Thesis for the Degree of Master of Engineering

Autonomous Navigation of Mobile Robot in Dynamic Environments Based on Laser Range Finder Sensor

by

Nguyen Van Phuc

Graduate School

Department of Mechatronics Engineering

Pukyong National University

February 2013

AUTONOMOUS NAVIGATION OF MOBILE ROBOT IN DYNAMIC ENVIRONMENTS BASED ON LASER RANGE FINDER SENSOR

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF MECHATRONICS
ENGINEERING AND THE COMMITTEE ON GRADUATE STUDIES
OF PUKYONG NATIONAL UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF ENGINEERING

Nguyen Van Phuc

February 2013

Autonomous Navigation of Mobile Robot in Dynamic Environments

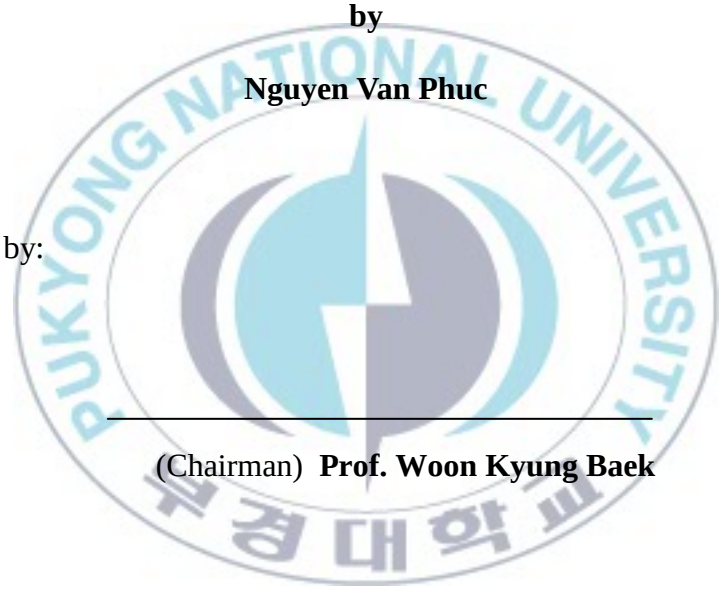
Based on Laser Range Finder Sensor

A Thesis

by

Nguyen Van Phuc

Approved by:



(Chairman) **Prof. Woon Kyung Baek**

(Member) **Prof. Young Seok Jung**

(Member) **Prof. Doo Sung Ahn**

19 December 2012

ACKNOWLEDGEMENTS

First of all I would like to express my appreciation to Professor Doo Sung Ahn for supporting me over all these years. It was a pleasure to work with him as my advisor. I am grateful for his suggestions and guidance. Actually, I cannot think of a more competent, more enthusiastic and also a more critical advisor. I have learned a lot during the last few years.

I would like to thank all the current members and alumni of my Laboratory, and Korean friends for their generous assistances. My special thanks go to Vietnamese friends in Korea for coming up with sharing in life at Pukyong National University, for helping to achieve invaluable documents. Everyone has been open-minded to discussions which contributed to this thesis. This thesis could not have been done without the help and support of numerous people. Especially, I would like to thank all of friends at Yongdang Campus for spending a lot of time helping me to solve soft- and hard ware problems with Pioneer 3DX robot.

I am grateful to Pukyong National University for financial support of my research, for creating good conditions for my study.

Finally, I have never forgotten to thank my family: my parents, my older brother and sister, my sister-in-law, my niece and nephews who always encourage me during the time I stayed far away from home.

Pukyong National University, Busan, Republic of Korea

Nguyen Van Phuc

Autonomous Navigation of Mobile Robot in Dynamic Environments

Based on Laser Range Finder Sensor

Nguyen Van Phuc

Graduate School

Department of Mechatronics Engineering

Pukyong National University

Abstract

Autonomous navigation of a mobile robot in dynamic environments is a fundamental requirement for effective autonomous navigation and widely used in large fields of industrial application. The past few years have been tremendous growth in the research areas of autonomous navigation of a mobile robotics. Navigation encompasses the ability of the robot to act based on its knowledge and sensor values so that it could reach its goal position as efficiently and reliably as possible. Navigation involves mapping, localization and predict motion planning while colliding avoidance for mobile robots. The mapping is the process whereby a robot can extract relevant information from its environment allowing it to remember it. Localization is the problem of estimating a robot's pose relative to a map of its environment. Localization and mapping is based on data collected from a robot using a dense range scanner to generate a bi dimensional map representation of the surrounding environment. This externally sensed range data is correlated to estimate the robot's position and build a map. Path

planning for a mobile robot is to find a collision free route, through the robot's environment with obstacles, from a specified start location to a desired goal destination while satisfying certain optimization criteria.

This thesis focuses on the problem of enabling mobile robots to build world models of their environment and to employ as a reference to self-localization and path planning. The robot usually needs a representation of the environment and the capacity to interpret that representation to be able to plan a path towards some target location and to move safely in an environment where there may be variations in the position of the robots. The system builds bi dimensional maps of the environment that surrounds the robot, through data collected from the laser range finder sensor and also the estimated position of the robot. This research is geared towards implementing Local Registration/ Global Correlation (LRGC) algorithm for reliable reconstruction of consistent global maps from scanning laser data. The localization is carried out through stored environment maps by using scan matching method. The path planning task uses a focused D* (FD) search to compute the shortest and safest path from the present robot position to any reachable point in the given robot environment map. Once the main path is planned, a local segment of the path to plan around any unmapped obstacles it sees with its range sensors is recomputed. The dynamic window method is used to compute the translational and rotational velocities necessary to follow the path as closely as possible.

Experiments using data collected from a SICK LMS-200 laser range finder illustrate the effectiveness of the algorithms and improvements over previous work. All the algorithms are implemented and verified using a Pioneer 3DX mobile robot equipped with the laser range finder. Experimental results for both the simulation and real world environment show that the method improves the accuracy of localization and mapping.

레이저 센서를 기반으로 하는 동적 환경에서의 이동로봇의 자율 주행

Nguyen Van Phuc

부경대학교 일반대학원 메카트로닉스공학과

요 약

동적 환경하에서의 이동로봇의 자율 주행은 많은 산업분야에서 이용되는 효과적인 자율주행을 위해서 근본적으로 필요한 요구사항이다. 과거 몇 년동안 이동로봇의 자율주행에 관한 많은 연구가 있어왔다. 주행은 로봇이 자신의 지식과 센서정보를 바탕으로 가능한 한 효율적이고 신뢰성있게 목표위치에 도달할 수 있는 능력을 포함한다. 주행은 매핑, 위치추정를 포함하고 충돌회피를 하면서 동작설계를 가능하여야 한다. 매핑은 환경으로부터 추출한 정보를 환경과 연관되게 하여 기억할 수 있도록 하는 과정이다. 위치추정은 로봇의 위치와 자세를 지도상에서 추정하는 문제이다. 위치추정과 매핑은 로봇이 스캐나 센서를 이용하여 표현한 주위환경의 양방향 지도를 근거로 한다. 이러한 외부 데이터들은 로봇의 위치를 추정하는데 연관을 시키고 지도를 구축한다. 이동 로봇의 경로계획은 지정된 시작위치에서 원하는 목표지점까지 주어진 최적화 기준을 만족시키면서 장매물과 충돌하지 않는 경로를 찾아내는 것이다.

본 논문은 로봇이 주위 환경의 모델을 구축하고 이것을 위치 추정과 경로 계획의 기준으로 사용하는 문제에 중점을 두고 있다. 로봇은 환경의 표현과 이것을 해석하여 원하는 위치로 안전하게 이동할 수 있는 능력을 필요로 한다. 레이저 센서로 획득한 데이터데 로봇의 추정 위치를 기반으로 하여 로봇의 주위환경에 대한 양방향 지도를 구축한다. 본 연구는 레이저 스캐닝 데이터로부터 전역 지도를 일관되고 신뢰성 있게 구축하는 지역등록/전역연관 (LRGC) 알고리듬을 채용하고 있다. 위치 추정은 스캔 매칭 방법(scan matching method)에 의해 저장된 환경 지도를 통해 이루어 진다. 경로계획은 FD 방법을 이용하여 구축된 지도에서 현 위치에서 목표위치까지의 안전한 최단경로를 찾는다. 일단 주경로가 계획되면 장애물이 있는 주위의 경로는 센서를 이용하여 재계산한다. 가능한 한 경로를 잘 추종하기 위한 전위속도와 회전속도를 도출하기 위해 동적윈도방법(dynamic window method)을 사용한다.

