^{N} W^i = 1
for( j = 1 to N )   %update the weights
    a^j = sqrt(W^j);
end
sum = 0;
for( j = 1 to N )   %calculate sum_{i=1}^{N} a^j
    sum = sum + a^j;
end
for( j = 1 to N )  % normalize the weights a to sum up to N
    a^j = N * a^j / sum;
end
i = 1;
for( j = 1 to N )
    if (a^j ≥ 1)  %big weight
        for(l = 1 to a^j)
            index[i] = j;
            i = i + 1;
        end
    else
        R = rand(1);
        if (a[j] ≥ R)   %high probability
            index[i] = j;
            i = i + 1;
        end if
    end if
end
return (index)
```

Figure 7. 7 Weights function re-sampling algorithm [85]

Three of the most common methods of re-sampling are “select with replacement”, “linear time re-sampling”, and “weights function re-sampling”. “Select with replacement” [86] is the simplest method. The probability of each particle is set to be equal to its weight in this method. “Linear time re-sampling” [103] is based on a manipulation of the random number sequence in order to get a new sorted number sequence in linear time. In “weights function re-sampling” [102], a set of numbers, which is from a function of particle weights (like the square root of weights), is used to decide which particles are

going to be propagated forward. For this method, the number of particles after re-sampling is not equal to the previous N as the choice of how many particles survive is stochastic.

The procedures of the three algorithms are presented in Figure 7.5 to 7.7. In every algorithm, the input is an array of the weights of the particles and the output is an array of indices of which particles are going to propagate forward. It has to be noted that the PDF re-built by the re-sampled population must be very close to the one before the re-sampling. Experiments show that there are no noticeable differences among the three re-sampling algorithms [85, 86].

7.3 Proposed Particle Filter Algorithm Based on Laser Data Association

From the example in section 7.2, the characteristics of particle filtering may be described as:

- The PDF is represented by a set of samples (particles)
- Each particle contains one set of values for the state variables
- The PDF may be arbitrary that enable both simple model (like Gaussian) and a complex model (multi-model).

In this section, to handle the multiple mode situation described in section 7.1 (see Figure 7.1), a particle filtering which apply the matching algorithm presented in this thesis previously is proposed. Considering the odometer error and GPS uncertainty in an outdoor application, the filtering technique has to be held for all the time when the robot is running.

This kind of particle filtering is discussed in four steps: initial sampling, prediction, update, and re-sampling.

In order to illustrate the algorithm logic, we will be using an example of an outdoor warehouse shown in Figure 7.8. The big rectangle with solid line is the planned trajectory of the robot patrol and the small rectangles enclosed by the trajectory are the cargo containers.

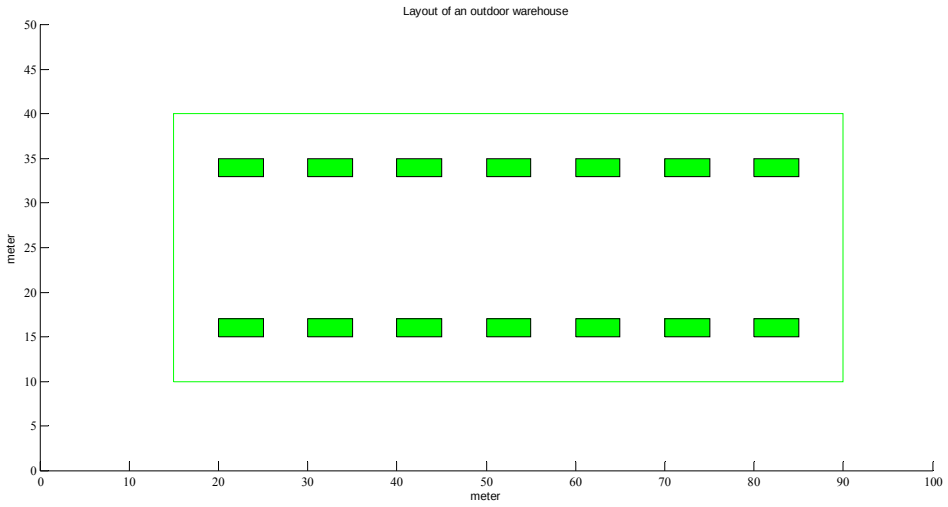


Figure 7. 8 Simulation of outdoor warehouse

Initial sampling

Assuming the robot is running in an outdoor warehouse simulated in Figure 7.8. As mentioned before, there are many locations in which the laser scanner equipped on the robot would get very similar scanning data due to the similarity of these cargo containers. If the range of the sensing from the equipped sensors is big enough, it is always hoped that the distribution of the particles could cover all the possible zones to catch all the possible models. In an outdoor application, the full covering of work space is either impossible or unnecessary. The reason to use particle filtering is to retrieve the correct

pose in spite of the error accumulation of odometry sensor and the uncertainty of GPS signals. If the range of the particle distribution is bigger than the numeric value of the sum of the errors, the particle distribution could be regarded as good enough. Another criterion for the choice of the particle distribution depends on the environmental characters. The range of the distribution may be defined to cover the space that includes similar objects. This method just works for special cases and will not be considered in this research.

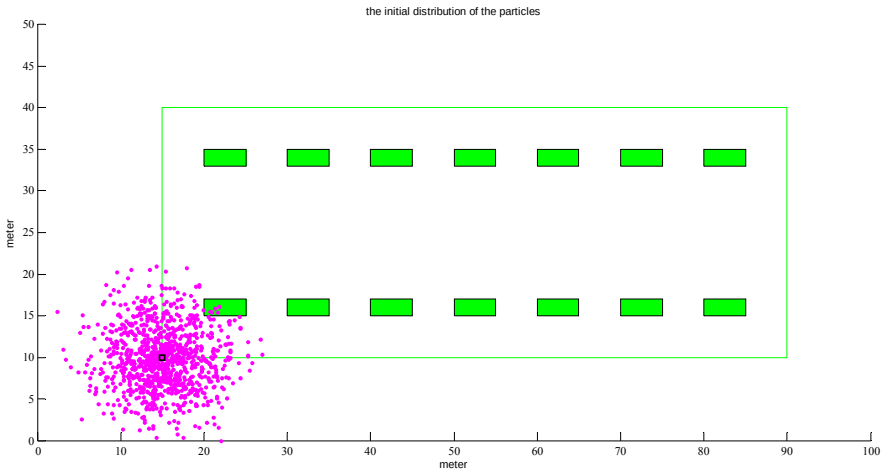


Figure 7. 9 The initial distribution of the particles

In this work, the maximum of the GPS error is defined as ± 5 *meters*. Considering other noises, the radius of the initial range of the particle distribution is set to 10 *meters*. The initial particle is created by a multivariate Gaussian/normal distribution method [104]. The mean vector is defined as $\begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \end{bmatrix}$ which is the reported robot pose. The covariance matrix is defined as:

$$P_{\text{cov}-init} = \begin{bmatrix} x_{offset}^2 & 0 & 0 \\ 0 & y_{offset}^2 & 0 \\ 0 & 0 & \theta_{offset}^2 \end{bmatrix}$$

where $x_{offset}, y_{offset}, \theta_{offset}$ is the maximum offset in the x, y axis and heading direction.

As shown in Figure 7.9, the robot is at a location close to the left bottom corner of the trajectory. The number of particles is assigned to be N. Obviously, some of these particles are not valid because they are located inside of occupied places or they are out of the boundary of the working space. A procedure is applied to check the validity of all the particles. If the particle is not valid, its weight is set to 0. All the particles which have passed the validity check will be set an average equal weight value as:

$$w^i = \frac{1}{N_{effective}}; i = 1..N_{effective}$$

Figure 7.10 shows the procedure of the initial sampling algorithm.

$Given: A\ reported\ pose \begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \end{bmatrix}$

$The\ required\ particle\ number\ N$

$The\ \max imum\ offsets\ in\ x\ direction, y\ direction, and\ heading \begin{bmatrix} x_{offset} \\ y_{offset} \\ \theta_{offset} \end{bmatrix}.$

$Initial\ sampling:$

$X_{mean} = \begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \end{bmatrix}$ $\%\ the\ reported\ pose\ is\ regarded\ as\ the\ mean\ vector$

$P_{\text{cov}} = \begin{bmatrix} x_{offset}^2 & 0 & 0 \\ 0 & y_{offset}^2 & 0 \\ 0 & 0 & \theta_{offset}^2 \end{bmatrix}$ $\%\ \text{cov}\ ariance\ matrix\ for\ particle\ distribution$

$L = length(X);\ \%\ length\ of\ the\ state\ vector$

$S = chol(P);\ \%\ Cholesky\ factorization$

$XX = randn(L, N)\ \%\ create\ random\ matrix$

$SS = S * XX + X * ones(1, N);\ \%create\ the\ particles$

$return\ SS\ \ \%N\ particles$

Figure 7. 10 Initial sampling algorithm

Prediction

Given : *Set of N particles* – *S*
The rotation aingle $\Delta\theta$
The translation dis tan *ce* ρ

for $i = 1 : N$
$x^i = x^i + \rho * \cos(\theta^i);$
$y^i = y^i + \rho * \sin(\theta^i);$
$\theta^i = \theta^i + \Delta\theta;$
$S'[i] = \begin{bmatrix} x^i \\ y^i \\ \theta^i \end{bmatrix};$
end for
return S'

Figure 7. 11 Algorithm for prediction step

The prediction step used in this research is similar to the description in section 7.2. Instead of the noise model, a simple motion mode is used. Every time-step, the error found by the matching results will be applied to all the particles. The error found from the matching algorithm is more trustworthy than any estimated noise models. Equation 7-6 is applied to each particle to predict the pose at the next time-step. The prediction procedure is shown in Figure 7.11.

Update

The weight of each particle from the initial step has been set to average $w^i = \frac{1}{N_{effective}}; i = 1..N_{effective}$ (for all the valid particles) or zero (for the invalid particles). In conventional algorithms for particle filtering, the weight of each particle has to be updated according to the measurements from the sensors. Many different estimation

techniques are applied [31, 32, 101, 102, 103]. For this proposed algorithm of particle filtering, instead of using the probability estimation model, the data from the laser range sensor and the data from the sensor simulator are matched to find the error on pose. The details of the matching process have been described in a previous chapter. For each particle, the offset between the particle state and the possible pose state is presented by the vector $\begin{bmatrix} \triangle x^i & \triangle y^i & \triangle\theta^i \end{bmatrix}$. The components are corresponding to the offsets on x axis, y axis, and the heading direction.

The new weights of each particle depend on its offsets from the possible poses. The weights are renewed by:

$$w^i = \frac{1}{\sqrt{(\triangle x^i)^2 + (\triangle y^i)^2 + (\triangle\theta^i)^2}} \tag{7-8}$$

For the cases in which the matching procedure failed, the weights for these particles will be set to zero.

The probability of each particle at this time is proportional to the weights. The PDF of particles could be represented by the normalized particle weights. The weight of each particle is normalized by the sum of the weights:

$$w^i = \frac{w^i}{\sum_{n=1}^{N_1} w^n} \tag{7-9}$$

where, N_1 is the number of the valid particles.

The upper portion of Figure 7.12 shows the PDF represented by the normalized weights of particles. The lower portion shows the corresponding particle distribution. Obviously, three possible poses have been caught by the particle filter. Note that there are three

components in the pose state vector and just x and y are presented in Figure 7.12. The updated algorithm is shown in Figure 7.13.

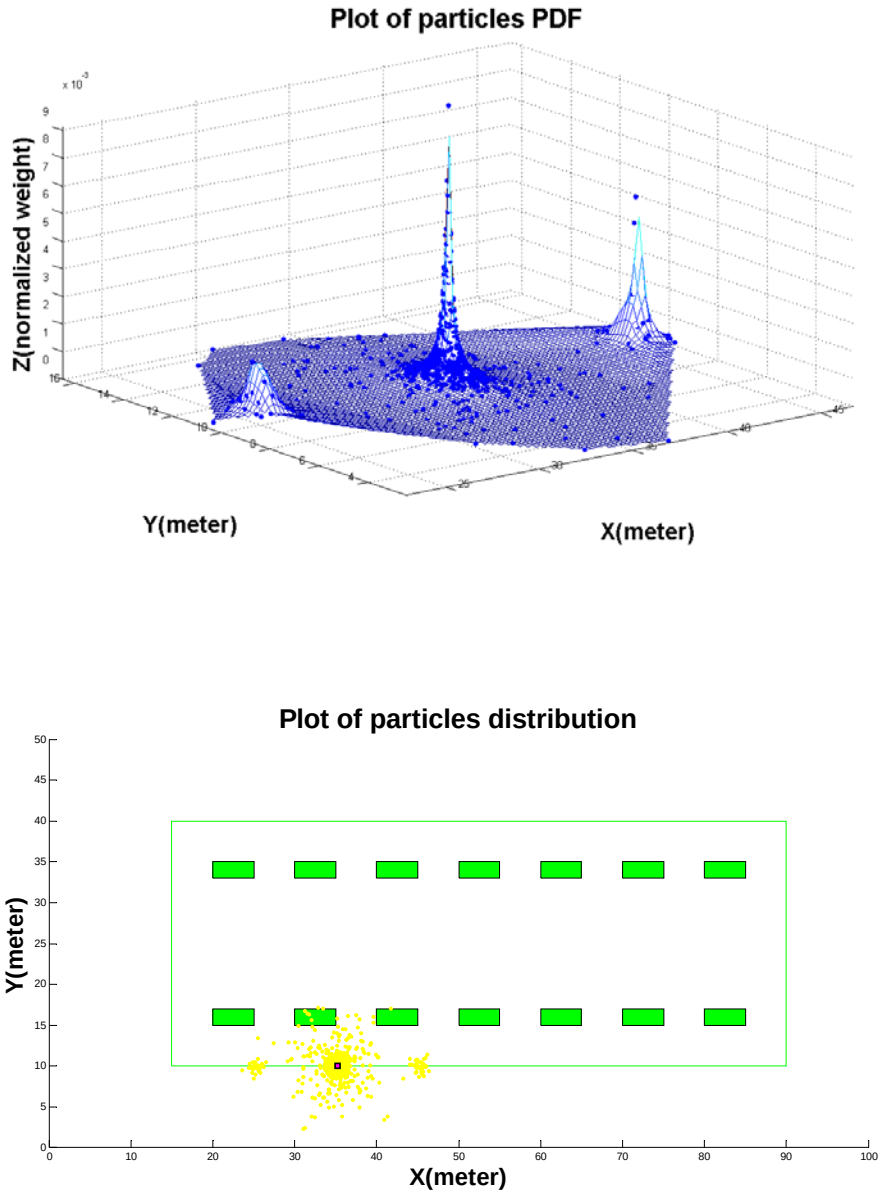


Figure 7. 12 Example of particle distribution and PDF

$$
\begin{aligned}
&Given: Set\ of\ \ N\ \ particles - S \\
&\qquad Scanning\ \ data\ \ from\ laser\ sensor - L(0) \\
&\\
&for\ i = 1:N \\
&\quad if\ \ (particle\ i\ is\ valid) \\
&\qquad L(i) = Scanning\ \ data\ \ from\ map\ scan\ at\ \ particle\ i; \\
&\qquad \begin{bmatrix} \triangle x^i \\ \triangle y^i \\ \triangle \theta^i \end{bmatrix} = match\ between\ L0\ and\ \ L(i); \\
&\qquad if\ \ (matching\ successful) \\
&\qquad\quad w^i = \frac{1}{\sqrt{(\triangle x^i)^2 + (\triangle y^i)^2 + (\triangle \theta^i)^2}}; \\
&\qquad else \\
&\qquad\quad w^i = 0; \\
&\qquad end\ if \\
&\quad end\ if \\
&end\ \ for \\
&W' = \frac{W}{SUM(W)}; \\
&return\ W'
\end{aligned}
$$

Figure 7. 13 Algorithm for update step

Re-sampling

As described before, the particles with very low weights will be eliminated and the ones with high weights will be duplicated during the re-sampling step. The criterion of when to and how to carry out re-sampling was also discussed before. Normally, the operation of re-sampling includes all the particles. As a result, the distribution of the particles after the re-sampling will be concentrated to the several locations represented by the particles with the highest weights.

As a dedicated feature in this proposed algorithm, a portion of the particles is always kept to distribute in a given range which is defined in the initial sampling step. This helps the distribution of particles can catch all the probabilities through all the running time.

The Equation 7-6 and 7-7 (see section 7.2) are used to judge the necessity of re-sampling. The Select with Replacement method (see Figure 7.4) is adopted to resample against the particles additional to the particles which have been chosen to distribute around the initial range $P_{\mathrm{cov}-init} = \begin{bmatrix} x_{offset} & 0 & 0 \\ 0 & y_{offset} & 0 \\ 0 & 0 & \theta_{offset} \end{bmatrix}$. Two important parameters are defined additionally: a) the threshold of the number of near zero points (a percentage of the number of particles) R and b) the number of particles N_{init} which will be distributed in the defined range ignoring the weights.

Figures 7.14 to 7.16 demonstrate the re-sampling step. The initial distribution of particles is shown in Figure 7.14. Figure 7.15 shows the PDF of the particles; the upper one is the view from isometric axis and the bottom one is the view from the y axis. In the PDF (normalized weights) from Figure 7.15, there are three peaks in the distribution which present the three possible poses caught by the particles. During the re-sampling step, most of the particles which have weights that have been set very low according to the matching procedure are eliminated and re-created at the three peak locations by the selection step with the Replacement re-sampling method. The distribution of particles after the re-sampling step is shown in Figure 7.16. It can be seen that the densities of particles around the possible locations is much higher than in other places.

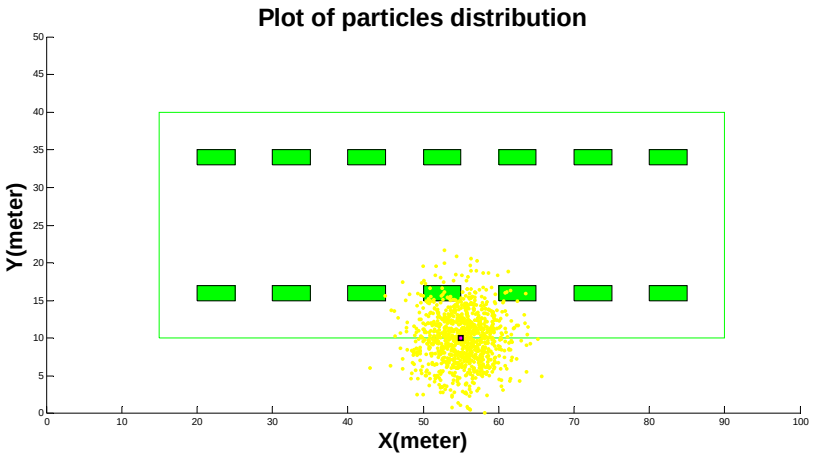


Figure 7. 14 The distribution of particles before re-sampling

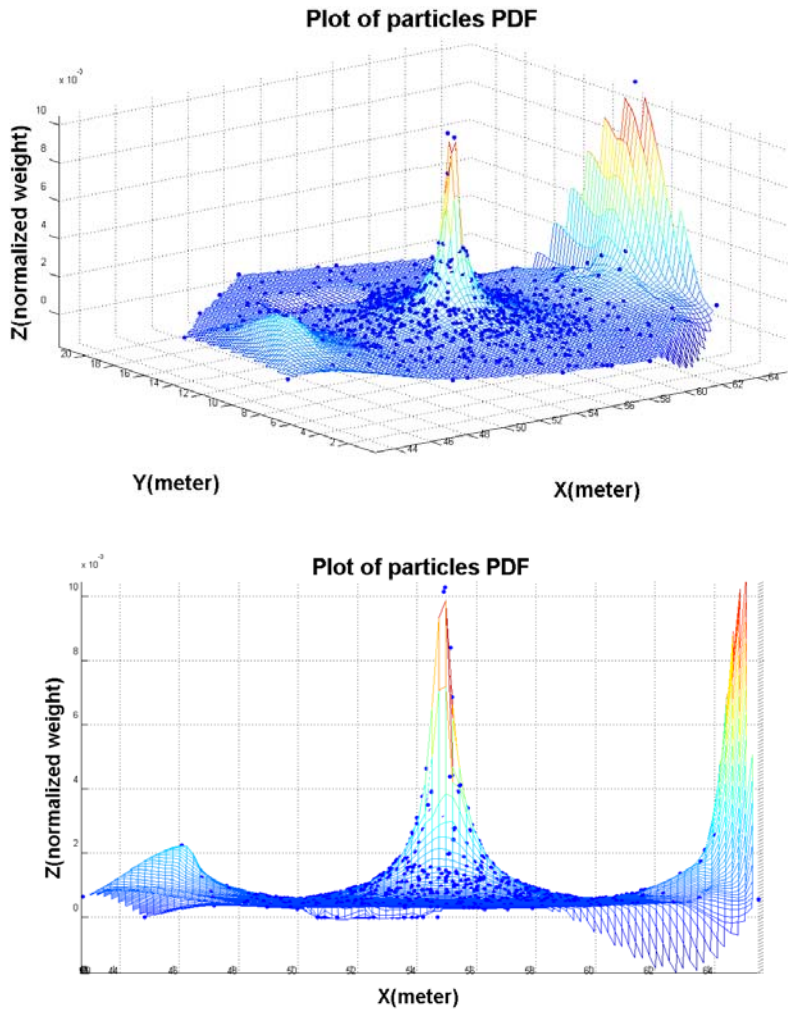


Figure 7.15 The plot of PDF before re-sampling

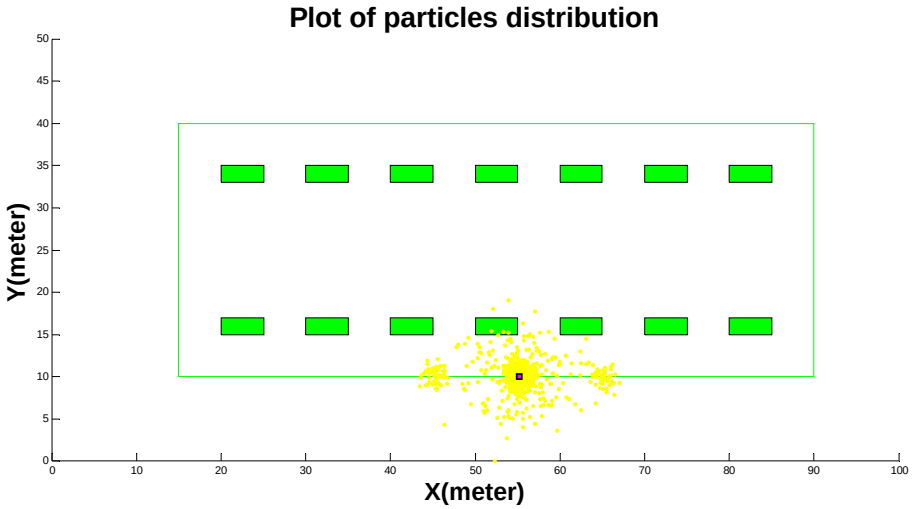


Figure 7.16 The distribution of particles after re-sampling

The algorithm of the re-sampling step is described in Figure 7.17.

Given : *Set of* N *particles* $- S\{x^i, w^i, i = 1...N\}$;

The threshold of the number near to zeros, R;

The percentage of particles in initial range, N_{init};

$$cv_t^2 = \frac{1}{N}\sum_{i=1}^{N}(N * w^i - 1)^2;$$

$$ESS_t = \frac{N}{1 + cv_t^2};$$

if $(ESS_t \le R)$

$S_{high} = \{x^i, i = 1...(N - N_{init})\}$;

$S_{low} = \{x^j, x^j < S_{high}(i), i = 1...(N - N_{init}), j = 1...N_{init}, \}$;

$S1 =$ *initial sampling with number* N_{init};

$S2 =$ *The selected replacement re* $-$ *sampling against to* S_{high};

$S = S1 + S2$;

end if

return S;

Figure 7. 17 Algorithm for re-sampling step

7.4 Simulation, Discussion, and Application to Localization Improvement

The development of the proposed algorithm particle filtering based on laser data matching has been discussed in section 7.3. This proposed algorithm is used to be an assistant to the existing localization functionalities. Specifically, the particle filter will help to catch the multiple possibilities in some environments. These environments include warehouses, airports, fenced boundaries and military bases and so on. For these environments, the whole place or portions of the place have many physical objects which have similar geometry sizes and have a similar distance between each other. This means that the laser range scanner will output similar scanning images at many locations. This problem is very difficult to be detected and be recovered by other technology.

The simulation is designed to test the ability of this proposed algorithm to find the multiple pose possibilities under a given error range. The environment of the simulation is shown in Figure 7.8. The simulated robot runs along the desired patrolling path (the big rectangle). It is assumed that the robot has the pose information from GPS/INS/odometry sensors and the equipped laser sensor will supply scan data. Also, the prior map of the environment is known, which is used to obtain the reference scan data.

The robot started from a point on the bottom edge and ran in an anti-clockwise direction. Figures 7.18 to Figure 7.29 show the appearances and disappearances of the multiple pose possibilities during the patrol, decided by the matching algorithm. Each figure has two plots: one is the distribution of particles and one is the PDF plotting.