SICK LMS-200 레이저를 이용한 실험은 이전의 결과들에 비해 효율성과 개선된 점을 보여준다.모든 알고리즘들은 레이저 센서가 장착된 파이오니아사 3DX 모바일로봇으로 실험하였다. 시뮬레이션과 실제환경에서의 실험 결과는 제시된 방법이 위치추정과 매핑에서 정확성을 향상시켰음을 보여준다.

CONTENTS

ACKNOWLEDGEMENTS .....

ABSTRACT ..... **i**

CONTENTS ..... **v**

LIST OF FIGURES ..... **vii**

LIST OF TABLES ..... **ix**

NOMENCLATURE ..... **x**

1. INTRODUCTION **1**

2. MAP BUILDING **6**

2.1. Scan matching 9

2.1.1. Scan 10

2.1.2. Feature extraction from scanning data 11

2.1.3. Median Filter 17

2.1.4. Scan matching 18

2.2. Consistent position estimate 33

2.2.1. Definition of the estimation problem 34

2.2.2. Application for the mapping 37

2.3 Map Correlation 38

2.4 LRGC algorithm 40

3. LOCALIZATION **42**

3.1 Dead Reckoning 43

3.2 Categories of localization methods 46

3.3 Scan matching localization 47

4 PATH PLANNING AND COLLISION AVOIDANCE **49**

4.1 Path planning 49

4.1.1 Approaches for path planning 51

4.1.2 FD * path planning algorithm 54
4.2 Collision Avoidance Behavior 64
4.2.1 Modeling of Wheeled Mobile Robot (WMR) 65
4.2.2 General Motion equations 66
4.2.3 Dynamic Window Approaches 68
5 EXPERIMENTAL RESULTS 71
5.1 Simulation results 71
5.2 Experimental results 73
6 CONCLUSIONS 81
6.1 Summary 81
6.2 Research limitations 82
6.3 Direction for future research 83
APPENDIX 85
REFERENCES 92

LIST OF FIGURES

Fig. 1. Table of figures entries found. Various types of service robots. (a)Cleaning robot (b) Tour guide robot (c) Darwin-Op entertainment robot, and (d) The Mars exploration Rovers. 2

Fig. 2. Add a new pose of an unexplored environment to the map. 8

Fig. 3. A typical scan which is detected by the laser scanner 180 ° SICK LMS-200. 10

Fig. 4. Extraction lines from a scan, links raw scan, right extracted lines 15

Fig. 5. Scheme of extended Cox algorithm 25

Fig. 6. Schematic structure of the extended IDC algorithm 32

Fig. 6. Schematic of the combined scan matching method 33

Fig. 7. Data flow of the LRGC mapping algorithm 40

Fig. 8. Tricycle kinematics as used for example on the Pioneer 3DX Robot 44

Fig. 9. Artificial Potential Field Method 53

Fig. 10. Kinematic variables of the WMR 65

Fig. 11. Mapping simulation (a) Simulation environment constructed with a number of objects, (b) Start scanning, (c) Final map. 72

Fig. 12. Sketch of the large environment including the trajectories of the robot. Lines of the walls correspond to the on-line detected obstacles which are incrementally incorporated in to the grid. The smooth curvature represents the true robot trajectory. Robot was able to follow the planned optimal path well. 72

Fig. 13. Motion control. (a) Path planning without obstacles, (b) and (c) Robot traveled from the start position until reaching the goal and avoid colliding with unexpected obstacles. 73

Fig. 14. The Intelligent Control Lab: The robot start in the lower left side

and runs around the loop. The right image depicts the resulting map generated. 74
Fig. 15. Corridors in the second floor of Department of Mechatronics at Pukyong National University. (a) The corridor. (b) Detail of the corridor 75
Fig. 16. 2-D environment map of a corridor in Mechatronics Building. (a) The whole environment. (b) A detail showing different artifacts . 76
Fig. 17. Outdoor area near the Mechatronics Department building 76
Fig. 18. Map of the path in the outdoor environment. 76
Fig. 19. Map of partial Yongdang Campus reconstructed by laser reading gathered in environment. 78
Fig. 20. Pioneer 3DX robot driving around an expected object. 78
Fig. 21. Typical trajectory taken by Pioneer 3DX when reaching the target position. 79
Fig. 22. Experimental run in the campus at Pukyong National University 80
Fig. 23 Trajectory of Pioneer 3DX travelling along the path shown in Fig. 22 80

LIST OF TABLES

Table 1. Some details of the data gathered in the semi-outdoor environment
.....77

NOMENCLATURE

FD*	: Focused D*
SLAM	: Simultaneous Localization And Mapping
LRGC	: Local Registration/ Global Correlation
DWA	: Dynamic Window Approach
EKF	: Extended Kalman Filter
PF	: Particle Filters
IDC	: Iterative Dual Correspondence
CAD	: Computer Aided Design

1. INTRODUCTION

Recently, developments in the residential application of mobile robot are envisioned to fulfil various kinds of tasks. In the last few years there has been a substantial progress in the field of service robots. A variety of mobile robots that are designed to operate in environments populated by humans has already been developed. These robots, for example, have been deployed in auto space exploration, underwater exploration, hospital, office building, autonomous production in factories, and department stores. Existing robotics systems are already able to perform various services such as delivery, surgeon, cleaning, education, robot-cop or entertainment. Figure 1.1 depicts four examples of existing robotic systems. The upper left image shows a Japanese cleaning robot which is designed to clean large surfaces, for example on the roads or airports. The robot in the upper right image has been developed within the tour guide robot [45]. The lower left image shows entertainment robot [46] and the lower right image depicts one of the NASA auto space exploration robots.

(a)

(b)

(c)

(d)

Fig. 1. Table of figures entries found. Various types of service robots. (a) Cleaning robot (b) Tour guide robot (c) Darwin-Op entertainment robot, and (d) The Mars exploration Rovers.

This research addresses the development for mobile robot path planning, localization and mapping based on the data collected from the laser range finder sensor. Laser range scanners have been used for many years for map building, localization, path planning, and obstacle detection and collision avoidance. Laser scanners operate by sweeping a laser across a scene and at each angle, measuring the range and returned the nearest distance from obstacles to the robot in range of scanning. In this research we focused on the range returned from the laser, since the range laser ranging demands fast and provide direct information useful for mapping. I demonstrate how a laser range alone can be used to detect and track objects in the environment. At long ranges and grazing angles, vertical objects reflect significantly more laser energy than the horizontal road. The object detection system uses a high performance laser scanner which provides fast single-line laser scans. Data analysis on the returned angle and optimal distance signal is used to select

objects candidates. After candidates are matched and merged with candidates from previous scans, the range to each object is estimated by a novel intensity and position tracking method. Finally, the positions of all objects are updated based on vehicle motion before the next laser scan is acquired.

The research field of localization and path planning for mobile robot was launched over several years ago. The researchers tried to solve the localization and map building problem for mobile robot in unstructured environments to create the bi dimensional maps and also estimated the robot's location [6], [7], [8], [9]. Over the past decades, a large of the algorithm was explored and applied to the navigation and mapping for mobile robots in a real world environment. [10], [5], [12], [13], [14]. The study of navigation and mapping for autonomous robotics can be viewed as the state estimation problem, which simultaneous predicts the mobile robot posture and environmental characteristics of the location; it has important theoretical and practical value. This paper describes a navigation system which unites three important capabilities. It enables a mobile robot to avoid obstacles, map-building the environment and plan local paths around or complex obstacles while navigating.

In order to be able to localize itself in an environment, the autonomous agent needs a representation or map of the environment. The robot should obtain a map by its own since obtaining maps from CAD models or measuring them by humans can be time consuming and inaccurate. In the literature, the mobile robot mapping is usually referred to as the simultaneous localization and mapping problem (SLAM) [1], [5]. Since mapping includes both, estimating the position of the robot relative to the map and generating a map using the data collected from the sensor and also estimate the location of the robot. Most of technique developed so far has been designed for

situations in which the environment is static during the map building process. Moving obstacles, however, can lead to serious errors in the resulting maps such as spurious objects or misalignments due to localization errors. Map building prefers to reconstruct the position and shape of objects and obstacles in the unknown environment where the robot is moving. We consider the problem of creating maps with mobile robots in dynamic environments. We present a scan matching approach that generates the map based on the laser data scanned; it then also detects lines in the corrected laser data.

The localization with respect to an internal map play an important role since the robot that cannot position itself accurately is at risk from obstacles or dangerous areas that are in the map but which cannot be easily sensed. There are a number of works that addressed the localization using pose information [2], [3]. These works update the position of the vehicle based on the determination of the transformation between the pose of the robot and the laser measurements. The laser has also been used to determine natural features in indoor environments. In [4] a comparison of the behavioral monocular, Trinocular and laser in localization applications is presented. At the same time the robots should perform the navigation tasks in a minimum amount of time. Thus, sophisticated path planning techniques are needed to fulfil these requirements.