For test purpose, the multiple pose possibilities are recorded only. Later, a discussion of how to apply these multiple pose possibilities together with other localization functionalities is provided.

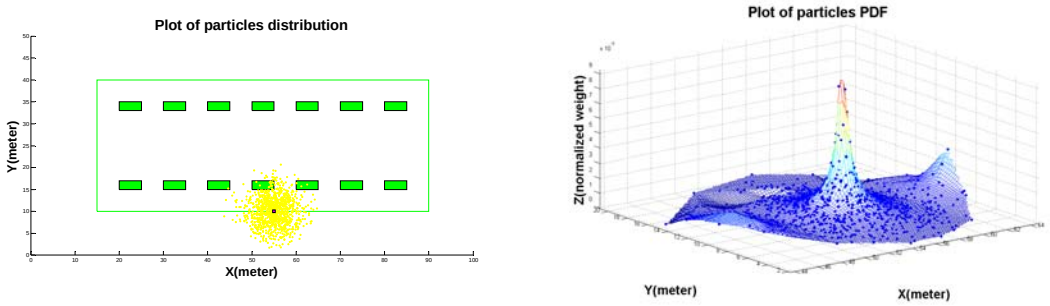


Figure 7. 18 Simulation of particle filtering – initial distribution

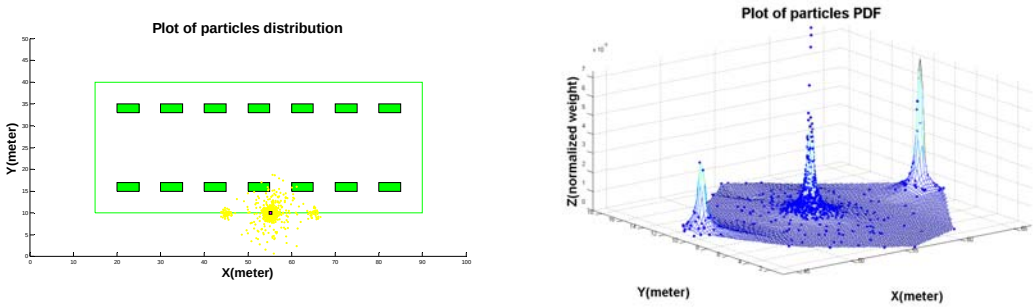


Figure 7. 19 Simulation of particle filtering – re-sampling (1)

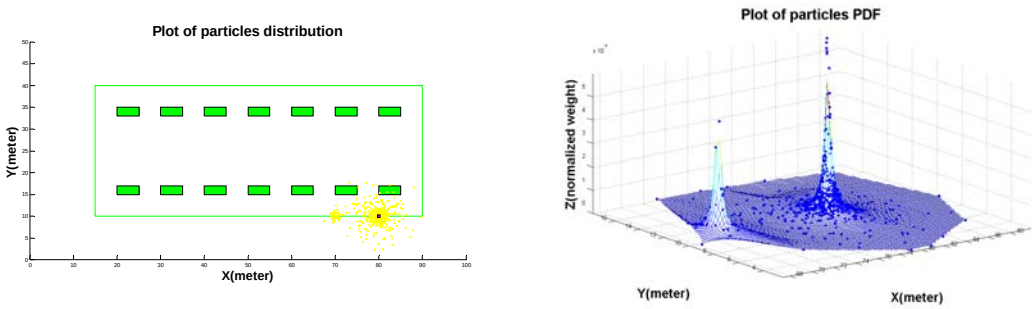


Figure 7. 20 Simulation of particle filtering – re-sampling (2)

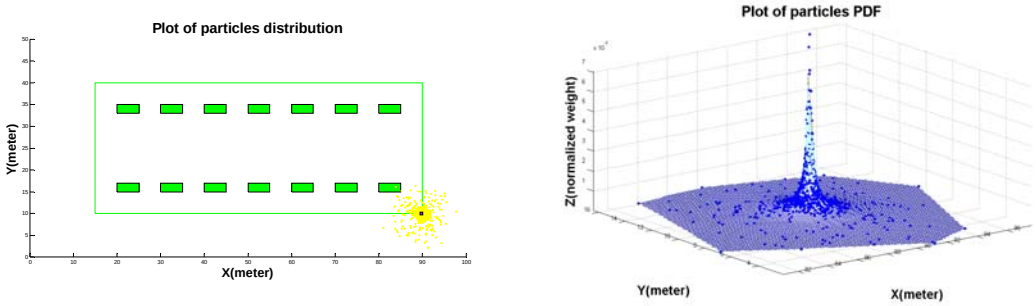


Figure 7. 21 Simulation of particle filtering – re-sampling (3)

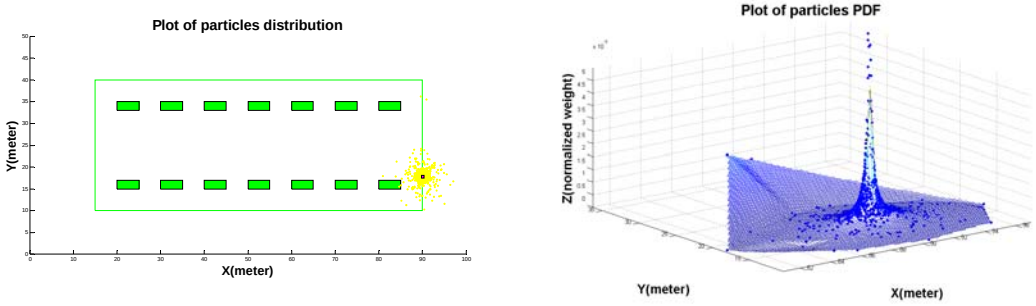


Figure 7. 22 Simulation of particle filtering – re-sampling (4)

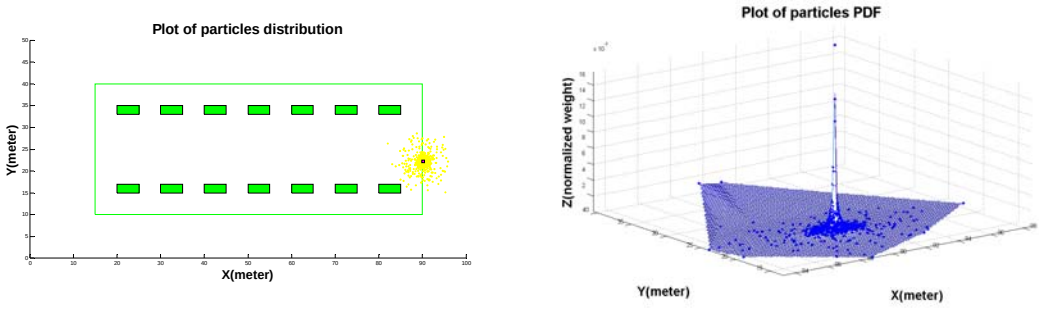


Figure 7. 23 Simulation of particle filtering – re-sampling (5)

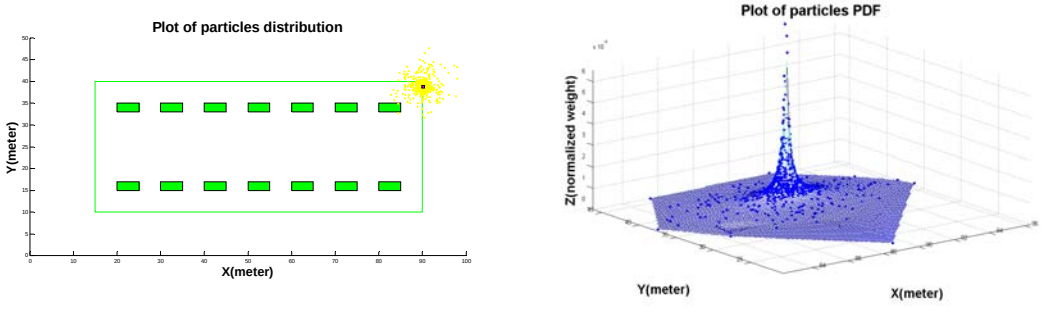


Figure 7. 24 Simulation of particle filtering – re-sampling (6)

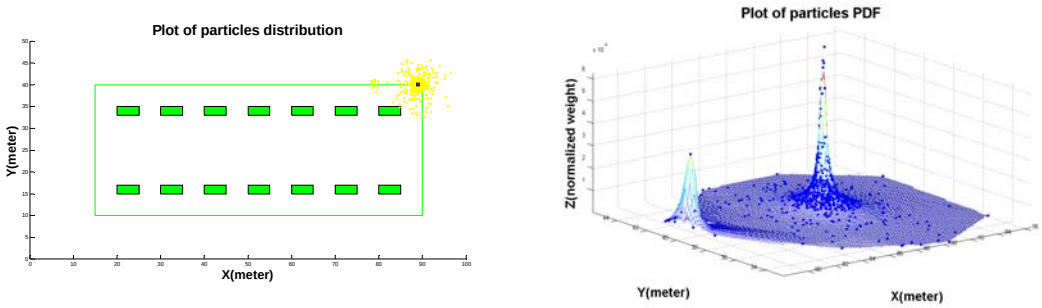


Figure 7. 25 Simulation of particle filtering – re-sampling (7)

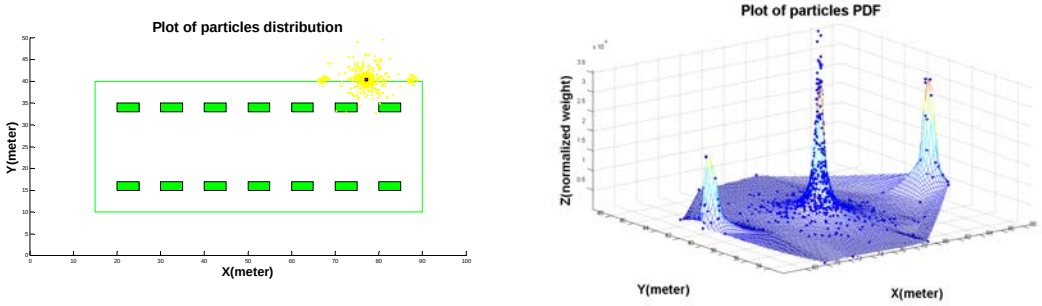


Figure 7. 26 Simulation of particle filtering – re-sampling (8)

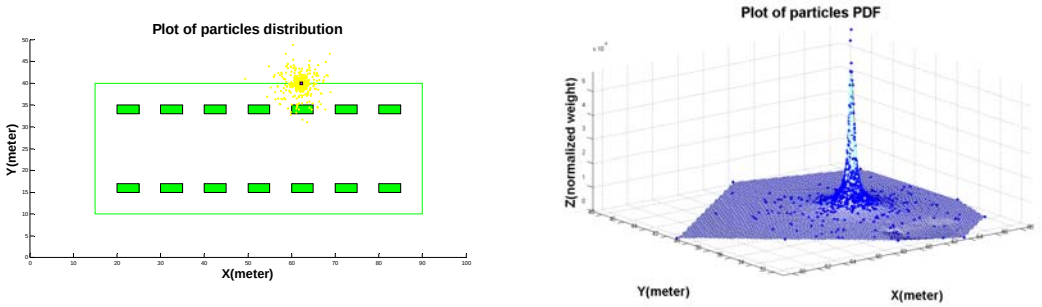


Figure 7. 27 Simulation of particle filtering – re-sampling (9)

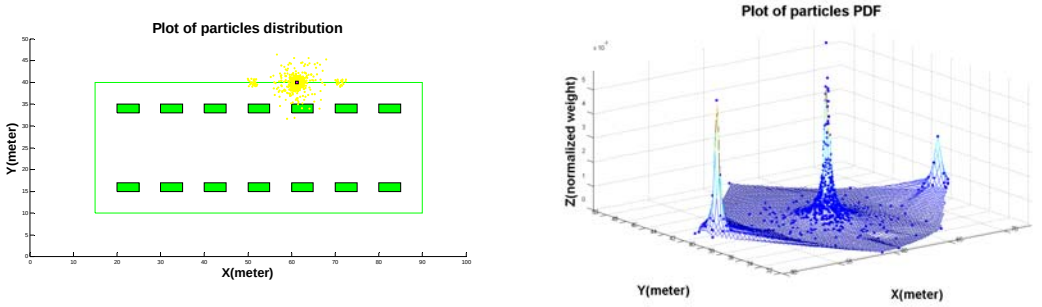


Figure 7. 28 Simulation of particle filtering – re-sampling (10)

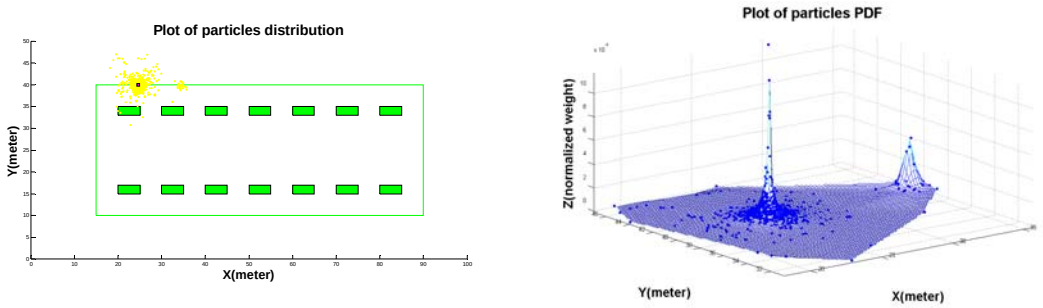


Figure 7. 29 Simulation of particle filtering – re-sampling (11)

The simulation shows that the proposed algorithm works well to find the multiple pose possibilities by the maximum pose error definition. Through the patrol on the given path, the particle filter detects the changing situation of the multiple possibilities. The mechanism is dependent on the matching algorithm. As mentioned before, the matching procedure may fail and there is no pose error information available from matching in this case. When it happens, that particle's weight will be set to zero. This situation is shown in Figure 7.27. Close to this location, there are three possible poses available by the laser scanning, but the matching process failed in the left and right locations due to the additive noise (including the added noise in the reference scan and the simulated scan). So, the particles just caught one possibility. After a few steps, the multiple cases are recovered automatically (see Figure 7.28).

The possible poses are recorded and then have to be applied to localization functionalities. Both of the localization components – Range Based Pose described in chapter 6 and particle filtering discussed here - are methods to enhance the quality of localization. The result from Range Based Pose is the corrected pose from the matching algorithm and can be applied immediately. The results from particle filtering may include multiple possible poses and can not be used directly for the localization function. Only one of the multiple possible poses from particle filtering is the true (close to true) pose. The intention of

using particle filtering here is to keep the multiple possibilities of pose and to recover the real pose when other localization functionalities have failed.

The most important factor which controls the behaviour of this algorithm is the range of the particle distribution. Generally, the number of the particles is decided by the requirements of the application and the computation capability. When the density of the particle distribution is fixed, the greater the number of particles, the larger is the particle distribution. For all the particle algorithms, a greater number of particles always results in a better catch on the probability aspect. For example, if the distribution of the particles in this simulation is enlarged to the entire working space, all the possible pose assumptions will be kept by the algorithm. However, the corresponding huge computation cost is not affordable for the robot system.

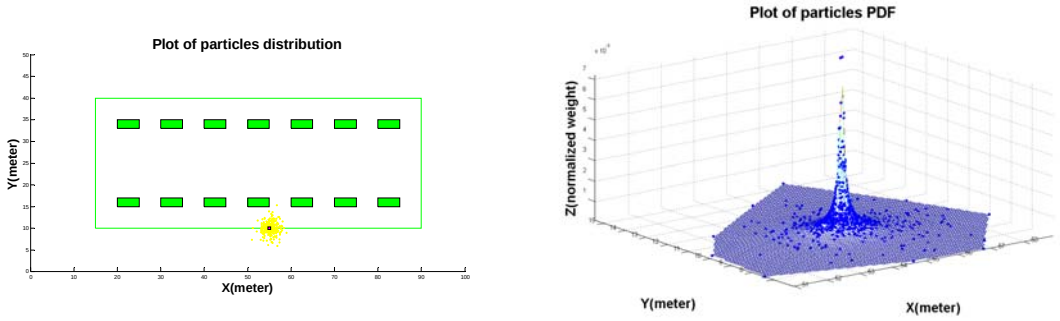


Figure 7.30 Simulation of particle filtering with small distribution – initial distribution

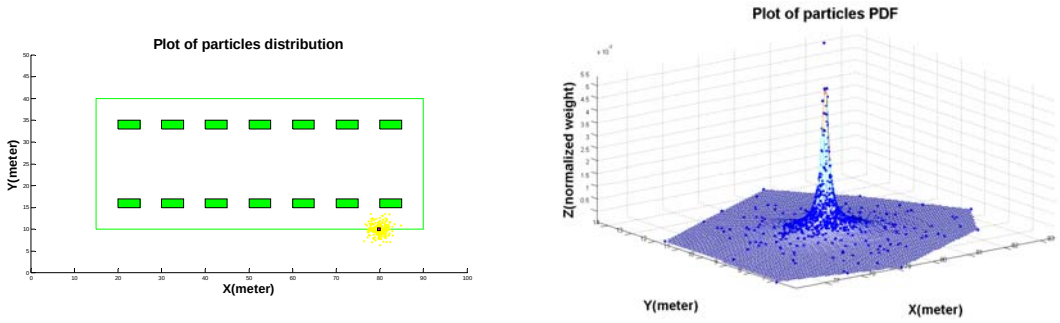


Figure 7.31 Simulation of particle filtering with small distribution – re-sampling (1)

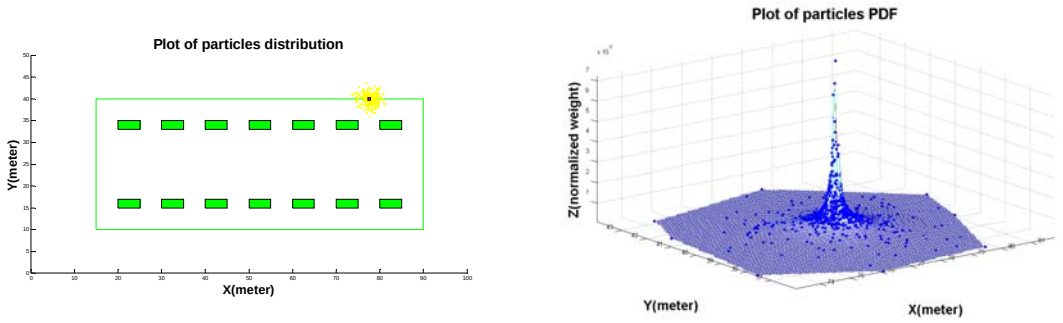


Figure 7.32 Simulation of particle filtering with small distribution – re-sampling (2)

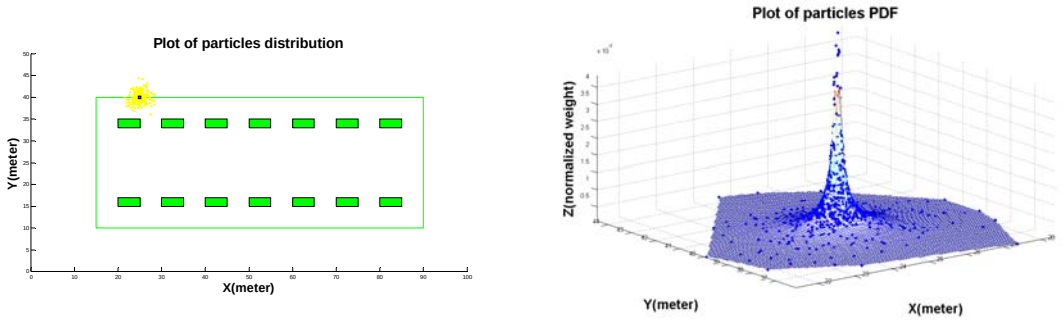


Figure 7.33 Simulation of particle filtering with small distribution – re-sampling (3)

On the other hand, a small distribution area (the small number of particles for the fixed density) can not catch the possible pose assumptions.

In the previous simulation, a 10 *meters* offset is set to the pose of the vehicle which then uses the multivariate Gaussian/normal distribution method [104] to create the particle distribution (see Figure 7.18 to 7.29). To test the affection of the particle distribution range, the offset is reduced to 5 *meters* and the simulation is repeated with this new smaller distribution range.

Figures 7.30 to 7.33 show the particle filtering behaviour with the smaller distribution range. Figure 7.30 shows the initial distribution which corresponds to Figure 7.18. Figure 7.31 shows the distribution after re-sampling which corresponds to Figure 7.20 on the robot's pose. Similarly, Figure 7.32 is to Figure 7.26 and Figure 7.33 is to Figure 7.29. Obviously, the smaller distribution failed to catch the multiple possibilities of the pose.

The same example illustrates the advantage of the particle filtering algorithm against the Kalman Filter type implementation. The Kalman Filter with any particular covariance will end up in computation of the Gaussian distribution which is uni-model. So it will always fail to catch the multiple possibilities in the pose estimation similarly to the case of the small range distribution described above.

So, choosing a proper distribution range is essential to the proposed algorithm. In the simulation case, the objects have the same size and have the same distance among each other. In other words, the feature of the layout is uniform. Considering the multiple layout features, the different distribution ranges would be chosen during the robot running time. Some work would be done in future to develop the algorithm which may decide the distribution range according to the map content and the pose of the robot.

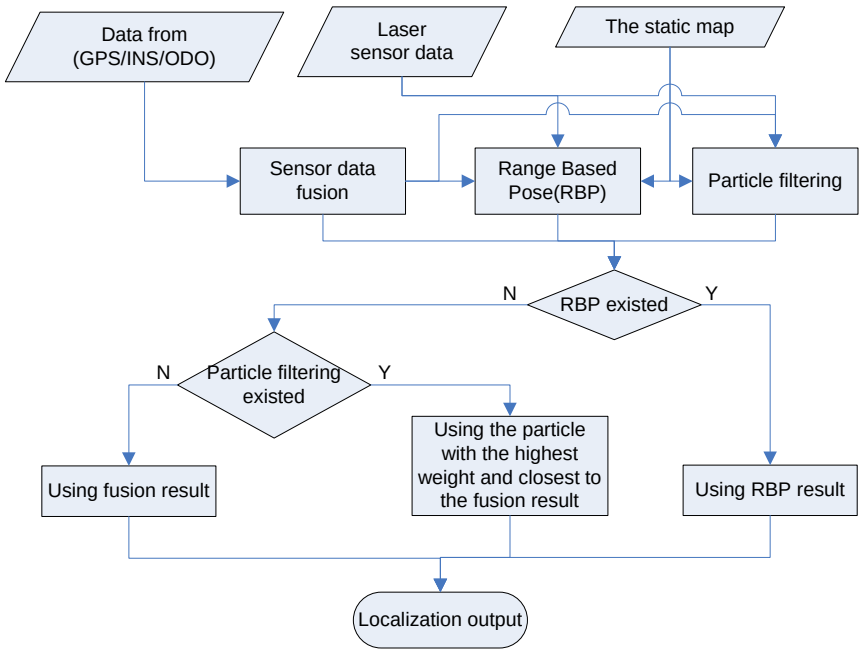


Figure 7. 34 Application of particle filtering for localization functionality

Figure 7.34 is the flowchart of the localization module with three localization functionalities: data fusion, Range Based Pose, and particle filtering, assuming there are

GPS, INS and odometer readings for the elementary localization functionality. The laser range sensor is used for both obstacle detection and localization improvement destinations. The static environment map is known. The data from GPS/INS/ODO is fused by Kalman Filtering to produce pose estimation. The data from the laser scanner is used for Range Based Pose and particle filtering to carry out matching calculations along with the reference scanning from the given map. The estimation from the fusion sub-module is the prior estimation which will be improved by the matching procedure in the Range Based Pose module. As the storage of multiple possibilities, the particle filtering just keeps tracing the possible poses when Range Based Pose works well. When Range Based Pose failed to find the current matching at some location due to the accumulation of sensor noise and drift, the particle filtering will supply the localization module pose estimation from the storage of the possible poses. In the worst case, the estimation from fusion sub-module can be used as output of the localization module when both the Range Based Pose and the particle filtering fail.

The particle filtering algorithm has been tested by simulation and will be integrated into the localization module as a sub-procedure to improve the quality of localization.

Chapter 8
Conclusions and Future Work

8.1 Conclusion

The research presented in this thesis is part of development being done on outdoor mobile robot systems at Carleton University. The framework of the system is constructed using the Joint Architecture of Unmanned Systems (JAUS) standard. The objective of this research work is to develop an autonomous mobile robot with advanced guidance, navigation and control system. Such system can be used for homeland security or defense related applications. This thesis is focused on the development of new methods and algorithms to improve the localization performance for out mobile robots using a 2-D laser sensor. The improvements on localization performance from the developed algorithms have been proved in physical experiments and/or simulations. These algorithms have been implemented in the form of modules or 'component' in JAUS terminology and integrated into the physical mobile robot system. The robot platform with the system has been applied to security tasks.

A special localization problem, gate recognizing and crossing was addressed. Also, the question of how to build and use the geometry and occupancy map for mobile robots was investigated. A JAUS compatible component, the Map Data Server, was developed. To

generalize the solution found for the gate recognizing and crossing problem, a more general data association matching algorithm was developed. This new matching algorithm together with the known map of the environment was shown to improve robot localization. Based on the matching algorithm, two JAUS compatible components - Range Based Pose and Object Tracking - were developed and tested. These components ca perform localization, obstacle detection, and moving object tracking tasks with higher accuracy. Moreover, a proposed algorithm based on particle filtering was developed and tested.

The detailed description of thesis contribution is listed below.

1. For gate recognition and crossing problem, the concept - signature of object, was proposed. A series of signatures were extracted from the pre-defined objects (gate-like object here) according to the extracted features while the object is observed from different view angles – canonical signatures. The scanning data from a 2-D laser range sensor was compared with these canonical signatures. The most similar signature against the scanning was chosen. The results of the comparison were used to find the relative position and angle between the robot (location of the sensor) and the gate objects recognized by the signatures. After the vehicle pose relative to the gate is found, a navigation control point was defined (a point on the middle-axis of the gate-like object). A controller is designed to guide the robot through the gate-like object. Matlab simulation proved that the gate recognition and crossing algorithm works well. Then, the algorithm was adapted from Matlab to standard C code and integrated into the Unmanned Ground Vehicles (UGV) system. The gate crossing

experiments with the real outdoor mobile robot showed that the vehicle recognizes and crosses the gate successfully.

2. The issues related to map building for outdoor mobile robots were addressed, including maps for mobile robot system and map-building problems. An improved polygon clipping algorithm was developed in standard C language to meet the requirements of extracting map information from a particular extents. A JAUS compatible component, the Map Data Server, was developed on a Linux platform. The Map Data Server component provides the map information of the working space. Two types of map data, vector map and occupancy map, are supported. An adaptive multi-level grid map was created to reduce the computation cost and data transferring time. The component was built using C/C++ under Linux platform. The Map Data Server has been integrated into the mobile robot system as a standard module.
3. A laser data association method based on geometry features used for outdoor mobile robot localization improvement sensor was achieved. With an internal global map, a simulated laser scanner is used to supply a reference scanning according to the reported pose of the robot. Three simplifications for the simulated scanning - clipping against the map, visibility check, and angle limitation, were developed to reduce the computation complexity and time cost. A set of algorithms was created to convert the raw outdoor sensor data to geometry features (segments). Using a segment to segment matching algorithm, the errors of vehicle's heading and location are calculated by comparing the reference scanning from the simulator and the real scanning. The validity of the matching result is checked with the simulated scanning applied the errors correction. When searching for the matched segment pairs (one from the

simulating scanning, one from the real scanning), a buffering rectangle is applied to reduce the computation cost and eliminate the chance of wrong matching. The method was tested in simulation and experiments. Both, simulation and experiments proved that this method is very promising for outdoor applications. With the improvement provided by the proposed algorithm, the errors of localization in a large outdoor field could be limited to less than $1°$ in heading and $1.5\,cm$ in location. The algorithm was implemented using Matlab originally and then translated into standard C language.

4. The proposed matching algorithm was used not only for the improvement of localization task, but also for other tasks, like non-map obstacle detection and moving object tracking. Two JAUS compatible components, Range Based Pose and Intruder Tracking, were developed based on the matching algorithm. Range Based Pose (RBP) is a component which finds the error of the current pose estimation using a laser range finder sensor, detects the non-map obstacles, and recognizes the pre-defined objects by size. Object Tracking (OT) tracks the special objects according to the setting from RBP or a higher level command. The strategy for using a range finder sensor in order to detect and track special objects has been proposed. The two components have been integrated into the mobile robot system and tested in many demonstrations.
5. A localization algorithm based on particle filtering was proposed. It may be used as an additional method for localization. The proposed algorithm can find, store, and update the multiple possibilities of the robot's pose when the robot is running in an environment with repetitive features. Each particle is located by location (x and y)