The existing methods for solving the problem of motion planning for robot systems are decoupled, which means that they first plan paths for the individual robots independently. Afterward, they check if the robots would get too close to each other if the paths were executed. In such a case the paths are recomputed to avoid these conflicts. Many decoupled methods assign priorities to the individual robots. These priorities define an order in which the paths of the robots have to be recomputed. By computing the path of a

robot, the paths of the robots with higher priority are considered as fixed. This way, the size of the search space is extremely reduced. Most of the existing prioritized decoupled methods use a fixed priority scheme.

In the path planning segment of this thesis we present an approach which determining a path that fulfills a specified direction represented by integrating of focused D* search algorithm and dynamic window local obstacle avoidance of moving objects. The moving obstacles are modelled as moving cells on the occupancy grid map and their motion is predicted by applying a procedure similar to dynamic window approach. The collision points of the robot and moving cells predicted trajectories from the new fictive obstacles in the environment, which should be avoided.

During the search, we utilize constraints between global path and local reactive obstacle avoidance algorithms and integrated into a single motion control module in order to compute the shortest and safest path from the present robot pose to any reachable point in the dynamics and unknown environments.

The remainder of this thesis is organized as follows: In the following Section is a presentation on solving the problem of map building for the robot. We present our approach to generate the environment that surround the robot based on the robot's path and matching the laser scan algorithm. After this we focus on localization environments and described in Section 3 how to updating the pose of the robot in an environment based on sensor reading. In Section 4 we demonstrate how a mobile robot can plan the path and collision avoidance in the partially unknown environment. The simulation and experimental results are given in Section 5. Finally, the conclusions are summarized in Section 6.

2. MAP BUILDING

In order to perform tasks in an environment for a mobile robot, first, it must be able to answer the question: “Where am I?” When this question is posed with respect to a known previously existing map of the environment, the problem is commonly referred to as Localization. Simultaneous Localization and Mapping (SLAM) answers the question in a more general case: where the robot has no map to begin with and must construct a map at the same time as performing Localization.

There are many different kinds of map building used for localization, are able to deal with noise in the odometry and noise in the sensor data, based on the form of sensor information and the representational requirements of localization. Traditionally, SLAM solutions involve estimating the positions of landmarks in the environment as well as the robot’s position based on measurements from robot-mounted sensors and odometry estimates. Numerous solutions to this problem have been considered such as scan matching for alignment [15], the Extended Kalman Filter (EKF) [16, 17], FastSLAM or Rao-Blackwellized Particle Filters (PF) [18, 19] and others to form the now very mature SLAM field.

However, if the objects appear or travel through the sensor range of the robot during mapping, the resulting map will contain evidence about an object at the corresponding location. Moreover, if the robot returns to this location and scans area a second time, position estimate will be less accurate. The reduced accuracy of the resulting maps may have a negative influence on the overall performance of the robot, since it can obstruct the execution of typical navigation tasks such as localization and path planning. Maps can be based on topological or metric information, or a combination of the two. Metric maps can be further refined by whether they use features or rely on

dense surface information that does not distinguish the features. The dense sensor methods [20, 21, 22] attempt to use whatever sensor information is available to generate a map and they recreate a geometric representation of the surfaces in the environment. When localization the robot on the map, dense sensor matching can take the merit of whatever surface features are present, without having to explicating decides what constitutes a landmark.

Our emphasis is on reliability, efficient techniques that can be used to create maps as the robot moves in a new environment. We concentrate on metrically precise maps that are derived from dense range readings-scanning laser range finder system. Using a recently proposed new method in global mobile robot localization, this thesis develops a Local Registration/ Global Correlation (LRGC) mapping algorithm [20] for reliable construction of consistent global maps from scanning laser data without relying on features proprietor interactions. LRGC uses scan matching (local registration), map correlation (global correlation) and consistent pose estimation for creating accurate maps in real time.

In this section, we present a Local Registration and Global Correlation (LRGC) method for determining consistent global pose estimation. This method based directly on the algorithm of the consistent pose estimation of Lu and Millios and used on two techniques to efficiently adding new information to current maps, and determining topological correct relations between the poses.

Consider the case where a consistent map has already been created, and a new pose p_n is added as Fig. 2. This new pose will have a connection to the previous pose based on odometry, and to several of the previous poses based on overlapping scan and the resultant scan matches. These relationships are highlighted in Fig. 2 by the bold arcs between poses. As long as the robot is

forging ahead and exploring new areas, these arcs all clusters into local neighborhoods that are well-connected, with no long-distance relationship.

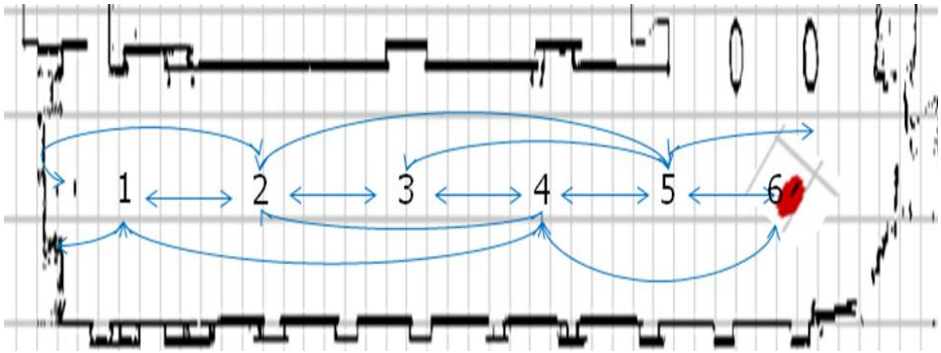


Fig. 2. Add a new pose of an unexplored environment to the map.

Local registration works well if the robot constantly explores new areas. As it completes a large cycle, however, the problem of topological correctness becomes important, because new poses must be related to the old ones. At this point, it is critical to make topological identifications reliably, because a mistake can cause the map to be badly misaligned. In general, single scans do not have enough information to yield good false positive rejection, especially if the environment is relatively uniform in one direction, as happens often along corridors. Instead, we integrate a set of local scans into a map patch, and use this more extended template to find matches on the old map.

The map patch technique is obviously more reliable than single scans in rejecting false positives, but it leaves open the question of how to efficiently perform matching, since single-scan techniques are no longer applicable. Fortunately, one author has recently investigated correlation concepts for matching map patches in the context of localization [23]. The resulting techniques have been shown to be both efficient and reliable, and we make use of them here to determine topological correctness in map-building with

cycles. Correlation operates in a “background” mode, checking for matches against the old map whenever the robot moves to an appropriate location, and adds in links between the new map and the old whenever appropriate.

The Local Registration and Global Correlation algorithm are based on three different techniques: scan matching, consistent pose estimation, and map correlation. In the following section will present the techniques in detail, showing the modifications necessary to work under LRGC. Thereafter, the overall algorithm is described and illustrated how the individual components are assembled into a practical real-time system for the mapping of dense distance information.

2.1.Scan matching

Scan matching is the process of translating and rotating a range scan, which is obtained from the SICK laser range finder. The matching algorithm returns a position probability distribution of where to place the scan in order to have the range measurements correspond to map features. There might be more than one location where a scan fits and this is expressed by the probability distribution.

It is critical that scan-matching does not overestimate the certainty of a pose, or else it can be difficult to find a consistent interpretation of a set of overlapping poses [24]. Scan matching should also produce quantitatively good results, e.g. straight lines in a corridor environment should be aligned accurately.

First, the term of the scan will be described.

Second, line segments and features extraction algorithms are presented, which operate directly on the scan data from scanning on a different position.

Third, describe the methods are used for classifying the points of a scan

and filter out unwanted scan points. In this manner, e.g. Scans are smoothed, large amount of data can be reduced without losing information, or it may be only the polygonal portion of a scan are considered.

Finally, the concept of the scan-matching method is defined and presented for covering different solution of two scans.

2.1.1. Scan

A scan is a set of measurements $\left\{ m_i = (\alpha_i, r_i)^T \mid i = 0...n-1 \right\}$ where the term of $(\alpha_i, r_i)^T$ are presented as a polar coordinates.

A scan point $m_i = (\alpha_i, r_i)^T$ can be converted to a given recording position $l = (x, y, \theta)^T$ in absolute Cartesian coordinates as follows:

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + R_2(\theta) \begin{pmatrix} r_i \cos \alpha_i \\ r_i \sin \alpha_i \end{pmatrix} \quad (2.1)$$

Fig. 3. Shows a typical scan as it is taken from a commercial 180-laser scanner with 1 degree angular resolution and range resolution 5cm.

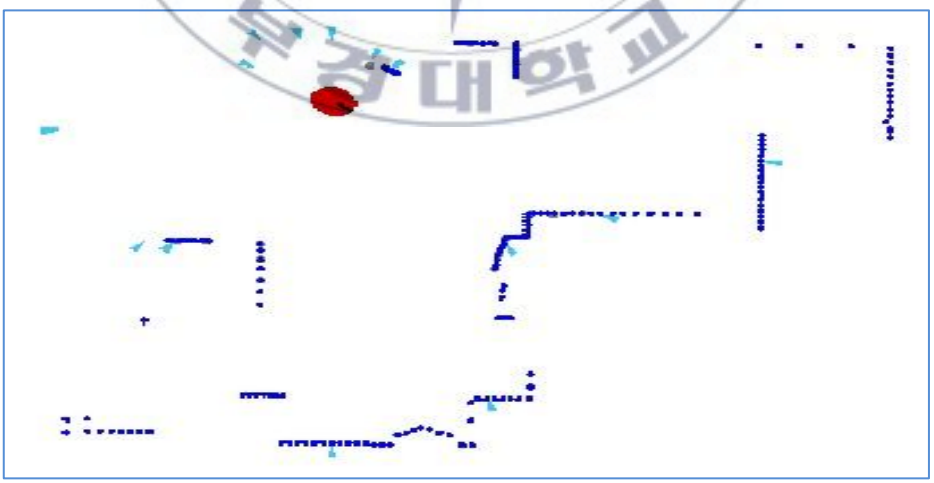


Fig. 3. A typical scan which is detected by the laser scanner 180 ° SICK LMS-200.

The measured data of a laser scanner are characterized by a high angular resolution (typically <1 °) and a high accuracy in distance measurement and therefore laser scanner is more comprehensive, more accurate and more reliable than many other distance sensors such as sonar sensor.

The measurements of a scan performed usually in a fixed number, for example with rising angle, this property is utilized in the following procedure.

2.1.2. Feature extraction from scanning data

It is always advantageous not only work directly on the individual scan points, but also extracts features from the scans. This section describes how can extract lines and corners from a scan.

2.1.2.1.Line Extraction

The following algorithm describes how line segments can be obtained from scanning points. This scan point is grouped and line segments are extracted from each group by using the split function.

Algorithm 1: line-extraction(s)

Input: Scan s

Output: Set of line segments l

Procedure:

```
l = empty;
start = 0;
for i= 1 to numpoints(s) - 1 do
    p1 = n-th-scanpoints(s,i-1);
    p2 = n-th-scanpoints(s,i);
    if distance(p1,p2)  > MAX_DISTANCE then
        l = l U split(s, start, i-1);
```

```
            start = i;
        end
    end
    l = l U split(s,start,numpoints(s)-1);
    return l
```

The constant MAX DISTANCE determines the maximum distance between two points of successive scan for grouping. The split function performs the actual problem of the line extraction:

Algorithm 2: split(s, start, end)

Input: Set of scan points defined by s, start and end

Output: Set of line segments l

Procedure:

```
l = empty;
line = make-line(s, start, end) ;
if numpoints(line) >= MIN-POINTS-ON-LINE then
        if σ (line)  <  MAX-SIGMA  then
                l := l U {line};
         else
            P_start = n-th-scanpoint(s, start) ;
            p_end  = n-th-scanpoint(s,end) ;
            i_split = start; d = 0;
            for i = start + 1 to end - 1 do
                p = n-th-scanpoint(s,i)
                if distance-to-line(p,p_start,p_end) > d then
                    i_split = i;
```

d = distance-to-line(p,p_{start},p_{end})

end

end

l = l U split (s, start, i_{split})

l = l U split(s, i_{split}, end)

end

end

return l

The split function is recursive. First, through the whole group of scan points a line of best fit (make-line) is created. If the generated line contains sufficient scan points and the standard deviation σ is not too large, the line is recorded in the line extension and returned. However, if the standard deviation is too large, but still enough points are presented, to form at least one line, then the set of scan points is divided at a certain point into two sub-groups and the split function with two subgroups called recursively. Then, the connection of the two line extension given by the split function will be returned. The scanning point at the group is divided and determined by the point which has the maximum distance to the line through the starting and ending point of the group. It is determined in the for-loop. The function distance-to-line (p, p_1, p_2) supplies and increase the distance of the point P to the straight line, which is determined through the set of points p_1 and p_2.

The constants MIN-POINTS -ON-LINE and MAX SIGMA determine the number lines and quality of the lines. A smaller value for MAX SIGMA generated more lines than a high value. Therefore, the accuracy of the lines for small values is better than for large ones. For the SICK Laser range finder LMS 200 provide MAX SIGMA values 5mm, a good compromise between low line number and high accuracy.

MIN-POINTS-ON-LINE sets the minimum number of points per line fixed. The smaller value is selected; the additional lines can be generated. However, this also increases the risk that lines are placed in wrong sensor data which have been generated by non-polygonal objects. The useful values for MIN- POINTS- ON-LINE range between 5 ~20.

The split function calls at the beginning of make-line function. This sets a regression line through the given set of scan points, the split function can be implemented as follows: Let set (x_i, y_i), i = 0… n – 1 is Cartesian coordinates of the scan points which is determined by a straight line. The straight line defined by the parameters α and d, such that for all points (x, y) is on the line:

$$x\cos\alpha + y\sin\alpha - d = 0 \tag{2.2}$$

The straight line is created which minimizes the sum of the squares of the distance

$$E_{fit} = \sum_{i=1}^{n} (x_i \cos\alpha + y_i \sin\alpha - d)^2 \tag{2.3}$$

Then, the solutions for α and d and the associated standard deviation σ are calculated as [15, 21]:

$$\alpha = \frac{1}{2}\tan^{-1}\frac{-2S_{xy}}{S_{y^2} - S_{x^2}} \tag{2.4}$$

$$d = \bar{x}\cos\alpha + \bar{y}\sin\alpha \tag{2.5}$$

$$\sigma^2 = \frac{1}{2n}\left(S_{x^2} + S_{y^2} - \sqrt{4S_{xy}^2 + \left(S_{y^2} - S_{x^2}\right)^2} \right) \tag{2.6}$$

where:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.7}$$

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{2.8}$$

$$S_{x^2} = \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (2.9)$$

$$S_{y^2} = \sum_{i=1}^{n} (y_i - \bar{y})^2 \quad (2.10)$$

$$S_{xy} = (x_i - \bar{x})(y_i - \bar{y}) \quad (2.11)$$

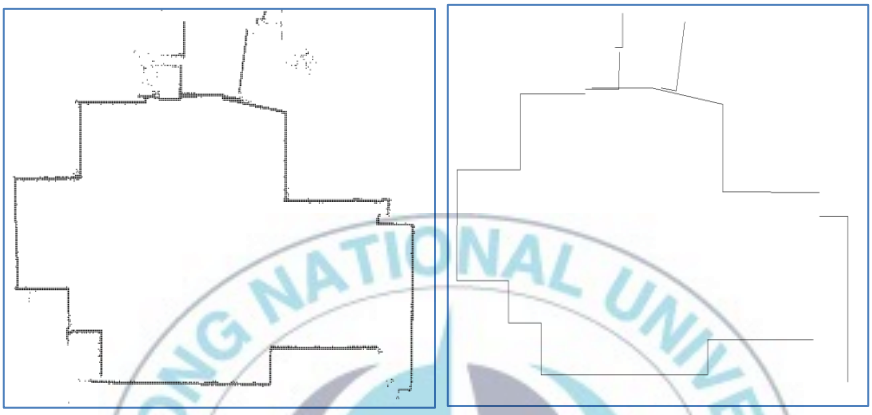


Fig. 4. Extraction lines from a scan, links raw scan, right extracted lines

2.1.2.2.Extraction of corners

In a similar way as line extraction from the scan data, corners can also be generated from straight lines intersect together. For this purpose, the split function in the line extraction must only be replaced by the following function.

Algorithm 3: split(s, start, end)

Input: group of scan points is defined by s, start and end

Output: set of edges e

Procedure:

e = empty;

if (end-start) >= 2 MIN-POINTS-CORNER then

p_{start} = n-th-scanpoints(s,start);

p_{end} = n-th-scanpoints(s,end);

i_{split} = start; d= 0;

for i = srtart + 1 to end -1 do

p= n-th-scanpoints(s,i);

if distance-to-line(p,ps_{tart},p_{end}) > d then

i_{split} = i; d = distance-to-line (p,p_{start},p_{end});

end

end

if (i_{split} - start) > = MIN-POINTS-CORNER and

(end - i_{split}) > =MIN-POINTS-CORNER then

$line_1$ = make-line(s, i_{split} - MIN-POINTS-CORNER, i_{split}) ;

$line_2$ = make-line(s, i_{split}, i_{split} + MIN-POINTS-CORNER) ;

if σ($line_1$) < MAX-SIGMA and σ($line_2$) < MAX-SIGMA then

e = e U {maks-corner ($line_1$, line2)};

end

end

e = e U split(s, start, i_{split});

e = e U split(s, i_{split}, end);

end

return e

First, the function examines whether sufficient scan points for the extraction of a corner. Then, search the line extraction from the scan point. If the straight lines through the start and end point of the group of points have the maximum distance. At this point, each line is a subset of MIN-POINTS-CORNER scan points before and after this point and formed if sufficient scan points are in two in two subgroups, each straight line is defined by these points. Finally, if the standard deviations of both lines are not too large, a

corner is generated using the make-corner function. Make-corner intersection determined the position and angle of the two lines. The parameters MAX SIGMA and MIN-POINTS-CORNER determine how number the line extraction and quality of the extracted corners.