and angle (heading direction θ). The weight of each particle is dependent on the composite offset between the particle and the robot's possible pose. The prediction step is done by a simple motion model. The update step uses the developed matching algorithm to calculate the weight of each particle. It is done by comparing the real scanning data to the reference scanning from the particle's pose. The normalized weights of all particles are regarded as the probability of the particles. During the re-sampling step, the particles with low weight will be deleted and the densities of particles close to the high weights will be increased. An important feature in the re-sampling step from this proposed algorithm is that a defined percentage of particles is always distributed by an initial range during each re-sampling step. It will ensure that the particles can catch and keep the multiple possibilities during the running time. The developed algorithm may be used for localization improvement of the outdoor mobile robots, especially in these environments in which the objects have similar geometry characteristics.

The publications reflecting the results obtained from the thesis are listed in Appendix A.

8.2 Future Work

Many Geodetic Information Systems (GIS) have been developed and applied to Unmanned Ground Vehicles (UGV) applications. The GIS system supplies the various pieces of map information to users and updates the stored information in time. One of the future directions is to develop an interface between the current mobile robot system and

existing GIS system. With such interface, the robot may get the global map of a working space from the GIS by the defined extents. That will give the robot system a wider applicability.

The developed matching algorithm is based on the geometry (vector) level. It is not the original data format from the laser sensor which has raw point outputs. The error in the translation procedure from points to vectors can not be fully avoided. The possible future work on this issue is to develop the data association method based on the grids/points level. That will save the computation time and will reduce the pre-process errors.

Currently, the robot is equipped with only one laser range sensor. It is mounted in the front of the robot with a height of 60 cm and scans horizontally. It can not detect the obstacles lower than its mounting height and can not detect negative obstacles like holes, ditches, or pools on the ground. A second laser sensor (inclined, looking downward) will be used to detect such negative obstacles. To get a good path planning mechanism, a third laser sensor will be used to estimate the traversability of the working space.

For some special objects like fences and vegetation, the laser range sensor can not get stable response. A different type of sensor will be used to work in parallel with the laser sensor (e.g. – camera). The algorithm fusing the laser scanning and the images from a camera has yet to be developed.

Reference

[1] Golfarelli, M. et al, Correction of dead-reckoning errors in map building for mobilerobots, Robotics and Automation, IEEE Transactions on Volume 17, Issue 1, Page(s):37 – 47, Feb 2001.

[2] Borenstein, J., Internal Correction of Dead-reckoning Errors with the Compliant Linkage Vehicle, Journal of Robotic Systems, Page(s):257-273, vol.12, 1995.

[3] Hakyoung Chung Ojeda, L., Borenstein, J., Sensor fusion for mobile robot dead-reckoning with a precision-calibrated fiber optic gyroscope, Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, Volume: 4, Page(s): 3588- 3593 vol.4, 2001

[4] Leonard, J.J.; Durrant-whyte, H.F., Simultaneous map building and localization for an autonomous mobile robot, Intelligent Robots and Systems Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on: Page(s):1442–1447, 1991.

[5] Sasiadek, J., Lu, Y., and Polotski, V., Navigation of autonomous mobile robot with gate recognition and crossing, 8th International IFAC Conference on Robot Control (SYROCO 06), Bologna, Italy, September 2006.

[6] Lu, Y., Sasiadek, J., and Polotski, V., Navigation of autonomous mobile robots – Robot Motion and control 2007, Page(s):187-208, Lecture Notes in Control and Inform Science 360, Springer, London, 2007.

[7] JAUS. (2006). Joint Architecture for Unmanned Systems Reference Architecture, version 3.2: JAUS Working Group (http://www.jauswg.org).

[8] Mobile Robot Platform Design, Carnegie Mellon University, http://www.cs.cmu.edu/~biorobotics/projects/prj_mblplat.html.

[9] Borenstein, J., et al, Mobile robot positioning: Sensors and techniques, Journal of Robotic Systems, Page(s):231-249, vol.14, 1998.

[10] LMS200/211/221/291 Laser Measurement Systems, SICK AG Waldkirch, Germany, 2006.

[11] Castellanos, J.A., et al, The spmap: A probabilistic framework for simultaneous localization and map building, IEEE Trans. on Robotics and Automation, 15(5), Page(s)::948-952, 1999.

[12] Dissanayake, G., Durrant-Whyte, H.F., and Bailey, T., A computationally efficient solution to the simultaneous localization and map building (SLAM) problem, In Proc. IEEE Int. Conf. Robotics and Automation, Page(s):1009-1014, April 2000.

[13] Welch, G., Bishop, G., An introduction to the Kalman Filter, Technical report, University of North Carolina, Department of Computer Science, 1995. Technical Report TR 95-041.

[14] Richter, S., Faschinger, M., Mobile Robot Localization, Technique Report, Graz University of Technology, German, Oct. 2003.

[15] Gowdy, J., Stentz, A., and Herbert, M., Hierarchical Terrain Representations for Off-road Navigation, Proceeding of SPIE – Mobile Robots V, vol. 1388 (1990): Page(s):131-140.

[16] Olin, K., Hughes, D., Autonomous Cross-country Navigation – An Integrated Perception and Planning System, IEEE International Conference on Robotics and Automation (1995), Page(s): 2900-2906.

[17] Nashashibi, F., Devy, M., and Fillatreau, P., Indoor Scene Terrain Modeling Using Multiple Range Images for Autonomous Mobile Robots, Proceeding of the 1992 IEEE International Conference on Robotics and Automation, (1992), Page(s):40-46.

[18] Kweon, I., Kanade, T., High-resolution Terrain Map from Multiple Sensor Data, IEEE Transactions on Pattern Analysis and Machine Intelligence, 14-2(1992), Page(s):278-292.

[19] Arakawa, K., Krotkov, E., Estimating Fractal Dimensions of Natural Terrain From Irregularly Spaced Data, Proceeding of 1993 IEEE/RSJ International Conference of Intelligent Robots and Systems, (1993), Page(s):1364-1370.

[20] Stuck, E., et al, Map Updating and Path Planning For Real Time Mobile Robot Navigation, IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, 1(1994), Page(s):431-438.

[21] Lacroix, S., et al, Autonomous Navigation In Outdoor Environment Adaptive Approach And Experiment, IEEE Conference on Robotics and Automation, 1(1994), Page(s):426-432.

[22] Castellanos, J., TardÓs, J., Laser Based Segmentation And Localization For A Mobile Robot, Proceeding of Sixth International Symposium on Robotics and Manufacturing, (1996), Page(s):101-109.

[23] Hancock, J., Hebert, M., and Thorpe, C., Laser Intensity-Based Obstacle Detection, Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems, (1998), Page(s):1541-1546.

[24] Mázl, R., Přeučil, L., Building a 2D Environment Map from Laser Range-Finder Data, Proceedings of the IEEE Intelligent Vehicles Symposium, (2000), Page(s):290-295.

[25] Cox, I. J., Blanche - An experiment in guidance and navigation of an autonomous robot vehicle. IEEE Transactions on Robotics and Automation, Page(s):193–204, 1991.

[26] Bengtsson, A., Baerveldt, J., Localization in Changing Environments by Matching Laser Range Scans, Third European Workshop on Advanced Mobile Robots, 1999, Page(s):169-176.

[27] Xiang, Z., Liu, J., Initial Localization for Indoor Mobile Robots using a 2D Laser Range Finder, Proceeding 2002 SPIE in Mobile Robot XVI, vol.4573, Douglas W. Gage, Page(s):168-177.

[28] Weber, J., Jorg, K. W., and Puttkamar, E., APR – Global Scan Matching Using Anchor Point Relationships, Proc. 2000 the 6th International Conference on Intelligent Autonomous Systems, Page(s): 471-478.

[29] Gutmaan, J.S., Schlegel, C., AMOS: Comparison of Scan Matching Approaches for Self-Localization in Indoor Environment, Proc. 1996 the 1st Euromicro Workshop in Advanced Mobile Robots, Page(s):61-67.

[30] Jensfelt, P., Christensen, H.I., Laser Based Pose Tracking, Proceeding of the 1999 IEEE International Conference on Robotics and Automation, Detroit, Michigan, 1999.

[31] Thrun, S., Particle filters in robotics. In Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI), 2002.

[32] Adams, M., Zhang, S., and Xie, L., Particle filter based outdoor robot localization using natural features extracted from laser scanners, IEEE International Conference on Robotics and Automation (ICRA '04) , page(s):1493- 1498 Vol.2, 2004.

[33] Karlsson, R., Gustafsson, F., Particle Filter For Underwater Terrain Navigation, Statistical Signal Processing workshop, 2003.

[34] Fox, D., et al, Particle Filters for Mobile Robot Localization, Sequential Monte Carlo Methods in Practice, Springer Verlag, New York, 2000.

[35] Gustafsson, F., et al, Particle filters for positioning, navigation, and tracking, IEEE Transaction on Signal Processing, (Vol50-2), Page(s):425-437, Feb 2002.

[36] Kim, S.-J., Iltis, R.A., Performance Comparison of Particle and Extended Kalman Filter Algorithms for GPS C/A Code Tracking and Interference Rejection, 2002 Conference on Information Sciences and Systems, Princeton University, March, 2002

[37] Milstein, A., Sánchez, J.N., and Williamson, E.T., Robust Global Localization Using Clustered Particle Filtering, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007), 2007.

[38] Grisetti, G., Stachniss, C., and Burgard, W., Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2005), page(s): 2432- 2437, 2005.

[39] JAUS Working Group, Domain Model (DM), Volume I, Version 3.2, The Joint Architecture For Unmanned Systems, March 2005.

[40] JAUS Working Group, Reference Architecture Specification, Volume II, Part 1, Version 3.2, The Joint Architecture For Unmanned Systems, August 2004.

[41] JAUS Working Group, Reference Architecture Specification, Volume II, Part 2, The Joint Architecture For Unmanned Systems, August 2004.

[42] JAUS Working Group, Reference Architecture Specification, Volume II, Part 3, The Joint Architecture For Unmanned Systems, August 2004.

[43] Evans III, C.P., Development of a Geospatial Dada Sharing Method for Unmanned Vehicles Based on the Joint Architecture for Unmanned Systems (JAUS), Master Thesis, University of Florida, 2005.

[44] JAUS Working Group, Reference Architecture, Proposed Addendum, World Model Knowledge Store Components, The Joint Architecture For Unmanned Systems, October 2005.

[45] Crane, C., et al, Team CIMAR's NaviGATOR: An Unmanned Ground Vehicle for Application to the 2005 DARPA Grand Challenge, Journal of Field Robotics, 2006

[46] Touchton, R., et al, Planning and modeling extensions to the Joint Architecture for Unmanned Systems (JAUS) for application to unmanned ground vehicles, SPIE Defense and Security Symposium, Orlando, Mar 2005.

[47] Steven, J., et al, Vision Based Vehicle Localization for Autonomous Navigation, Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation, Jacksonville, FL, USA, June 20-23, 2007.

[48] Gothing, G., et al, Implementation of JAUS on a 2004 Cadillac SRX Using a Potential Fields Architecture, Project Report, Department of Mechanical Engineering, Virginia Technique Institute, 2007.

[49] Ye, C., Borenstein, J., Characterization of a 2-D Laser Scanner for Mobile Robot Obstacle Negotiation, Proceedings of the 2002 IEEE ICRA, Washington DC, USA, 10-17 May 2002, Page(s):2512-2518.

[50] Vale, A., Lucas, J.M., and Ribeiro M.I., Feature Extraction and Selection for Mobile Robot Navigation in Unstructured Environments, Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles, Lisbon, 5 - 7 July 2004.

[51] An, D., Wang, H., VPH: A new laser radar based obstacle avoidance method for intelligent mobile robots, Intelligent Control and Automation, 2004, Fifth World

Congress on Volume 5, Hangzhou, China, 15-19 June 2004, Page(s):4681 - 4685 Vol.5.

[52] Sasiadek, J.Z. and Wang, Q., Low cost automation using INS/GPS data fusion for accurate positioning, Robotica *(2003)* volume 21, Page(s):255-260. 2003.

[53] Jansfelt, P., Christensen, H. I., Path Tracking using Laser Scanning and Minimalistic Environment Model, IEEE Transaction on Robotics and Automation, Vol. 17, No.2, 2001.

[54] Egido, V., et al, A door Lintel Locator Sensor for Mobile Robot Topological Navigation, IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Application, 2005.

[55] Amir, E., Maynhard, P., Door Identification, Project Report, Stanford University, http://www.formal.stanford.edu/eyal/cs223b/report.html, 1999.

[56] Lazkano, E., et al, Door Crossing Behavior for a Mobile Robot using Bayesian Networks, Proceeding of CIMCA, 2003.

[57] Stoeter, S. A., et al, Real Time Door Detection in Cluttered Environments, Proceedings of IEEE Int. Symp. On Intelligent Control, 2000.

[58] Bakambu, J.N., et al, Heading-Aided Odometry and Range data Integration for Positioning of Autonomous Mining Vehicles, Proceedings of the IEEE Int. Conf. on Control Applications, Anchorage AK, Sept. 2000.

[59] Roberts, J. et al, Autonomous Control of Underground Mining Vehicles using Reactive Navigation, Proceedings of the Int. Conference on Robotics and Automation, San Francisco, April 2000.

[60] Fernandez R., et al, Recognition of Contexts with Scanner-laser for Mobile Robot Navigation, Proceedings of the 9th international symposium on intelligent robotic systems : Page(s):507-515, Toulouse, France, July 2001.

[61] Pfister, S.T. Burdick, J.W., Multi-scale Point and Line Range Data Algorithms for Mapping and Localization, Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, Page(s):1159-1166, Orlando, FL, May 2006.

[62] Akca, D., Full Automatic Registration of Laser Scanner Point Clouds, Optical 3-D Measurement Techniques VI, vol. I, Page(s):330-337. Zurich, Switzerland, September 2003,

[63] Gumhold, S., et al, Feature Extraction From Point Clouds, 10th International Meshing Roundtable, Newport Beach, California, U.S.A. October, 2001.

[64] Astolfi, A., Exponential Stabilization of a Wheeled Mobile Robot via Discontinuous Control, Journal of Dynamical Systems Measurements and Control, 1999 V.121, Page(s):121-125.

[65] Goel, P., Roumeliotis, S.I., and Sukhatme, G.S., Robot Localization using Relative and Absolute Position Estimates, In Proc IEEE Int. Conference on Robots and Systems 1999, October, Kyongju, Korea.

[66] Pfister, S.T., et al, Weighted Range Sensor Matching Algorithms for Mobile Robot Displacement Estimation, In Proc. 2002 IEEE Int. Conference on Robotics and Automation, 2002 May, Washington DC

[67] Angelopoulou, E., et al, World Model Representations for Mobile Robots, Intelligent Vehicles '92 Symposium, Proceedings of the, Page(s):293-297, Detroit, MI, USA, July 1992.

[68] Broten, G., et al, World Representations Using Terrain Maps, Technical Report TR 2005-248, Defence Research and Development Canada, December 2005.

[69] Maps in Computers, Class Notes, Department of Geography, San José State University, www.sisu.edu/depts/geography/classes/geo170/taketa/g170stor.doc.

[70] Zhang, X.Z., et al, A Comparative Study of Three Mapping Methodologies, Journal of Intelligent Robot Systems (2007), (49), Page(s):385–395.

[71] Priester S., Polygon Clipping, http://www.codeguru.com/cpp/misc/misc/graphics/article.php/c8965, 2005.

[72] Hsieh, H., Computer Graphics – Clipping, http://www.cc.gatech.edu/grads/h/Hao-wei.Hsieh/Haowei.Hsieh/mm.html, 1995.