2.1.3. Median Filter

For a number of existing scan processing algorithm, it is advantageous to edit the previously scanning with different filters, for example, to remove unwanted scan points, or to reduce the amount of data. In this section, median filter techniques are presented, which is used for smoothing the individual measurements.

The median filter is capable to detect outliers in one scan, and to be replaced by an appropriate measurement. The median filter is applied for each scan point. A window is placed around the scan point. The scan point is then replaced by a new scanning point which has the same recording angle, but to remove the scan point increases the median of the distance values in the observed window.

The following algorithm implements the median filter:

Algorithm 4: median-filter(s)

Input: scan s

Output: scan s'

Procedure:

```
for i = 0 to numpoints(s) - 1 do
        p = n-th-scanpoint(s,i);
        for j = 0 to MEDIAN-NUM-POINTS - 1 do
                k=(i+j-MEDIAN-NUM-POINTS/2)mod numpoints(s)
                p_k = n-th-scanpoint(s, k);
```

```
                d(j) = distance-value(pk) ;
            end
            dmedian = median{d};
            n-th-scanpoint(s',i) = (angle-value(p), dmedian);
        end
        return s'
```

The parameter MEDIAN-NUM-POINTS determines the window size at which the median filter operates as well. A larger value indicates a strong smoothing of the scan, i.e. there are no more long distance fluctuations. A low value allows large variations in the distance, but at the same time be less outliers detected. In practice, we chose the value of MEDIAN-NUM-POINTS = 5.

The advantages of the median filter are in the removal of outliers and in the reduction of sensor noise and it shapes corners are rounded. The median filter often used to remove noise. Such noise reduction is a typical pre-processing step to improve the results of the later processing such as object recognition, segmentation, and feature extraction etc. Here the median filter is applied in order to obtain smoother contours.

2.1.4. Scan matching

In this section, scan matching techniques are presented. The methods procedures a scan match with another one. The problem is specified in more detail in the following. There are several different solution methods for Scan matching that have been developed and applied in the past. The Cox algorithm, which has been covered a scan match with a line model and extended for the matching of scans. The iterative dual correspondence (IDC)

algorithm matching scans directly to other scan. The combination of these two methods allows improving advantage and avoiding disadvantage of them.

Given two scan s and t. Scan s is denoted as a current scan and t is a reference scan. The receiving position of t defines the coordinate system of t. Now search for mapping match: $l \to p$, which position l in the coordinate system of it upon the maps probabilities $p \in [0,1]$ that indicate how well scan s and t match if s is replaced and rotated position l.

Thus, scan matching is a sub-problem matching in a multidimensional space. Through the determination a probability distribution, the scan poses can be expressed ambiguities, e.g. it may be that there are several positions where match the scans.

Most of scans matching method simplify the property issue; two assumptions are described in the following:

- ✓ Gaussian distribution: The distribution function, which is defined by match, can be approximated by Gaussian function. This has the consequence that only the first two moments (mean and variance) must be determined. However, no more arbitrary distributions are modeled, which is especially detrimental when several hypotheses exist.
- ✓ Locality Assumption: It is believed that the approximate position of the recording, s is in the coordinate system of t (by odometry) and only a local adjustment is necessary. This enables use of local search method, which often also has a closed-form solution.

These two scan matching assumptions can also be defined as a function of the scan - match which depicts two scans match on a Gaussian function with mean μ_{match} and covariance $\sum_{match}$ matrix.

$$scan - match(s,t) = \left(\mu_{match}, \sum_{match}\right) \quad (2.12)$$

$$\mu_{match} = \begin{bmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{bmatrix}^T \tag{2.13}$$

$$\sum_{match} \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 \end{bmatrix} \tag{2.14}$$

The vector $\begin{bmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{bmatrix}^T$ indicates the position to be replaced, and rotated the scan s in the coordinate system of t so that a maximum overlap of the scan is generated. The miss match component $\sum_{match}$ is a measure of the accuracy of the calculated match. Thus the calculated probability function is a normal distribution, which is derived from the calculated mean and error covariance matrix:

$$\begin{bmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{bmatrix}^T \sim N\left(\begin{bmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{bmatrix}^T, \sum_{match}\right) \tag{2.15}$$

2.1.4.1. Extended Cox algorithm

A process for matching a scan line with a model was developed by Cox [27]. Points of the current scan are matched against a prior model consisting of line segments and calculated a position correction.

First, the original method of Cox is presented. Thereafter, we described the extension algorithm which is usable for performing the pairs of scan matching.

2.1.4.1.1. Cox algorithm

The Cox algorithm is the original procedure for matching a scan line with a priori line model. Each scan point will be assigned to a line of the priori model. Then the displacement and rotation of the scan over the line model can be determined from this assignment. This method requires an

approximate initial estimate of the receiving position is obtained from odometry.

The process can be divided into the following steps:

1. Set $\begin{bmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{bmatrix}^T = \begin{bmatrix} s_x & s_y & s_\theta \end{bmatrix}^T$. Where, $\begin{bmatrix} s_x & s_y & s_\theta \end{bmatrix}^T$ is the initial position estimate of the scan recording, which is provided by odometry.
2. Translate and rotate the scan position $\begin{bmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{bmatrix}^T$
3. Determine the model line for each scan point that is nearest to the point. This model line will be referred to below as the target line.
4. Compute the transformation $\hat{b} = \begin{bmatrix} \Delta x & \Delta y & \Delta\theta \end{bmatrix}^T$, which minimizes the sum of the distance between the scanning points, and squares of the target line, respectively.
5. Set $\begin{bmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{bmatrix}^T = \begin{bmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{bmatrix}^T + \begin{bmatrix} \Delta x & \Delta y & \Delta\theta \end{bmatrix}^T$.
6. Repeat step 2-5 until the procedure converge. The result of the overlap is $\begin{bmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{bmatrix}^T$
7. Calculate error covariance $\sum_{match}$

Determining the target line

In determining the goal line for a scan point, the line is selected that has the smallest distance (Euclidean) from the set of scan points. If this distance exceeds a predetermined maximum distance d_{max}, there is no correlation. In this case, the scan point treated as an outlier and removed for further steps.

Determining the transformation

The calculation of the transformation in Step 4 is constructed as a trans function which rotates the scanning at an angle $\Delta\theta$ and translates with a

vector $\Delta t = \begin{bmatrix} \Delta x & \Delta y \end{bmatrix}^T$. However, the rotation is always around the position $l_s = \begin{bmatrix} s_x & s_y \end{bmatrix}^T$ of scan recording, the function trans emulates a scan point $p_i = \begin{bmatrix} p_{ix} & p_{iy} \end{bmatrix}^T$ as follows:

$$trans(\Delta t, \Delta\theta)(p_i) = \begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) \\ \sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} (p_i - l_s) + l_s + \Delta t \qquad (2.16)$$

Under the assumption that the rotation angle $\Delta\theta$ is small, the trans function can be approximated as:

$$trans(\Delta t, \Delta\theta)(p_i) = \begin{bmatrix} 1 & -\Delta\theta \\ \Delta\theta & 1 \end{bmatrix} (p_i - l_s) + l_s + \Delta t \qquad (2.17)$$

At each scan point, the target line is given by the parameter $u_i = \begin{bmatrix} u_{ix} & u_{iy} \end{bmatrix}^T$ and r_i is all the points z on the line so that $z^T u_i = r_i$. For simplicity, we assume that the line is infinitely distance (in the determination of the target line in step 1 but assumed the finite line segments), the squared distance of a scan point p_i to the target line can be calculated as follows:

$$dist^2 = \left(\left(trans(\Delta t, \Delta\theta)(p_i) \right)^T u_i - r_i \right)^2 \qquad (2.18)$$

Substituting equation (2.17) into (2.18) we obtained:

$$dist^2 = \left(\left(\begin{bmatrix} 1 & -\Delta\theta \\ \Delta\theta & 1 \end{bmatrix} (p_i - l_s) + l_s + \Delta t \right)^T u_i - r_i \right)^2 \qquad (2.19)$$

$$dist^2 = \left(\left(\Delta t + \Delta\theta \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} (p_i - l_s) + p_i \right)^T u_i - r_i \right)^2$$

$$= \left(\left(\Delta x - \Delta\theta (p_{iy} - s_y) + p_{ix} \right) u_{ix} + \left(\Delta y + \Delta\theta (p_{ix} - s_x) + p_{iy} \right) u_{iy} - r_i \right)^2$$

$$= \left(\left(x_{i1}, x_{i2}, x_{i3} \right) b - y_i \right)^2 \qquad (2.20)$$

where:

$$x_{i1} = u_{ix} \tag{2.21}$$

$$x_{i2} = u_{iy} \tag{2.22}$$

$$x_{i3} = -(p_{iy} - s_y)u_{ix} + (p_{ix} - s_x)u_{iy} \tag{2.23}$$

$$x_{i3} = -(p_{iy} - s_y)u_{ix} + (p_{ix} - s_x)u_{iy} \tag{2.24}$$

$$b = (\Delta x, \Delta y, \Delta\theta)^T \tag{2.25}$$

Then, the sum of squares distance E_{fit} between scan points and target line for n scan points $p_i \ldots p_n$ is given by:

$$E_{fit}(b) = \sum_{i=1}^{n} \left((x_{i1}, x_{i2}, x_{i3})b - y_i \right)^2 = (Xb - Y)^T (Xb - Y) \tag{2.26}$$

where:

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ \vdots & \vdots & \vdots \\ x_{n1} & x_{n1} & x_{n1} \end{pmatrix} \tag{2.27}$$

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{2.28}$$

The function E_{fit} must be minimized and the associated transformation b is determined. Differentiates E_{fit} according to b and sets the resulting expression equal to zero:

$$\frac{\partial E_{fit}(\hat{b})}{\partial b} = 0 \tag{2.29}$$

$$2X^T(X\hat{b} - Y) = 0 \tag{2.30}$$

$$X^T X\hat{b} = X^T Y \tag{2.31}$$

$$\hat{b} = (X^T X)^{-1} X^T Y \tag{2.32}$$

The vector b is exactly the desired transformation in step 4

Convergence and error covariance

Steps 2-5 of the Cox algorithm is repeated until converges the algorithm. For convergence, we consider the calculated transformation vector $\hat{b}$ and checks if it is small enough. This requires two conditions as follows:

$$\sqrt{\Delta x^2 + \Delta y^2} < \epsilon_{dist} \tag{2.33}$$

$$|\Delta\theta| < \epsilon_{\theta} \tag{2.34}$$

The values of ϵ_{dist} ex and ϵ_{θ} depend on the laser scanner and must be chosen appropriately. For commercially available laser scanners provide values from 1mm - 10mm for e ϵ_{dist} and 0.5°~ 1.0 ° for ϵ_{θ} to achieve good results.

In addition, Cox method can use the result vector $\begin{bmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{bmatrix}^T$ in order to calculate the error covariance matrix $\sum_{match}$ which indicates the accuracy of the estimated transformation. This is calculated as [27]

$$\sum_{match} = \frac{1}{n-4}(Y - X\hat{b})^T (Y - X\hat{b})(X^T X)^{-1} \tag{2.35}$$

2.1.4.1.2. Extension of the Cox algorithm

An obvious extension of the Cox algorithm by using a pair of overlap scans extracted from the reference scan lines and used them as a priori model for Cox algorithm.

Fig. 5. Shows the structure of the extended Cox algorithm. The Cox algorithm forwarded as a priori model from the reference scan line and a line model obtained. The current scan is preprocessed with line filter before applying to the Cox algorithm as well. In this way, model lines are minimized

wrong overlap of scan points. As a result is obtained by mean and covariance matrix of the scan overlap error also an error value, which indicates the quality of the computed overlap. This value is calculated as the median of all distances assignments of scan points to model lines according to overlap, a high error value is so bad overlap since the distances of scan points to model lines are large, a low value indicates a good matching.

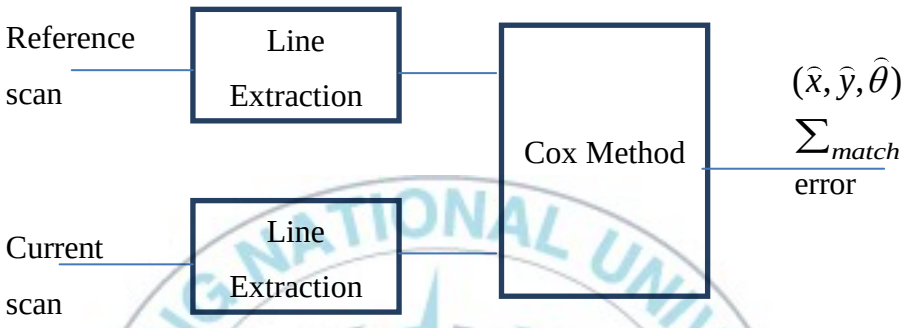


Fig. 5. Scheme of extended Cox algorithm

To reduce a number of incorrect matching, the Cox algorithm has been extended by two heuristics. First, the current scan is filtered through a line filter, that is, scan points that are not on a line segment are removed. This reduces the amount of false point to line segment assignments in situations where the environment is not totally polygon. Second, a hard coded threshold of dmax is used to remove assignments that have a larger distance than dmax.

2.1.4.2.IDC Algorithm

IDC algorithm is an effective method for matching pair of scans, which do not reply on a geometric interpretation of the two scans, was developed by Lu and Milios [15, 21, 28]. In particular, the algorithm requires no features in the scan data. Thus, this method is suitable in many environments.

2.1.4.2.1. General approach

The general concept of IDC algorithm scans points of the current scan matched with the reference scan points. In this case, this matching is similar to the Cox algorithm, a minimize sum of error and the displacement and rotation of the scans can be determined. For matching the set of scan points, two heuristic function is used: closest-point rule and matching-range rule.

2.1.4.2.2. Closest-point rule

For each scan point of the current scan determines the point in the reference scan which the scanning point is closest. There is interpolated between the scanning points of the reference scan. It connects the scan point with short line segments, therefore a partner must be not exactly a scanning point of the reference scan, but a point on the line may be connected with two adjacent nodes of scanning points.

The algorithm is very simple to determine the match partners of scan point p with the current scan. Consider two successive scan points p1 and p2 of the reference scan and calculate the distance from p to the line segment (p1, p2). The line segment with the smallest distance will contain this point, under the assumption, the scanning is only rotated a little, and the search area can be significantly reduced by the closest point. In addition, only scan point pi of the reference scans are considered, their angle with respect to the initial position of the current scan in an area around the angle of p. Mathematically, this means that the absolute value of the angle between the straight lines (l_s, p) and (l_s, p_i) must be less than an angular tolerance ω where l_s is the receiving position of the current scan. For implementation, it is advantageous if the scan points of the reference scan are existed with ascending receiving

angle (with respect to l_s), since the use of this property allows the efficient implementation of the search range limitation.

2.1.4.2.3. Matching-range rule

The second heuristic is assigned to each scan point of the current scan with the point in the reference scan, which has the same distance with the location point l_s. There is a re-interpolated between the scanning point and the reference scans. If no point is found, this means that these points have the same distance. So that, points are chosen which distance is closer to the desired distance.

The algorithm to determine the match partner for a scan point p of the current scan is designed similar to the closest- point rule. Here are only examined points p_i of reference scans, which lie in the angular interval described above. In this way can also be avoided completely incorrect matching. The determination of the match partner is presented as follows:

Algorithm 6: find-matching-point (p,p_i)

Input: scan point p; scan points p_i, $i \in \{1,...,n\}$

Output: point p'

Procedure:

d = distane(p,l_s);

d'= ∞ ; $d_{pp'} = \infty$;

for i = 1 to n -1 do

d_1= distance(p_i,l_s);

d_2 = distance(p_{i+1},l_s);

if (d_1 < d and d_2 < d) or (d_1 > d and d_2 > d) then

if | d_1 – d| < | d_2 – d| then

$d_h = |d_1 - d|$; $p_h = p_i$

else

$d_h = |d_2 - d|$; $p_h = p_{i+1}$;

end

else

p_h = interpolate(p_i, p_{i+1},d)

if d' > 0 or distance (p,p_h) < $d_{pp'}$ then

d' = 0;

$d_{pp'}$ = distance(p,p_h);

p' = p_h;

end

end

end

return p'

The algorithm examines all n-1 pairs of successive scan points in the maximum limit angular interval, for each pairs, the distance d1 and d2 of the position point is computed. If both distances are smaller or larger than the desired distance d, this pairs of distance cannot find the correspond point. If still not found better p_{int}, in this case, the point p_1 and p_{i+1} chooses as a temporary match partner which distance is closer to the desired distance. If the distances of the two points scanning, however, by the way that values less than and the other is greater than the desired value, the interpolate function can be interpolated between point p_1 and p_{i+1} which has exactly desired distance. There may have several pairs (p_i, p_{i+1}) including the distance distribution. In this case, there is ambiguous correspondence.