[73] Nguyen, V., et al, A Comparison of Line Extraction Algorithms using 2D Laser Rangefinder for Indoor Mobile Robotics, Proceedings of the 2005 Intelligent Robots and Systems IROS, 2005.

[74] Xu, Z., et al, Z., Map Building and Localization using 2D Range Scanner, Proceedings 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation, July 16-20, 2003, Kobe, Japan

[75] Xiang, Z., Liu, J., Initial Localization for Indoor Mobile Robots using a 2D Laser Range Finder, Proceeding 2002 SPIE vol.4573 mobile robot XVI, Douglas W. Gage, Page(s):168-177

[76] Lu, F., Milios, E.E., Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans, Technical Report, York University, 1995

[77] Weber, J., Jorg, K.W., Puttkamar, E., APR – Global Scan Matching Using Anchor Point Relationships, Proc. 2000 the 6th International Conference on Intelligent Autonomous Systems, Page(s):471-478

[78] Gutmaan, J.S., Schlegel, C., AMOS: Comparison of Scan Matching Approaches for Self-Localization in Indoor Environment, Proc. 1996 the 1st Euromicro Workshop in Advanced Mobile Robots, Page(s):61-67

[79] Jensfelt, P., Christensen, H.I., Laser Based Pose Tracking, Proceeding of the 1999 IEEE International Conference on Robotics and Automation, Detroit, Michigan, 1999

[80] Skrzypczynski, P., Building Geometrical Map of Environment using ID Range Finder Data, Intelligent Autonomous Systems, Page(s):408-412, 1995.

[81] Borges, G.A., Aldon, M., Line Extraction in 2D Range Images for Mobile Robotics, Journal of Intelligent and Robotic Systems, vol.40, Page(s):267-297, 2004.

[82] Forsyth, D.A. Ponce, J.. Computer Vision: A Modern Approach, Prentice Hall, 2003.

[83] Bengtsson, O., Baerveldt, A.J., Localization in Changing Environments by Matching Laser Range Scans, 3rd European Workshop on Advanced Mobile Robots, 1999, Page(s):169-176

[84] Differential Global Positioning System, Australian Maritime Safety Authority, http://www.amsa.gov.au/Shipping_Safety/Navigation_Safety/Differential_Global_Postitioning_System/DGPS_Fact_Sheet.asp.

[85] Rekleitis, I.M., Cooperative Localization and Multi-Robot Exploration. PhD thesis, School of Computer Science, McGill University, Quebec, Canada, February 2003.

[86] Rekleitis, I.M., A Particle Filter Tutorial for Mobile Robot Localization. Technical Report TR-CIM-04-02, Centre for Intelligent Machines, McGill University, Quebec, Canada, 2004.

[87] Gelb, A., Applied Optimal Estimation. MIT Press, Cambridge, Massachusetts, 1974.

[88] Bozic, S.M., Digital and Kalman Filtering. Edward Arnold, second edition, 1994.

[89] Maybeck, P., Stochastic Models, Estimation and Control, volume 1, Academic, New York, 1979.

[90] Smith, R., Cheeseman, P., On the Representation and Estimation of Spatial Uncertainty. International Journal of Robotics Research, 5(4), Page(s):56-68, 1986.

[91] Smith, R., etc. Estimating Uncertain Spatial Relationships in Robotics. In I. J. Cox and G. T. Wilfong, editors, Autonomous Robot Vehicles, Page(s):167 – 193, Springer-verlag, 1990.

[92] Curran, A., Kyriakopoulos, K.J., Sensor-based Self-localization for wheeled mobile robots. Journal of Robotic Systems, 12(3), Page(s):163-176, March 1995.

[93] Leonard, J.J., Durrant-Whyte, H.F., Mobile Robot Localization by Tracking Geometric Beacons. IEEE Transactions on Robotics and Automation, 7(3), Page(s):376-382, June 1991.

[94] Roumeliotis, S. I., Bekey, G. A., Collective Localization: a Distributed Kalman Filter Approach to Localization of Groups of Robots. In Proc. 2000 IEEE International Conference on Robotics and Automation, Page(s):2985-2992, San Francisco, California, April 2000.

[95] Kurazume, R., Nagata, S., Cooperative Positioning with Multiple Robots. In International Conference in Robotics and Automation, vol. 2, Page(s):1250-1257, 1994.

[96] Dellaert, F., Stroupe, A., Linear 2D Localization and Mapping for Single and Multiple Robots. In Proceeding of the IEEE International Conference on Robotics and Automation, May 2002.

[97] Isard, M., Blake, A., Condensation-conditional Density Propagation for Visual Tracking. International Journal of Computer Vision, 29(1), Page(s):2-28,1998.

[98] Yaqub, T., Katupitiya, J., laser scan matching for measurement update in a particle filter, Advanced Intelligent Mechatronics, IEEE/ASME International Conference on, Page(s):1-6, Zurich, Sept. 2007.

[99] Hsaio, K., et al, Particle Filters and applications, Cognitive Robotics, MIT, 2005. http://web.mit.edu/16.412j/www/html/Advanced%20lectures/Slides/Hsaio_plinval_miller_ParticleFiltersPrint.pdf

[100] Borenstain, J., et al, Navigating Mobile Robot: System and Techniques. Number ISBN 1-56881-058-X. A K Peters, Willesley, MA, 1996.

[101] Borenstain, J., Feng, L., Measurement and Correction of Systematic Odometry Errors in Mobile Robots. IEEE Transactions on Robotics and Automation, 12(6), Page(s):869-880, 1996.

[102] Liu, J.S., et al, A Theoretical Framework for Sequential Importance Sampling and re-sampling. Sequential Monte Carlo in Practice. Springer-Verlag, 2001.

[103] Carpenter, J., et al, An Improved Particle Filter for non-linear Problems. IEEE Proceedings – Radar, Sonar and Navigation, 146, Page(s):2-7, 1999.

[104] Cox, D.R., Small, N.J.H., Testing multivariate normality. *Biometrika* 65 (2), Page(s): 263–272, 1978.

[105] Dellaert, F., et al, Monte Carlo Localization for Mobile Robots. In IEEE International Conference on Robotics and Automation, May 1999.

[106] Specifications of ARGO vehicle, Ontario Drive and Gear Ltd., http://www.odg.com/odg.

[107] Lu, Y., Polotski, V., Sasiadek, J., Outdoor Mobile Robot Localization with 2-D Laser Range Sensor, Proceedings of the Tenth IASTED International Conference, 622-803, May 26-28, Quebec City, Quebec, Canada, 2008.

[108] Polotski, V., Sasiadek, J., Lu, Y., Gate Recognition and Crossing by an Autonomous Security Robot, 37th International Symp. on Robotics (ISR 2006), Munich, Germany, May 2006.

Appendix A: Publications of Thesis work

[1] Lu, Y., Sasiadek, J., Design of the Control System of LHD Vehicles, the International Federation of Automatic Control (IFAC), 2005.

[2] Sasiadek, J., Lu, Y., Polotski, V., Navigation of autonomous mobile robot with gate recognition and crossing, 8th International IFAC Conference on Robot Control (SYROCO 06), Bologna, Italy, September 2006.

[3] Polotski, V., Sasiadek, J., Lu, Y., Gate recognition and crossing by an autonomous security robot, 37th International Symp. on Robotics (ISR 2006), Munich, Germany, May 2006.

[4] Sasiadek, J., Lu, Y., Polotski, V., Navigation of autonomous mobile robots – Robot Motion and control 2007, 187-208, Lecture Notes in Control and Inform Science 360, Springer, London, 2007.

[5] Lu, Y., Polotski, V., Sasiadek, J., Outdoor Mobile Robot Localization with 2-D Laser Range Sensor, Proceedings of the Tenth IASTED International Conference, 622-803, May 26-28, Quebec City, Quebec, Canada, 2008.

[6] Lu, Y., Polotski, V., Sasiadek, J., Particle Filtering with Range Data Association for Mobile Robot Localization in Environments with Repetitive Geometry, Robot Motion and Control, 2009.

Appendix B: Specifications of the Robot Platform – ARGO Conquest 6x6 from ODG [106]

Engine	4 cycle OHV V-Twin gasoline engine,electronic ignition,full pressure lubrication and oil filter, 2 year warranty
Model	Briggs & Stratton Vanguard
Horsepower	23
Displacement	627cc
Cooling	Air cooled
Starting	Electric/Recoil
Brakes	Hydraulic vented disc brakes
Steering	One-piece ergonomic handlebar steering control with mounted brake lever and emergency/parking brake
Controls	Right-hand twist grip throttle and dash mounted choke control. Light switch and ignition switch.
Clutch & Transmission	Belt-driven,Continuously Variable Transmission (CVT) maximizes engine power to the transmission with high and low range forward,neutral and reverse,compactly housed with an efficient planetary differential.
Drive System	Roller chains drive machined sprockets that are spline-fit onto 1.25" diameter axles.Bearings are greaseable and are protected with double-sealed outer flange assemblies.
Frame	Formed steel channel construction,welded for high strength and durability. Polyester powder coated for lasting protection.
Body	Vacuum formed High Density Polyethylene (HDPE)
Full Skid Plate	Standard
Load Capacity on Land (without accessories)	700 lbs/ 317 kg total - 140 lbs/ 63 kg max in rear compartment

Towing Capacity	1400 lbs/635 kg
Seating Capacity	4 persons on land, 2 persons on water
Fuel Capacity	7.1 US Gallons (27 litres) See through polyethylene fuel tank. 8 hours of operation based on load and operating conditions.
Speed (on land)	22 mph / 35 km per hour
Speed (on water)	3.5 mph / 5 km per hour
Shipping Weight	890 lbs/404 kg
Axle Bearing Extensions	Front and Rear Standard
LCD Digital Gauge Cluster	Speed, Distance, Voltmeter, Low Oil Pressure and Parking Brake Reminder Lights
Tire	Argo 24 x 10.00-8 NHS
Ground Pressure	2.1 psi (14.5 kPa) using tires, 0.67 psi (4.6 kPa) using tracks
Ground Clearance Tires	9.5 inches - 240 mm
Ground Clearance Tracks	10.5 inches - 265 mm
Operating Conditions	All weather,all-terrain,-40°C to +40°C

Appendix C: JAUS Message Set for MapDataServer

Add object(s) in object map

Field #	Name	Type	Units	Interpretation
1	Local Request ID	Byte	N/A	Local request identifier
2	Number of objects	Unsigned short integer	N/A	1-…
3	Object 1 Name	Byte	N/A	Object name
4	Object 1 Type	Unsigned short integer	N/A	Enumeration 0: Point 1: Line 2: Polygon
5	Object 1 Number of Features	Byte	N/A	
6	Object 1 Feature 1	Varies	Varies with features	
…	…	…	…	…
	Object 1 Number of Point	Unsigned short integer	N/A	
	Object 1 Point 1 Latitude	Double	Degrees	
	Object 1 Point 1 Longitude	Double	Degrees	
…	…	…	…	…
	Object 1 Point n Latitude	Double	Degrees	
	Object 1 Point n Longitude	Double	Degrees	
…	…	…	…	…
	Object p Name	Byte	N/A	Object name

	Object p Type	Unsigned short integer	N/A	Enumeration 0: Point 1: Line 2: Polygon
	Object p Buffer	Float	Meters	
	Object p Number of Features	Byte	N/A	
	Object 1 Feature 1	Varies	Varies with features	
…	…	…	…	…
	Object p Number of Point	Unsigned short integer	N/A	
	Object p Point 1 Latitude	Double	Degrees	
	Object p Point 1 Longitude	Double	Degrees	
…	…	…	…	…

Delete object in object map

Field #	Name	Type	Units	Interpretation
1	Local Request ID	Byte	N/A	Local request identifier
2	Object ID	Byte	N/A	Object identifier
3	Object Name	Byte	N/A	

Modify attribute of a special object

Field #	Name	Type	Units	Interpretation
1	Local Request ID	Byte	N/A	Local request identifier
2	Object ID	Byte	N/A	Object identifier
3	Object Name	Byte	N/A	
4	Number of Modification	Byte	N/A	
5	Name of Attribute 1	Byte	N/A	
6	New Attribute 1	Byte	N/A	

…	…	…	…	…
	Name of Attribute n	Byte	N/A	
	New Attribute n	Byte	N/A	

Query vector data with specified parameters (extents, types of objects)

Field #	Name	Type	Units	Interpretation
1	Local Request ID	Byte	N/A	Local request identifier
2	Map Extent in Latitude	Double	Degrees	
3	Map Extent in Longitude	Double	Degrees	
4	Base Point Latitude	Double	Degrees	
5	Base Point Longitude	Double	Degrees	

Response to vector data query

Field #	Name	Type	Units	Interpretation
1	Local Request ID	Byte	N/A	Local request identifier
2	Number of Objects	Byte	N/A	
3	Attributes of Object 1	Byte	N/A	Text description for the attributes
4	Number of Vectors of Object 1 in Extent	Byte	N/A	
5	Attributes of Vector 1	Byte	N/A	
6	Latitude of Start/Intersection Point from Vector 1	Byte	N/A	
7	Longitude of Start/Intersection Point from Vector 1	Byte	N/A	

8	Latitude of End/Intersection Point from Vector 1	Byte	N/A	
9	Longitude of End/Intersection Point from Vector 1	Byte	N/A	
…	…	…	…	…
	Attributes of Vector n	Byte	N/A	
	Latitude of Start/Intersection Point from Vector 1	Byte	N/A	
	Longitude of Start/Intersection Point from Vector 1	Byte	N/A	
	Latitude of End/Intersection Point from Vector 1	Byte	N/A	
	Longitude of End/Intersection Point from Vector 1	Byte	N/A	
…	…	…	…	…

Query occupancy data with specified Resolution

Field #	**Name**	**Type**	**Units**	**Interpretation**
1	Local Request ID	Byte	N/A	Local request identifier
2	Map Extent in Latitude	Double	Degrees	
3	Map Extent in Longitude	Double	Degrees	
4	Base Point Latitude	Double	Degrees	
5	Base Point Longitude	Double	Degrees	
6	Required Resolution	Short Integer	Meters	

Response to occupancy data query

Field #	Name	Type	Units	Interpretation
1	Local Request ID	Byte	N/A	Local request identifier
2	Map Extent in Latitude	Double	Degrees	
3	Map Extent in Longitude	Double	Degrees	
4	Base Point Latitude	Double	Degrees	
5	Base Point Longitude	Double	Degrees	
6	Number of Different Resolution Level	Short Integer	N/A	
7	Number of Row in Resolution 1	Short Integer	N/A	
8	Number of Column in Resolution 1	Short Integer	N/A	
9	Cell Size in Resolution 1	Short Integer	Meters	
10	Unoccupied Cells in Resolution 1	Short Integer	N/A	{(r1,c1),(r2,c2)…}
11	Number of Occupied Cells	Short Integer	N/A	
12	Occupied Cells in Resolution 1	Short Integer	N/A	{(R1,C1),(R2,C2) …}

Appendix D: JAUS Message Set for RangeBasedPose

Set intruder found message

Field #	Name	Type	Units	Interpretation
1	Local Request ID	Byte	N/A	Local request identifier
2	Intruder Latitude	Double	Degrees	
3	Intruder Longitude	Double	Degrees	

Set tracking message

Field #	Name	Type	Units	Interpretation
1	Local Request ID	Byte	N/A	Local request identifier
2	Tracking mode	Unsigned short integer	N/A	1: tracking enable 0: tracking disable
3	Intruder Latitude	Double	Degrees	
4	Intruder Longitude	Double	Degrees	

Set obstacles message

Field #	Name	Type	Units	Interpretation
1	Local Request ID	Byte	N/A	Local request identifier
2	Number of segments in obstacles	Unsigned short integer	N/A	1-…
3	Segment 1 Start point Latitude	Double	Degrees	

4	Segment 1 Start point Longitude	Double	Degrees	
5	Segment 1 End point Latitude	Double	Degrees	
6	Segment 1 End point Longitude	Double	Degrees	
…	…	…	…	…
…	Segment #n Start point Latitude	Double	Degrees	
…	Segment #n Start point Longitude	Double	Degrees	
…	Segment #n End point Latitude	Double	Degrees	
…	Segment #n End point Longitude	Double	Degrees	

Set global pose message

Field #	**Name**	**Type**	**Units**	**Interpretation**
1	Local Request ID	Byte	N/A	Local request identifier
2	Robot Latitude	Double	Degrees	
3	Robot Longitude	Double	Degrees	
4	Robot heading angle	Double	Degrees	

Appendix E: JAUS Message Set for ObjectTracking

Report element message

Field #	Name	Type	Units	Interpretation
1	Local Request ID	Byte	N/A	Local request identifier
2	To way point Latitude	Double	Degrees	
3	To way point Longitude	Double	Degrees	

Report open space path plan message

Field #	Name	Type	Units	Interpretation
1	Local Request ID	Byte	N/A	Local request identifier
2	Task ID	Unsigned short integer	N/A	ID of planned task
3	Plan count	Unsigned short integer	N/A	Number of segments in the plan
4	Number 1 to way point Latitude	Double	Degrees	
5	Number 1 to way point Longitude	Double	Degrees	
6	Number 2 to way point Latitude	Double	Degrees	
7	Number 2 to way point Longitude	Double	Degrees	
…	…	…	…	…
…	Number n to way point Latitude	Double	Degrees	
…	Number n to way point Longitude	Double	Degrees	

Appendix F: Selected Codes

Prototype of RangeBasedPose