2.1.4.2.4. Determination of the rotation and displacement

Once set of pairs correspondence (pi,pi'), i= 1…n have been determined by one of the among rules described above, hence, the rotation and translation can be calculated by minimizing a sum of squares of the distance. The points pi of the current scan is also rotated and translated with $\Delta\theta$ and $\Delta t = \begin{bmatrix} \Delta x & \Delta y \end{bmatrix}^T$, respectively. Return to the receiving position of the scans ls. Now, the sum of the square distance between the rotation and translation points pi and pi' are constructed.

$$E_{fit}(\Delta t, \Delta\theta) = \sum_{i=1}^{n} | R_2(\theta)(p_i - l_s) + l_s + \Delta t - p_i^{'} |^2 \tag{2.36}$$

This function only needs to be minimized. The right side parameters Δt and $\Delta\theta$ are calculate directly to [28]

$$\Delta\theta = \tan^{-1} \frac{S_{xy'} - S_{yx'}}{S_{xx'} + S_{yy'}} \tag{2.37}$$

$$\Delta t = \bar{p}' - l_s - R_2(\theta)(\bar{p} - l_s) \tag{2.38}$$

$$\bar{p} = \frac{1}{n} \sum_{i=1}^{n} p_i \tag{2.39}$$

$$\bar{p} = \frac{1}{n} \sum_{i=1}^{n} p_i^{'} \tag{2.40}$$

$$\begin{pmatrix} S_{xx'} & S_{xy'} \\ S_{yx'} & S_{yy'} \end{pmatrix} = \sum_{i=1}^{n} (p_i - \bar{p})(p_i^{'} - \bar{p}') \tag{2.41}$$

It was found that the closest-point rule is particularly suitable for determining the displacement and the matching-range-point rule is well suited for determining the rotation. Therefore, both methods were combined and designed for the IDC- algorithm.

2.1.4.2.5. IDC algorithm

The IDC algorithm can be formulated as follows:

1. Set $\begin{bmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{bmatrix}^T = \begin{bmatrix} s_x & s_y & s_\theta \end{bmatrix}^T$. Here $\begin{bmatrix} s_x & s_y & s_\theta \end{bmatrix}^T$ is the initial estimate of the receiving position of the current scan which is given by odometry.
2. Translation and rotation current scan position $\begin{bmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{bmatrix}^T$
3. For each scan point p, determine the correspondence $p_i^{'}$ towards the closest-point rule and apply the matching-range-point rule to determine a correspondence point $P_i^{"}$.
4. Compute the solution $(\Delta x_1, \Delta y_1, \Delta \theta_1)^T$ from the minimum sum of square distance between the set of correspondence pairs $(p_i, p_i^{'})$.
5. Compute the solution $(\Delta x_2, \Delta y_2, \Delta \theta_2)^T$ from the minimum the sum of square distance between the set of correspondence pairs $(p_i, p_i^{"})$.
6. Set $\begin{bmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{bmatrix}^T = \begin{bmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{bmatrix}^T + \begin{bmatrix} \Delta x_1 & \Delta y_1 & \Delta \theta_2 \end{bmatrix}^T$
7. Repeat step 2-6 until the algorithm converges. The result of the overlap is $\begin{bmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{bmatrix}^T$
8. Compute error covariance $\sum_{match}$

The IDC –Algorithm is to take the translation component from the closest-point rule solution and the rotation component from the matching-range-rule solution to form the current solution for transformation.

2.1.4.2.6. Error covariance

The error covariance matrix $\sum_{match}$ is calculated from the correspondence

pairs $(p_i, p_i^{'})$, i = 1 … n as follows [28]:

$$\sum_{match} = s^2 (M^T M)^{-1} \tag{2.42}$$

where:

$$M = \begin{pmatrix} M_1 \\ \vdots \\ M_n \end{pmatrix} \tag{2.43}$$

$$M_i = \begin{bmatrix} 1 & 0 & -y_i \\ 0 & 1 & x_i \end{bmatrix} \tag{2.44}$$

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \frac{1}{2}(p_i + p_i^{'}) \tag{2.45}$$

$$s^2 = \frac{(Z - M\bar{D})^T (Z - M\bar{D})}{2n - 3} \tag{2.46}$$

$$Z = \begin{pmatrix} p_1 - p_1^{'} \\ \vdots \\ p_n - p_n^{'} \end{pmatrix} \tag{2.47}$$

$$\bar{D} = (M^T M)^{-1} M^T Z \tag{2.48}$$

Because of the simultaneous application of both heuristics, there are two sets of correspondence pairs. It is not much difference exist between two sets of correspondence pairs. Therefore, any of these two quantity correspondence are used for calculating the error covariance matrix.

2.1.4.2.7. Extended IDC algorithm

An obvious optimization of the IDC- Algorithm is to edit the both previously scans with median filter. In this way, the number of scan points is greatly reduced without losing essential information. Furthermore, as Cox

algorithm, an error value is determined error which is calculated as the median distance of scan point assignments for overlapping. This value is smaller when two scans are good overlap, and large when the overlap is weak. Fig. 6. Shows the schematic structure of the extended IDC algorithm

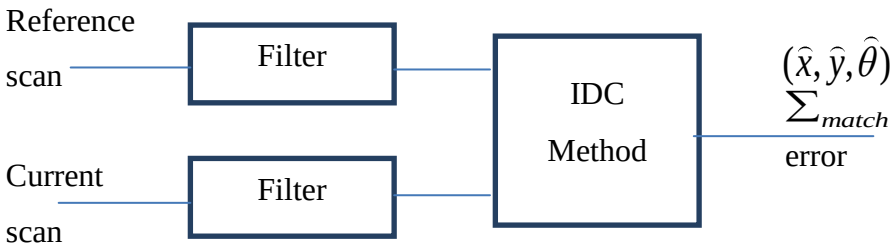


Fig. 6. Schematic structure of the extended IDC algorithm

2.1.4.3.Combined scan matching algorithm

In the previous section, both scans matching algorithms are presented. These techniques are appropriate for exploiting polygonal structures where the overlaps are determined for all variability, given accurate results and compared with neither algorithm in order to provide significantly better results. The Cox method requires a lower complexity and runtime as the IDC algorithm. In non-polygonal environment the IDC algorithm yields better results than the Cox algorithm. However, the IDC method in a polygonal environment does not reduce all variance, for example, a neighborhood with a long corridor, extremely optimistic.

Therefore, the combination of the Cox algorithm which developed for polygonal environments with the IDC method which works well even in non-polygonal environments are constructed, call combine scan matching method

Fig. 7. Shows the schematic structure of this combined scan-matching method, the core of the process is a decision logic, which investigates to overlapping scans and then uses one of two methods Cox Scan-Matching or

IDC Scan-Matching in order to calculate overlap. For the decision logic of the two scans each line segments are extracted, and the percentage of the line segments calculated in the total extent of the scans. If there are enough scan points lying on line segments then the extended Cox algorithm is used. Otherwise, the extended IDC is used.

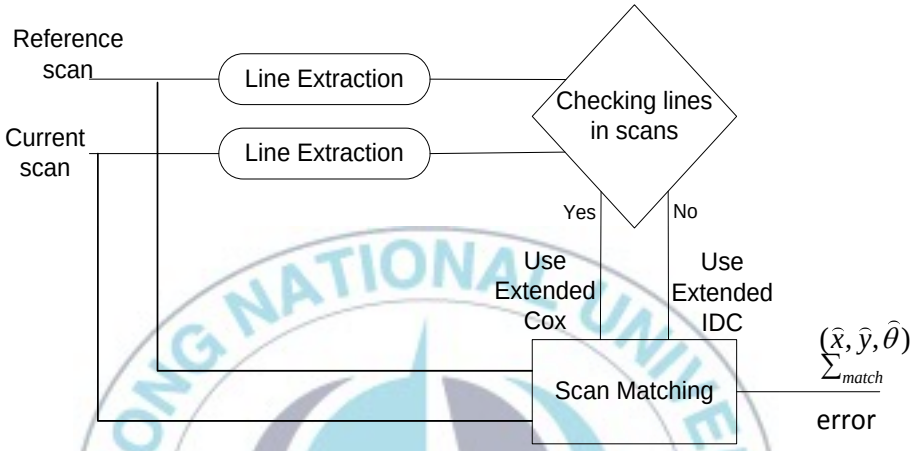


Fig. 7. Schematic of the combined scan matching method

2.2.Consistent position estimate

Lu and Milios following a different approach to the creation of the map extracted from sensor data. They consider the full pose set p, and try to global optimize p based on how well neighboring sensor scans match but it can be directly processed the raw data of a laser scanner. There is no explicit estimation of the map m; instead, the scans themselves are an explicit representation of the map surfaces. The whole process is called consistent pose estimation, since it finds a set of posed that minimizes the total error of the system. Error terms come from robot motion, and also overlapping scans: the better the scans match, the lower the error. The advantage here is that the method can be used in any environment and not any features that must be presented in the environment requires. The approach uses a combination of

relational-based and position-based representations, relations between positions obtained by odometry information and scan matching, while the positions are even free variables and are determined by solving an optimization problem.