```
function RBP = RangeBasePose(OBJECTS_MAP,NUM_OBJECTS,SENSOR_POINTS,...
                    NUM_SENSOR_POINTS,NUM_SENSOR_OBS,INDEX_SENSOR_POINTS,POSE)
% locate the real pose of vehicle by matching the point from scanner and
% the segments from map scanning
% Inputs:
%    OBJECTS, see description in GLOBAL_OBJECTS.m
%    NUM_OBJECTS, number of objects in current map(in a given area)
%    SENSOR_POINTS, points from laser scanner
%    NUM_SENSOR_POINTS, number of points from laser sensor
%    INDEX_SENSOR_POINTS, index, for real case, it would be in the sensor
%                    data from range sensor
%    Estimated POSE, (x,y,theta)
% Outputs:
%    RBP structure:
%       tag, if successful -  1;
%           otherwise     - -1 - too few points; -2 - unequal object numbers
%                       -3 - failed matching;
%       pose, new estimated POSE (x,y,theta)
%       delta, (delta x, delta y, delta theta)
%       d,    average distance from each point to the matched segments
%       num_segs, number of matched segments
%       num_obs,  number of matched objects

global TRANS_MAX
global ROTATION_MAX
global num_total
global fig_num
global max_range
global res
global TRANS_MAX
global ROT_MAX
global ob
global MAX_NUM_POSSIBILTIES

RBP.tag = 0;
RBP.pose = POSE;
RBP.d = 1000;
RBP.num_segs = 0;
RBP.num_obs = 0;

% map scanning
```

```
MAP_SCANNING=LMS_SIMULATOR_MAP(POSE(1),POSE(2),POSE(3),OBJECTS_MAP,NUM_OBJ
ECTS);
if (MAP_SCANNING.Npoints <= 3 | NUM_SENSOR_POINTS <= 3)% too few points to continue
   RBP.tag = -1;
   return;
end

% if object numbers from map and sensor are different, don't do matching
if (MAP_SCANNING.Nobs ~= NUM_SENSOR_OBS)
   RBP.tag = -2;
   return;
end

points_sesor_1 = zeros(NUM_SENSOR_POINTS,2);
points_sensor_1 = SENSOR_POINTS(INDEX_SENSOR_POINTS(1,1):INDEX_SENSOR_POINTS(1,2),
4:5);
for i = 2 : NUM_SENSOR_OBS
   points_sensor_1 = [points_sensor_1;
[SENSOR_POINTS(INDEX_SENSOR_POINTS(i,1):INDEX_SENSOR_POINTS(i,2), 4:5)]];
end

seg = zeros(MAP_SCANNING.Nseg,4);
for i = 1:MAP_SCANNING.Nseg
   seg(i,:) = [MAP_SCANNING.Seg(i).Start(1),MAP_SCANNING.Seg(i).Start(2),...
            MAP_SCANNING.Seg(i).End(1),MAP_SCANNING.Seg(i).End(2)];
end

[deltaX, deltaY, deltaTHETA, D]=matching(seg, MAP_SCANNING.Nseg, points_sensor_1,
NUM_SENSOR_POINTS,fig_num-1);

if D ~= -100
   RBP.pose(1)  = POSE(1)*cos(-deltaTHETA) - POSE(2)*sin(-deltaTHETA) - deltaX;
   RBP.pose(2)  = POSE(1)*sin(-deltaTHETA) + POSE(2)*cos(-deltaTHETA) - deltaY;
   RBP.pose(3)  = angle_wrap(POSE(3) - deltaTHETA);
   RBP.delta(1) = -deltaX;
   RBP.delta(2) = -deltaY;
   RBP.delta(3) = -deltaTHETA;
   RBP.d        = D;
   RBP.num_segs = MAP_SCANNING.Nseg;
   RBP.num_obs  = MAP_SCANNING.Nobs;
   RBP.tag      = 1;
else
   RBP.tag = -3;
end

%**************************************************************************

function [localX, localY] = trans_coordinates(globalX, globalY, transX, transY, rot)
% transform from Global coordinates to local coordinates
% global origin: (0,0)
% local origin: (transX, transY), rotate rot (in rad)
% translation at first, then rotation
% to get right playback, need to do rotation then translation

%warning: clockwise is -, counterclockwise is +
```

```
localX = (globalX + transX)*cos(rot) - (globalY + transY)*sin(rot);
localY = (globalX + transX)*sin(rot) + (globalY + transY)*cos(rot);

%*************************************************************************

function [p2, p3]=rotation_create(p0,p1,angle)

p2=p1*sin(angle)+p0*cos(angle);
p3=p1*cos(angle)-p0*sin(angle);

%*************************************************************************

function EQs=points_EQ_REC(p1,p2,p3,p4)
% create the four equations for the rectangle object from the given four
% points

EQs=zeros(1,25);
% equation 1 from point 1 and 2
A1=p2(2)-p1(2);
B1=-(p2(1)-p1(1));
C1=p1(2)*(p2(1)-p1(1))-p1(1)*(p2(2)-p1(2));
% equation 2 from point 2 and 3
A2=p3(2)-p2(2);
B2=-(p3(1)-p2(1));
C2=p2(2)*(p3(1)-p2(1))-p2(1)*(p3(2)-p2(2));
% equation 3 from point 3 and 4
A3=p4(2)-p3(2);
B3=-(p4(1)-p3(1));
C3=p3(2)*(p4(1)-p3(1))-p3(1)*(p4(2)-p3(2));
% equation 4 from point 4 and 1
A4=p1(2)-p4(2);
B4=-(p1(1)-p4(1));
C4=p4(2)*(p1(1)-p4(1))-p4(1)*(p1(2)-p4(2));

EQs(1)=1;
EQs(2)=A1;
EQs(3)=B1;
EQs(4)=C1;
EQs(5)=A2;
EQs(6)=B2;
EQs(7)=C2;
EQs(8)=A3;
EQs(9)=B3;
EQs(10)=C3;
EQs(11)=A4;
EQs(12)=B4;
EQs(13)=C4;
EQs(14)=p1(1);
EQs(15)=p2(1);
EQs(16)=p3(1);
EQs(17)=p4(1);
EQs(18)=p1(2);
EQs(19)=p2(2);
EQs(20)=p3(2);
EQs(21)=p4(2);
```

```
%************************************************************************

function objects=OBJECT_creat()
% add any objects to the space
% the retured objects is a 2D array [n,25]
% the first dimension expresses the number of object
% the second dimension is the type + the paprameters of object
% type and paremeters of objects
% 1: rectangle
%    A1,B1,C1,A2,B2,C2,A3,B3,C3,A4,B4,C4,x1,x2,x3,x4,y1,y2,y3,y4
% 2: circle
%    r,x1,y1
% 3: triangle
%    A1,B1,C1,A2,B2,C2,A3,B3,C3,x1,x2,x3,y1,y2,y3
% 4: ellipse
%    a,b,x1,y1

global fig_num

objects=zeros(15,25);

figure(fig_num)
fig_num = fig_num + 1;
hold on
% object #1, 2m by 3m
% vertexes:
p1=[17,40];
p2=[19,40];
p3=[19,43];
p4=[17,43];
plot([p1(1),p2(1),p3(1),p4(1),p1(1)],[p1(2),p2(2),p3(2),p4(2),p1(2)],'-')
objects(1,:)=points_EQ_REC(p1,p2,p3,p4);

% object #2, 2m by 3m
% vertexes:
p1=[34,40];
p2=[36,40];
p3=[36,43];
p4=[34,43];
plot([p1(1),p2(1),p3(1),p4(1),p1(1)],[p1(2),p2(2),p3(2),p4(2),p1(2)],'-')
objects(2,:)=points_EQ_REC(p1,p2,p3,p4);

% object #3, 2m by 3m
% vertexes:
p1=[51,40];
p2=[53,40];
p3=[53,43];
p4=[51,43];
plot([p1(1),p2(1),p3(1),p4(1),p1(1)],[p1(2),p2(2),p3(2),p4(2),p1(2)],'-')
objects(3,:)=points_EQ_REC(p1,p2,p3,p4);

% object #4, 2m by 3m
% vertexes:
p1=[68,40];
```

```
p2=[70,40];
p3=[70,43];
p4=[68,43];
plot([p1(1),p2(1),p3(1),p4(1),p1(1)],[p1(2),p2(2),p3(2),p4(2),p1(2)],'-')
objects(4,:)=points_EQ_REC(p1,p2,p3,p4);

% object #5, 3m by 5m
% vertexes:
p1=[80,25];
p2=[83,25];
p3=[83,30];
p4=[80,30];
plot([p1(1),p2(1),p3(1),p4(1),p1(1)],[p1(2),p2(2),p3(2),p4(2),p1(2)],'-')
objects(5,:)=points_EQ_REC(p1,p2,p3,p4);

axis([0,100,0,60])
grid on
axis equal

%*************************************************************************

function [points, points_num, obs_num,
index_points]=LMS_SIMULATOR_BOARD(x,y,theta,objects,max_range,res)
% simulate the LMS scanner and find which objects should be detected from
% current position (x,y,theta)
% the scanner is located at (x,y,theta), the heading angle is theta, the maxium of range is max_range
% the resolution is res= 0.5 degree or 1 degree;
% objects are the obstacles in the space expressed by a set of equations
% the returned array is in this format [actual angle,order of scanning,range, x, y]

global MAX_NUM_OBS_ONE_SCAN
MAX_NUM_OBS_ONE_SCAN = 25;

num_total=0;
num_line=0;
num_circle=0;
num_ellipse=0;
i=1;
% count the number of total objects(rectangle,circle,ellipse and triangle
a=size(objects);
num_total=a(2)*4;

% extract the equations expressing the related objects
num_line_current=1;
num_cir_current=1;
% line equation [A,B,C,x1,y1,x2,y2], efficients and two points
equation_line=zeros(num_total,9);
% circle equation [r,x,y], radius and center
equation_circle=zeros(num_circle,3);

for i=1:a(2)
   for i1=1:4
      [A,B,C]=general_line(objects(i).SEGMENT(i1).ENDS(1,1),objects(i).SEGMENT(i1).ENDS(1,2),...
                  objects(i).SEGMENT(i1).ENDS(2,1),objects(i).SEGMENT(i1).ENDS(2,2));
      equation_line(num_line_current,1)=A;
```

```
        equation_line(num_line_current,2)=B;
        equation_line(num_line_current,3)=C;
        equation_line(num_line_current,4)=objects(i).SEGMENT(i1).ENDS(1,1);
        equation_line(num_line_current,5)=objects(i).SEGMENT(i1).ENDS(1,2);
        equation_line(num_line_current,6)=objects(i).SEGMENT(i1).ENDS(2,1);
        equation_line(num_line_current,7)=objects(i).SEGMENT(i1).ENDS(2,2);
        equation_line(num_line_current,8)=i;
        equation_line(num_line_current,9)=i1;
        num_line_current=num_line_current+1;
    end
end

points_num=0;
obs_num=0;
index_obs=zeros(1,MAX_NUM_OBS_ONE_SCAN);
index_points=zeros(MAX_NUM_OBS_ONE_SCAN,3);
points=zeros(361,7);
scan_current=zeros(361,2);
x1=x; %+offset_x; % consider pos offset
y1=y; %+offset_y;
temp = 0;
for i=1:361
    angle = theta+pi/2-(pi-pi/360*(i-1)); %+rotation; % consider rotation
    angle = angle_wrap(angle);
    x2=x1+30*cos(angle);
    y2=y1+30*sin(angle);
    A=y2-y1;
    B=-(x2-x1);
    C=y1*(x2-x1)-x1*(y2-y1);
    scan_line=[A,B,C,x1,y1,x2,y2];
    range1=inter_lines_board(scan_line,equation_line);
    if num_circle>0
        range2=inter_circle(scan_line,equation_circle);
    else
        range2=[80,0,0,0,0];
    end
    range=closest_range(range1,range2);
    scan_current(i,1)=i*pi/360;
    scan_current(i,2)=range(1);
    if range(1)<=25
        points_num=points_num+1;
        R=range(1);
        x2=range(2);
        y2=range(3);
    end
        x2=range(2);
        y2=range(3);
    points(i,:)=[angle*180/pi,361-i,R,x2,y2,range(4),range(5)];

    if (i==1)
        if (points(i,6)~=0&points(i,3)<=25)
            temp=points(i,6);
            obs_num=1;
            index_obs(obs_num)=temp;
        end
```

```
   else
      if (points(i,6)~=0 & points(i,3)<=25)
         if temp == 0
            temp = points(i,6);
            obs_num = 1;
            index_obs(obs_num) = temp;
         else
            if temp ~= points(i,6)
               temp1=0;
               for i1 = 1:obs_num
                  if index_obs(i1)==points(i,6)
                     temp1=1;
                  end
               end
               if temp1 == 0
                  obs_num=obs_num+1;
                  index_obs(obs_num)=points(i,6);
                  temp = points(i,6);
               end
            end
         end
      end
   end
end

i1=0;
for i = 1:361
   if points(i,3)>25
      points(i,3)=80;
      points(i,4:7)=0;
   else
      if i1==0
         index_points(i1+1,1)=i;
         index_points(i1+1,2)=i;
         index_points(i1+1,3)=points(i,6);
         i1=i1+1;
      else
         if (points(i-1,6) == points(i,6))
            index_points(i1,2)=index_points(i1,2)+1;
         else
            index_points(i1+1,1)=i;
            index_points(i1+1,2)=i;
            index_points(i1+1,3)=points(i,6);
            i1=i1+1;
         end
      end
   end
end

%*************************************************************************
 function range=inter_lines_board(scan,lines)
% find the intersection of one line and a set of lines
% return the range,x,y,objects_number,segments_number

t=size(lines);
```

```
num=t(1);
r=zeros(num,5);

for i=1:num
   if (scan(1)/lines(i,1))==(scan(2)/lines(i,2)) % parallel
      r(i,1)=80;
   else % solution of two equations
      x(i)=(scan(2)*lines(i,3)-lines(i,2)*scan(3))/(scan(1)*lines(i,2)-lines(i,1)*scan(2));
      y(i)=(scan(1)*lines(i,3)-lines(i,1)*scan(3))/(lines(i,1)*scan(2)-scan(1)*lines(i,2));
      x_l=min(lines(i,4),lines(i,6)); %range of line
      x_r=max(lines(i,4),lines(i,6));
      y_b=min(lines(i,5),lines(i,7));
      y_t=max(lines(i,5),lines(i,7));
      s_x_l=min(scan(4),scan(6)); % range of scanning line
      s_x_r=max(scan(4),scan(6));
      s_y_b=min(scan(5),scan(7));
      s_y_t=max(scan(5),scan(7));
      x_min=abs(x(i)-x_l);
      x_max=abs(x(i)-x_r);
      y_min=abs(y(i)-y_b);
      y_max=abs(y(i)-y_t);
      if (x(i)>x_l&x(i)<x_r)|x_min<0.00001|x_max<0.00001
         if (y(i)>y_b&y(i)<y_t)|y_min<0.00001|y_max<0.00001
            if (x(i)>s_x_l&x(i)<s_x_r)|x(i)==s_x_l|x(i)==s_x_r
               if (y(i)>s_y_b&y(i)<s_y_t)|y(i)==s_y_b|y(i)==s_y_t
                  r(i,1)=distance_line(scan(4),scan(5),x(i),y(i));
                  r(i,2)=x(i);
                  r(i,3)=y(i);
                  r(i,4)=lines(i,8);
                  r(i,5)=lines(i,9);
               end
            end
         end
      end
   end
   if r(i)==0
      r(i)=80;
   end
end

range(1)=min(r(:,1));
for i=1:num
   if r(i,1)==range(1)
      range(2)=r(i,2);
      range(3)=r(i,3);
      range(4)=r(i,4);
      range(5)=r(i,5);
   end
end

%*************************************************************************
 function OBJECTS=GLOBAL_OBJECTS(obj)
% LIMIT: only use for rectangles
% format all the objects in objects to structure array: OBJECTS
% OBJECTS has structure as:
```

```
% OBJECTS {
%       ID;
%       NUM_SEGS   NUMBER of segments; %XXXXXXXXXXXXXXXXX
%       TYPE;  1:rectangle
%       CENTER; [x,y] coordinate of virtual center of the object
%       VERTEX; [x1,y1;x2,y2;x3,y3;...] coordinate of vertex points,
%                               circle: ???
%       DIM; [length, width] dimension of the object
%       POINTS; [x1,y1;x2,y2;...] all points belonged to this object,
%            deleted after found the points in segments
%       SEGMENTS; another structure array, describe the segments in this
%                                                   object
%            }
% line segment structure
% SEGMENTS {
%        NUM(ID); [ID# in object, object ID#]
%        CENTER; [x,y]
%        ENDS; [x1,y1;x2,y2]
%        SLOPE; angle in rad
%        POINTS; [x1,y1;x2,y2;...] all points belonged to this line
%                                              segment
%        LENGTH; length of this line segment
%                    }

global num_total

num_total=0;
i=1;

while obj(i,1)
   num_total=num_total+1;
   OBJECTS(i).ID = num_total;
   OBJECTS(i).TYPE = obj(i,1);
   OBJECTS(i).NUM_SEGS = 4;
   OBJECTS(i).VERTEX = [obj(i,14),obj(i,18);obj(i,15),obj(i,19);
                obj(i,16),obj(i,20);obj(i,17),obj(i,21)];
   OBJECTS(i).CENTER = [( OBJECTS(i).VERTEX(1,1)+ OBJECTS(i).VERTEX(2,1)...
                 +OBJECTS(i).VERTEX(3,1)+ OBJECTS(i).VERTEX(4,1) )/4,
                ( OBJECTS(i).VERTEX(1,2)+ OBJECTS(i).VERTEX(2,2)...
                 +OBJECTS(i).VERTEX(3,2)+ OBJECTS(i).VERTEX(4,2) )/4];

   for j=0:3
      OBJECTS(i).SEGMENT(j+1).ID = [j+1,i];
      k=mod(j+1,4)+1;
      OBJECTS(i).SEGMENT(j+1).ENDS = [OBJECTS(i).VERTEX(j+1,1),OBJECTS(i).VERTEX(j+1,2);
                          OBJECTS(i).VERTEX(k,1),OBJECTS(i).VERTEX(k,2)];
      OBJECTS(i).SEGMENT(j+1).CENTER =
[(OBJECTS(i).SEGMENT(j+1).ENDS(1,1)+OBJECTS(i).SEGMENT(j+1).ENDS(1,2))/2,

(OBJECTS(i).SEGMENT(j+1).ENDS(1,2)+OBJECTS(i).SEGMENT(j+1).ENDS(2,2))/2];
      OBJECTS(i).SEGMENT(j+1).SLOPE = atan2((OBJECTS(i).SEGMENT(j+1).ENDS(2,2)-
OBJECTS(i).SEGMENT(j+1).ENDS(1,2)), ...
                                (OBJECTS(i).SEGMENT(j+1).ENDS(2,1)-
OBJECTS(i).SEGMENT(j+1).ENDS(1,1)));
```

```
      OBJECTS(i).SEGMENT(j+1).LENGTH = sqrt((OBJECTS(i).SEGMENT(j+1).ENDS(2,2)-
OBJECTS(i).SEGMENT(j+1).ENDS(1,2))^2 ...
                                 +(OBJECTS(i).SEGMENT(j+1).ENDS(2,1)-
OBJECTS(i).SEGMENT(j+1).ENDS(1,1))^2);
   end
   i=i+1;
end

%************************************************************************
% Find the shortest distance from a point to a finite line
% if two lines are normal, k1*k2=-1
   function D=point_line_shortest(x, y, x1, y1, x2, y2)

   [A1,B1,C1]=general_line ( x1, y1, x2, y2);

   if (A1==0)
      A2=1;
      B2 = 0;
      C2 = -x;
      if (x < min(x1,x2) | x> max(x1,x2))

         D1 = sqrt((x - x1)*(x - x1)+(y - y1)*(y - y1));
         D2 = sqrt((x - x2)*(x - x2)+(y - y2)*(y - y2));
         D  = min(D1,D2);
      else
         D = abs(y - y1);
      end
   else
      if (B1 == 0)
         A2 = 0;
         B2 = 1;
         C2 = -y;
         if (y < min(y1,y2) | y> max(y1,y2))

            D1 = sqrt((x - x1)*(x - x1)+(y - y1)*(y - y1));
            D2 = sqrt((x - x2)*(x - x2)+(y - y2)*(y - y2));
            D  = min(D1,D2);
         else
            D = abs(x - x1);
         end
      else
         K1 = -A1/B1;
         K2 = -1/K1;
         A2 = K2;
         B2 = -1;
         C2 = y - K2*x;
         X = (B1 * C2 - B2 * C1) / (A1 * B2 - A2 * B1);
         Y = (C1 * A2 - C2 * A1) / (A1 * B2 - A2 * B1);
         if ((X >= min(x1,x2)) & (X <= max(x1,x2)) & (Y >= min(y1,y2)) & (Y <= max(y1,y2)))
            D = sqrt((X - x)*(X -x) + (Y - y)*(Y - y));
         else
            D1 = sqrt((x1 - x)*(x1 -x) + (y1 - y)*(y1 - y));
            D2 = sqrt((x2 - x)*(x2 -x) + (y2 - y)*(y2 - y));
            D  = min(D1,D2);
         end
```

```
      end
   end

%************************************************************************

function INVI = orientation(S,E,V)
% find the visibility of segment by the orientation
% S is the start point of segment, E is the end point of segment
% V is the sensor location
% return
%     1: clockwise, visible from V
%    -1: counterclockwise, invisible from V
%     2: collinear, one point visible

global ERR
ERR = 1e-6;
INVI = 0;

D = ((E(1) - S(1))*(V(2) - S(2))) - ((V(1) - S(1))*(E(2) - S(2)));

if D >  ERR
   INVI = -1;
   return;
else
   if D < -ERR
      INVI =  1;
      return;
   else
      INVI = 2;
      return;
   end
end

%************************************************************************
function [deltaX, deltaY, deltaTHETA, D]=MATCHING(seg, seg_num, points, points_num, fig)
% use Fox' algo to find the matching segments of a set of points
% return the difference between the set of points and the segments
% to apply the returned deltaX, deltaY, and deltaTHETA, translate at first,
% then rotate
% D is the average distance from each point to matched segment

global TRANS_MAX
global ROT_MAX

% calculate the parameters of segment line
% Format: Ax + By + C = 0
% seg(5,:) to seg(7,:) - A,B,C
% seg(8:9,:) are used in fox matching algo
for i = 1:seg_num
   [seg(i,5), seg(i,6), seg(i,7)]=general_line( seg(i,1), seg(i,2), seg(i,3), seg(i,4));
   seg(i,8) = seg(i,5)/sqrt(seg(i,5)*seg(i,5)+seg(i,6)*seg(i,6));
   seg(i,9) = seg(i,6)/sqrt(seg(i,5)*seg(i,5)+seg(i,6)*seg(i,6));
   seg(i,10) = -seg(i,7)/sqrt(seg(i,5)*seg(i,5)+seg(i,6)*seg(i,6));
end
```

```
points0=points;
X=zeros(361,3);
X_trans=zeros(3,361);
Xtrans_X=zeros(3,3);
inv_Xtrans_X=zeros(3,3);
Y=zeros(361,1);
b=zeros(3,1);
b1=[10;10;pi];
B=zeros(3,1);
N=1;
SEGavgX=zeros(seg_num);
SEGavgY=zeros(seg_num);
SEGavg_X=0;
SEGavg_Y=0;
b=zeros(3,100);

while max(abs(b1))>=0.005&N<=3
   Xavg = 0;
   Yavg = 0;
   m1=zeros(1,2);
   m2=[0,-1;1,0];
   m3=zeros(2,1);
   m1m2=zeros(1,2);
   m1m2m3=zeros(1,1);
   P_S=zeros(points_num,2);
   for i = 1:points_num
      Xavg = Xavg + points(i,1);
      Yavg = Yavg + points(i,2);
   end
   Xavg = Xavg/points_num;
   Yavg = Yavg/points_num;
   if N==1
      for i=1:seg_num
         SEGavgX(i)=(seg(i,1)+seg(i,3))/2;
         SEGavgY(i)=(seg(i,2)+seg(i,4))/2;
      end
      for i=1:seg_num
         SEGavg_X=SEGavg_X+SEGavgX(i);
         SEGavg_Y=SEGavg_Y+SEGavgY(i);
      end
      SEGavg_X=SEGavg_X/seg_num;
      SEGavg_Y=SEGavg_Y/seg_num;
      b0=[SEGavg_X-Xavg,SEGavg_Y-Yavg];
      for i=1:points_num
         points(i,1)=points(i,1)+b0(1);
         points(i,2)=points(i,2)+b0(2);
      end
      points2=points;
      Xavg=0;
      Yavg=0;
      for i = 1:points_num
         Xavg = Xavg + points(i,1);
         Yavg = Yavg + points(i,2);
      end
      Xavg = Xavg/points_num;
```

```
    Yavg = Yavg/points_num;
    Xavg0=Xavg;
    Yavg0=Yavg;
end
for i =1:points_num
    d0 = 100;
    for j =1:seg_num
        d = point_line_shortest(points(i,1), points(i,2), seg(j,1), seg(j,2), seg(j,3), seg(j,4));
        if (d < d0)
            d0 = d;
            P_S(i,1) = j;
            P_S(i,2) = d0;
        end
    end
    % in case, the algo has been dis-convergent
    % return as fail
    if P_S(i,1) == 0
        D = -100;
        deltaX = 0;
        deltaY = 0;
        deltaTHETA = 0;
        return
    end
end

for i =1:points_num

    X(i,1) = seg(P_S(i,1),8);
    X(i,2) = seg(P_S(i,1),9);
    m1(1,1) = X(i,1);
    m1(1,2) = X(i,2);
    m3(1,1) = points(i,1) - Xavg;
    m3(2,1) = points(i,2) - Yavg;
    m1m2=m1*m2;
    m1m2m3=m1m2*m3;
    X(i,3) = m1m2m3(1,1);

    m1(1,1) = X(i,1);
    m1(1,2) = X(i,2);
    m3(1,1) = points(i,1);
    m3(2,1) = points(i,2);
    m1m2m3=m1*m3;
    Y(i) = seg(P_S(i,1),10) - m1m2m3(1,1);
end
X_trans=X';
Xtrans_X=X_trans*X;

inv_Xtrans_X=inv(Xtrans_X);

inv_Xtrans_X_Xtrans=inv_Xtrans_X*X_trans;
b(:,N)=inv_Xtrans_X_Xtrans*Y;
b1=b(:,N);
B=B+b1;
N=N+1;
```

```
    for i=1:points_num
        pp1=points(i,1);
        pp2=points(i,2);
        points(i,1)=b1(1)+(pp1-Xavg)*cos(b1(3))-(pp2-Yavg)*sin(b1(3))+Xavg;
        points(i,2)=b1(2)+(pp1-Xavg)*sin(b1(3))+(pp2-Yavg)*cos(b1(3))+Yavg;
    end
end

D = 0;
for i = 1:points_num
    D = D + P_S(i,2);
end
D = D/points_num;

if D <= 0.1
    A = zeros(seg_num, 4);
    % 1: the number of points in this segment
    % 2: the average distance for this segment
    % 3: index of start point
    % 4: index of end point
    A(1,3) = 1;
    a = P_S(1,1);
    b = 0;
    i1 = 0;
    i2 = 1;
    for i=1:points_num
        if (P_S(i,1) == a )%points are same as the start
            i1 = i1 + 1;
            b = b + P_S(i,2);
        else % new segment matching start
            A(i2, 1) = i1;
            A(i2, 2) = b/i1;
            A(i2, 4) = A(i2, 3) + i1 -1;
            a = P_S(i,1);
            b = P_S(i,2);
            i1 = 1;
            i2 = i2 + 1;
            A(i2, 3) = A(i2-1, 4) + 1;
        end
        if i == points_num % the last point case
            A(i2, 1) = i1;
            A(i2, 2) = b/i1;
            A(i2, 4) = A(i2, 3) + i1 - 1;
            match_seg = i2; % the total number of matched segments
        end
    end

    B = 10;
    b = 0;
    for i = 1:match_seg % find the segment with minimum distance
        if (A(i,2) <= B && A(i,1) >=3)
            b = i;
            B = A(i,2);
        end
    end
```

```
    if b == 0 % D is too big to be a good matching
        D = -100;
        deltaX = 0;
        deltaY = 0;
        deltaTHETA = 0;
        return
    else % find the rotation angle at first
        th0=atan2(points0(A(b,4),2)-points0(A(b,3),2),points0(A(b,4),1)-points0(A(b,3),1));
        th1=atan2(points(A(b,4),2)-points(A(b,3),2),points(A(b,4),1)-points(A(b,3),1));
        th=th0-th1;
        th = angle_wrap(th);
    end

    C = 10;
    for i = A(b,3):A(b,4) % find one point in this segment which has the minimum distance to the segment
        if P_S(i,2) < C
            c = i;
            C = P_S(i,2);
        end
    end
    % apply the rotation angle
    t1 = points0(c,1)*cos(th) + points0(c,2)*sin(th);
    t2 = -points0(c,1)*sin(th) + points0(c,2)*cos(th);
    % find the translation
    deltaX = t1 - points(c,1);
    deltaY = t2 - points(c,2);
    deltaTHETA = th;
    if abs(deltaX) > TRANS_MAX | abs(deltaY) > TRANS_MAX |abs(deltaTHETA) > ROT_MAX
        D = -100;
        deltaX = 0;
        deltaY = 0;
        deltaTHETA = 0;
        return
    end
else
    D = -100;
    deltaX = 0;
    deltaY = 0;
    deltaTHETA = 0;
    return;
end


%*************************************************************************
function flag = BREAK_DETECTION(phi0, r0, x0,y0,...
                        phi1, x1, y1,...
                        delta, sigma)

delta_phi = (phi1 - phi0)*pi/180;
threshold = r0*(sin(delta_phi)/sin(delta - delta_phi)) + 3*sigma;

D = sqrt((x0 - x1)*(x0 - x1) + (y0 - y1)*(y0 -y1));

if (D < threshold)
    flag = 0;
```

```
else
   flag = 1;
end

%*************************************************************************

%  iterative algo to find how many segments in the object presented by points,
%    results are the turning points' index in all points and the count of segment,
%    put in lms_seg_index and seg_count
%    NOTE: always set global int HANOI to ZERO before calling this function
% HANOI = 0,
% HANOI_INDEX[MAX_VEC_ONE_OB][2],lms_seg_index[MAX_VEC_ONE_OB][2]
 function ITERATIVE_SEG_SPLIT(P, startP, endP)

global   HANOI
global   HANOI_INDEX
global   lms_seg_index
global   seg_count
global   segment_threshold

%int i, j = 0, m;
p = zeros(361, 7);

j = 1;
for i = startP:endP
   p(j,:) = P(i,:);
   j = j + 1;
end

m = turning_point(p, endP - startP + 1, segment_threshold);
if(m ~= 0)
%  have turning point
%    process the first portion and push the second portion into stack
   HANOI_INDEX(HANOI + 1,:) = [startP + m - 1, endP];
   HANOI = HANOI + 1;
   iterative_seg_split(P, startP, startP + m - 1);
else
%  no turning, take the newest stored portion from stack*/
   HANOI = HANOI - 1;
   if (seg_count == 0)
      lms_seg_index(seg_count + 1, 1) = startP;
   else
      lms_seg_index(seg_count + 1, 1) = startP + 1;
   end

   if (HANOI < 0)
      lms_seg_index(seg_count + 1, 2) = endP;
   else
      lms_seg_index(seg_count + 1, 2) = endP - 1;
   end

   if ( (lms_seg_index(seg_count + 1,2)...
               -  lms_seg_index(seg_count +1, 1)) >= 2)
      %at least 3 points required for one segment
      seg_count = seg_count + 1;
```

```
    end

    if (HANOI >= 0)
        iterative_seg_split(P, HANOI_INDEX(HANOI + 1, 1), ...
                            HANOI_INDEX(HANOI + 1, 2));
    end

end

while(HANOI > 0)
    j = 1;
    for i = startP:endP
        p(j,:) = P(i,:);
        j = j + 1;
    end

    m = turning_point(p, endP - startP + 1, 1);
    if(m ~= 0)
    %  have turning point
    %    process the first portion and push the second portion into stack
        HANOI_INDEX(HANOI + 1,:) = [startP + m - 1, endP];
        HANOI = HANOI + 1;
        iterative_seg_split(P, startP, startP + m - 1);
    else
    %  no turning, take the newest stored portion from stack*/
        HANOI = HANOI - 1;
        if (seg_count == 0)
            lms_seg_index(seg_count + 1, 1) = startP;
        else
            lms_seg_index(seg_count + 1, 1) = startP + 1;
        end

        if (HANOI < 0)
            lms_seg_index(seg_count + 1, 2) = endP;
        else
            lms_seg_index(seg_count + 1, 2) = endP - 1;
        end

        if ( (lms_seg_index(seg_count + 1,2)...
                    -  lms_seg_index(seg_count +1, 1)) >= 2)
            %at least 3 points required for one segment
            seg_count = seg_count + 1;
        end

        if (HANOI >= 0)
            iterative_seg_split(P, HANOI_INDEX(HANOI + 1, 1), ...
                                HANOI_INDEX(HANOI + 1, 2));
        end

    end
end

%*********************************************************************

function [x,y,r] = intersection_segments(x1, y1, x2, y2, x3, y3, x4, y4)
```

```
eps = 0.0000001;
r = -1;
x = 0;
y = 0;

[A1,B1,C1] = line_equation_two_points ( x1, y1, x2, y2);
[A2,B2,C2] = line_equation_two_points ( x3, y3, x4, y4);

D  = ( A1 * B2  - B1 * A2);
Dx = (-C1 * B2  + B1 * C2);
Dy = ( A1 * -C2 + C1 * A2);

if (D == 0)
   a = [A1,A2];
   b = [B1,B2];
   c = [C1,C2];
   if (abs(a(1)*b(2) - b(1)*a(2)) < eps...
      && abs(b(1)*c(2) - c(1)*b(2)) < eps)
      D1 = sqrt((x3 - x1)*(x3 - x1) + (y3 - y1)*(y3 - y1));
      D2 = sqrt((x4 - x1)*(x4 - x1) + (y4 - y1)*(y4 - y1));
      if(D1 < D2)
         x = x3;
         y = y3;
      else
         x = x4;
         y = y4;
      end
      r = -1;
   else
      r = -2;
   end
   return;
else
   x = Dx / D;
   y = Dy / D;
end

if (((x <= max(x1,x2) && x >= min(x1,x2))...
         || (abs(max(x1,x2) - x) <= eps)...
         || (abs(max(x1,x2) - x) <= eps))...
         && ((x <= max(x3,x4) && x >= min(x3,x4))...
         || (abs(max(x3,x4) - x) <= eps)...
         || (abs(max(x3,x4) - x) <= eps))...
         && ((y <= max(y1,y2) && y >= min(y1,y2))...
         || (abs(max(y1,y2) - y) <= eps) ...
         || (abs(max(y1,y2) - y) <= eps))...
         && ((y <= max(y3,y4) && y >= min(y3,y4))...
         || (abs(max(y3,y4) - y) <= eps)...
         || (abs(max(y3,y4) - y) <= eps)))
   r = 1;
else
   r = 0;
end
```

```
return;

%*************************************************************************
 % line fitting from points using least square method
function [startX, startY, endX, endY, r] = LEAST_SQUARES_LINE_FITTING(points, num_points)

sumX = 0;
sumY = 0;
sumXY = 0;
sumXsquare = 0;
sumYsquare = 0;
for i = 1:num_points
   sumX = sumX + points(i,4);
   sumY = sumY + points(i,5);
   sumXY = sumXY + points(i,4)*points(i,5);
   sumXsquare = sumXsquare + points(i,4)*points(i,4);
   sumYsquare = sumYsquare + points(i,5)*points(i,5);
end

D = sumXsquare*num_points - sumX*sumX;
m = (sumXY*num_points - sumX*sumY)/D;
b = (sumXsquare*sumY - sumX*sumXY)/D;
r = 1;

tx1 = points(1,4);
ty1 = m*tx1 + b;
tx2 = points(num_points,4);
ty2 = m*tx2 + b;

[startX, startY] = segs_inter_perpendicular(points(1,4),points(1,5),tx1,ty1,tx2,ty2);
[endX, endY] = segs_inter_perpendicular(points(num_points,4),points(num_points,5),tx1,ty1,tx2,ty2);

%*************************************************************************
% judge if a point inside of a polygon
function inside = point_in_polygon(rec, x,y)

inside = 0;
for i = 0:3
   V1 = rec(i+1,:);
   V2 = rec(mod(i+1,4)+1,:);
   if(x == V1(1) && y == V1(2))
      inside = 1;
      return
   end
   if(( (((V1(2)) <= y)&& (y < V2(2)) )...
      || (V2(2) <= y) && (y < V1(2)))...
      && (x < ((V2(1) - V1(1))*(y - V1(2))/(V2(2) - V1(2)) + V1(1))) )
      inside = ~inside;
   end
end

return

%*************************************************************************
 function  [x,y,theta] = robot_kinematics_model_ideal(v,omega,x0,y0,theta0,delta_t)
```

```
if (v ~= 0)
   x = x0 + v*delta_t*cos(theta0);
   y = y0 + v*delta_t*sin(theta0);
   theta = angle_wrap_2PI(theta0);
end
if(omega ~=0)
   x = x0;
   y = y0;
   theta = theta0 + omega*delta_t;
   theta = angle_wrap_2PI(theta);
end

%**************************************************************************

function [delta_x, delta_y, delta_theta] = seg_to_seg_matching(lms_map_seg, num_lms_seg, map_seg)

D_angle = 0;
for i = 1:num_lms_seg
   n = lms_map_seg(i,10);
   lms_map_seg(i,13) = lms_map_seg(i,7) - map_seg(n,7);
   D_angle = D_angle + lms_map_seg(i,13);
end

D_angle = D_angle/num_lms_seg;

D_x = 0;
D_y = 0;
for i = 1:num_lms_seg
   D_x = D_x + (-map_seg(lms_map_seg(i,10),8) + lms_map_seg(i,8));
   D_y = D_y + (-map_seg(lms_map_seg(i,10),9) + lms_map_seg(i,9));

end

D_x = D_x/num_lms_seg;
D_y = D_y/num_lms_seg;

delta_x = D_x;
delta_y = D_y;
delta_theta = D_angle;

%**************************************************************************

function [map_seg, num_map_seg, lms_seg, num_lms_seg,...
        lms_seg_map, num_lms_seg_map, lms_seg_unknown, num_lms_seg_unknown]...
                = SEGMENTS_PROCESSING(map_obs_seg, lms_obs_seg)

global MATCHING_BUFFER
global MATCHING_ROT_MAX

map_seg = zeros(100,15);
num_map_seg = 0;
lms_seg = zeros(100,15);
num_lms_seg = 0;
lms_seg_map = zeros(100,15);
```

```
num_lms_seg_map = 0;
lms_seg_unknown = zeros(100,15);
num_lms_seg_unknown = 0;

n = size(map_obs_seg);
k = 1;
for i = 1:n(2)
   map_seg(k,:) = zeros(1,15);
   for j = 1:map_obs_seg(i).seg_count
      map_seg(k,1) = map_obs_seg(i).obs_id;
      map_seg(k,2) = j;
      map_seg(k,3) = map_obs_seg(i).segment(j).startX;
      map_seg(k,4) = map_obs_seg(i).segment(j).startY;
      map_seg(k,5) = map_obs_seg(i).segment(j).endX;
      map_seg(k,6) = map_obs_seg(i).segment(j).endY;
      map_seg(k,7) = map_obs_seg(i).segment(j).slope;
      map_seg(k,8) = map_obs_seg(i).segment(j).centerX;
      map_seg(k,9) = map_obs_seg(i).segment(j).centerY;
      k = k + 1;
   end
end
num_map_seg = k - 1;

n = size(lms_obs_seg);
k = 1;
for i = 1:n(2)
   lms_seg(k,:) = zeros(1,15);
   for j = 1:lms_obs_seg(i).seg_count
      lms_seg(k,1) = lms_obs_seg(i).obs_id;
      lms_seg(k,2) = j;
      lms_seg(k,3) = lms_obs_seg(i).segment(j).startX;
      lms_seg(k,4) = lms_obs_seg(i).segment(j).startY;
      lms_seg(k,5) = lms_obs_seg(i).segment(j).endX;
      lms_seg(k,6) = lms_obs_seg(i).segment(j).endY;
      lms_seg(k,7) = lms_obs_seg(i).segment(j).slope;
      lms_seg(k,8) = lms_obs_seg(i).segment(j).centerX;
      lms_seg(k,9) = lms_obs_seg(i).segment(j).centerY;
      k = k + 1;
   end
end
num_lms_seg = k - 1;

for i = 1: num_map_seg
   rec = zeros(5,2);
   rec = segment_to_rectangle(map_seg(i,:), MATCHING_BUFFER);
   plot([map_seg(i,3),map_seg(i,5)],[map_seg(i,4),map_seg(i,6)],'--','LineWidth',2)
   hold on
   grid on
   plot([rec(1,1),rec(2,1),rec(3,1),rec(4,1),rec(1,1)],...
       [rec(1,2),rec(2,2),rec(3,2),rec(4,2),rec(1,2)],'--')
   for j = 1:num_lms_seg
      if(point_in_polygon(rec, lms_seg(j,3), lms_seg(j,4))...
       &&point_in_polygon(rec, lms_seg(j,5), lms_seg(j,6))...
       &&(abs(rec(5,1) - lms_seg(j,7)) <= MATCHING_ROT_MAX))
       if(lms_seg(j,10) == 0)
```

```
            lms_seg(j,10) = i;
            break
        end
       end
    end
end

for i = 1:num_lms_seg
    if(lms_seg(i,10) ~= 0)
        num_lms_seg_map = num_lms_seg_map + 1;
        lms_seg_map(num_lms_seg_map,:) = lms_seg(i,:);
    else
        num_lms_seg_unknown = num_lms_seg_unknown + 1;
        lms_seg_unknown(num_lms_seg_unknown,:) = lms_seg(i,:);
    end
end

for i = 1:num_lms_seg_map
    plot([lms_seg_map(i,3),lms_seg_map(i,5)],[lms_seg_map(i,4),lms_seg_map(i,6)],'-r','LineWidth',2)
end

%**************************************************************************

function [x1,y1,theta1,v,omega,tour_id_new] = robot_trajectory_ideal(tour_seg, seg_count, tour_id,
x,y,theta,v, delta_t)

r = v*delta_t;

angle_eps = 0.0436;%in rad, 2.5 degree
G = 0.3; %tuning by performance
tour_id_new = tour_id;

PHI = tour_seg(tour_id,6);

% if robot is close to next tour segment
d1 = sqrt((x - tour_seg(tour_id,4))*(x - tour_seg(tour_id,4)) + (y - tour_seg(tour_id,5))*(y -
tour_seg(tour_id,5)));
if (d1 <= r)
    tour_id_new = mod(tour_id, seg_count) + 1;
end

d2 = sqrt((x - tour_seg(tour_id,2))*(x - tour_seg(tour_id,2)) + (y - tour_seg(tour_id,3))*(y -
tour_seg(tour_id,3)));

if (tour_id_new == 1 && tour_id == 4)
    tour_id = 4;
end

if(tour_id_new ~= tour_id)
    delta_PHI = theta - tour_seg(tour_id_new,6);
    delta_PHI = angle_wrap(delta_PHI);
else
%if robot is departured too much from the current tour
    if (d2 <= r)
        delta_PHI = theta - PHI;
```

```
        delta_PHI = angle_wrap(delta_PHI);
    else
        PHI_1 = atan2(tour_seg(tour_id,5) - y, tour_seg(tour_id,4) - x);
        PHI_1 = angle_wrap_2PI(PHI_1);
        delta_PHI_1 = PHI - PHI_1;
        delta_PHI = theta - PHI;
        delta_PHI = delta_PHI + delta_PHI_1;
        delta_PHI = angle_wrap(delta_PHI);
    end
end

if (tour_id_new == tour_id && abs(delta_PHI) <= angle_eps )
    omega = 0;
    [x1,y1,theta1] = robot_kinematics_model_ideal(v,0,x,y,theta,delta_t);
else
    v = 0;
    omega = -G*delta_PHI*5;
    [x1,y1,theta1] = robot_kinematics_model_ideal(0,omega,x,y,theta,delta_t);
end

return;
```