The knowledge acquired during odometry and scan matching relations between positions a network is created that consists of positions as nodes and relationships as links. In general, the relationships in this network are inconsistent or contradictory, since relationships are not independent variables and are subject to errors. The task now is to determine all positions so that the inconsistencies are resolved as far as possible, i.e. be minimal. This is achieved sum of error, which contains all the positions as free variables and each relationship is transformed into an expression that can be regarded as a spring between two positions. A spring has minimum energy when the parties' positions correspond exactly to the relationship. The sum of error then calculates the total energy in the network, and the positions are determined so that this energy is minimized.

2.2.1. Definition of the estimation problem

The estimation problem is considered as the following generic optimal estimation problem. Assume that a network with uncertain measurement $n+1$ nodes $X_0, X_1, ..., X_n$, each node X_i represents a d-dimensional position vector. A link D_{ij} between two nodes X_i and X_j represents a measurable difference of the two positions. Generally, D_{ij} is a function of X_i and X_j, which may be non-linear.

We model an observation of D_{ij} as $\bar{D}_{ij} = D_{ij} + \Delta D_{ij}$, where ΔD_{ij} is a random Gaussian-distributed error with zero mean and known covariance

matrix C_{ij}. The goal now is to determine the best estimate of all positions given a set of measurements $\bar{D}_{ij}$ and covariance C_{ij}. In addition, the covariance matrices of the estimated position vectors are determined based on the covariance matrices of the measurements.

The optimality criterion is the minimum variance (or maximum likelihood). The node X_i is determined in such a way that the conditional probability of the derived D_{ij}, given their observations $\bar{D}_{ij}$, is maximized. Assuming that all the observational error and Gaussian distributed independently, the optimality criterion is equivalent to minimizing the following Mahalanobis distance:

$$W = \sum_{(i,j)} \left(D_{ij} - \bar{D}_{ij} \right)^T C_{ij}^{-1} \left(D_{ij} - \bar{D}_{ij} \right) \tag{2.49}$$

where W is the sum of all measurement overflows. In this case, a pair of nodes D_{ij} is no observation, the inverse of the associated covariance matrix is set to zero, $C_{ij}^{-1} = 0$.

By the linearity assumption, the measurement equation can be rewritten as the following formulas:

$$W = \sum_{(i,j)} \left(X_i - X_j - \bar{D}_{ij} \right)^T C_{ij}^{-1} \left(X_i - X_j - \bar{D}_{ij} \right) \tag{2.50}$$

Since only relative information is used in the function W, a position can be freely selected. Without loss of generality, $X_0 = 0$ and $X_1, ..., X_n$ positions are relative to X_0.

Noted that the measurement equations in matrix form as

$$D = HX \tag{2.51}$$

where X is an nd-dimensional vector consisting of the concatenation of $X_1, ..., X_n$, D is the concatenation of all the positional differences of the form

$D_{ij} = X_i - X_j$ and H is an incidence matrix which all entries being 1, -1 or 0. Then the function W is formulated in follows matrix form:

$$W = \left(\bar{D} - HX\right)^T C^{-1} \left(\bar{D} - HX\right) \tag{2.52}$$

Here $\bar{D}$ is the concatenation of all observations and C is the covariance matrix of $\bar{D}$ which is a square matrix consists of C_{ij} as sub-matrices.

The solution for X which minimizes W is given by

$$X = \left(H^T C^{-1} H\right)^{-1} H^T C^{-1} \bar{D} \tag{2.53}$$

and the covariance of X as

$$C_X = \left(H^T C^{-1} H\right)^{-1} \tag{2.54}$$

If the measurement errors are independent, C will be block-diagonal matrix and the solution can be simplified. Let G is nd x nd matrix $H^T C^{-1} H$. The dxd sub-matrices of G may be determined by

$$G_{ij} = -C_{ij}^{-1} \tag{2.55}$$

$$G_{ii} = \sum_{j=0}^{n} C_{ij}^{-1} \tag{2.56}$$

In addition, let B is the nd-dimensional vector $H^T C^{-1} \bar{D}$, the d-dimensional sub-vectors of B can be determined by

$$B_i = \sum_{j=0; j\#i}^{n} C_{ij}^{-1} \bar{D}_{ij} \tag{2.57}$$

Then the position estimate X and covariance matrix C_X are obtained:

$$X = G^{-1} B \tag{2.58}$$

$$C_X = G^{-1} \tag{2.59}$$

Above equation requires $G = H^T C^{-1} H$ is invertible matrix. Lu and Milios suggest that it is possible to prove the inevitability of G, where the network is

fully connected and the individual error covariance are normally behaving.

2.2.2. Application for the mapping

A typical application of this optimal estimation is in mobile robot navigation, to estimate the robot position $(x_i, y_i, \theta_i)^T$ with position uncertainties Σ_i at different time i. The observations are relative to position information which determined from scan matching. Then the method can record positions that are determined from data taken during travel distance and entries the data into a global coordinate system, a consistent environment map is created.

In this application, the measurement equation D_{ij} is non-linear because of the angle θ component in the robot pose. Lu and Milios [15], however, show how the linearized measurement equation in this case and the optimization method can be applied. Since the linearization errors occur which is proportional to the initial position estimate, the method is applied iteratively. In practice four or five iterations are sufficient to converge on the result precision.

The time complexity of the consistent position estimate for a record of n scan consists of the time for the development of the network and to determine the optimum positions. For the assembly of each pair of scanning must be a check whether there is enough overlap exists, and, if necessary, a scan overlap is performed.

This method requires a good initial estimate of the scan pose in order to generate useful results. Therefore, it is used for two different purposes:

1. In order to create local maps patches of the previous few scans the robot obtained. In this case the scan pose are always topological

correct data, since very little odometrical error has been accumulated. Even though if the larger odometry errors, use of scan matching and local registration can often recover the correct geometry.

2. For closing a loop after topological relationships are obtained from the map correlation. In this case, consistent pose estimate is first run with the new links added to the map and then, after closing the loop leading to a topological correct map, return with new scan matches between the newly linked poses for fine-tuning the map.

A typical network topology is shown in Figure 2.4. If a new pose l_n is added to the chain, so only the last K scan pose for updating the map is used. Several properties of this incremental update should be obtained. First, the computation complexity at each step is constant since the number of modes is limited. In particular, the computational cost is independent of the map size, since the overlap of the individual scans is very efficient; the whole procedure is very quick.

1.3 Map Correlation

Map correlation is the main reason for matching a patch that integrates several scans and also for providing post match filters to reject false positives. To determine the relationship between poses that close a loop, a recent portion of the maps around the current pose created from the robot compared with the previous parts of the pre-generated map. Where there is a good match, it is likely that the new pose is topologically connected to one of the older poses.

In the current method of the LRGC algorithm, once a topological connection is made, it is not possible inserted or removed it, since the closing of a loop all poses are updated and no history are maintained. For this reason,

any such connection needs to be very certain before it is made. This is the main reason for matching a patch that integrates several scans and also for providing post match filters to reject false positives.

A further constraint on map matching is that it must be efficient. The correlation [23] procedure has provided a fast and precise matching. The justification for correlation technique lies in a Bayesian analysis of the match probability. For slightly given new map patch r and previous map m, the posterior probability is sought that the robot is at pose l. From the Bayes' rule, the posterior probability can be obtained as

$$p(l/r,m) = k.p(r/l,m)\,p(l,m) \tag{2.60}$$

Here gives the response function $p(l/r,m)$ is the probability that we would see the map patch r from the robot pose l, given the old map m. As shown in [23], the sensor model can be approximated by a correlation operator. A regular grid is imposed on the map area, and for each cell i, the occupancy probability $p(r_i)$ of the map patch impinging on the cell is calculated and $p(m_i)$ of the previous map impinging on the cell i. The correlation operator is:

$$corr(r,m) = \sum_i p(r_i)\,p(m_i) \tag{2.61}$$

In practice it is convenient to put all the uncertainty into the map probability $p(m_i)$, simplifying the above sum and leads to an optimized implementation (for details, see [23]).

In general, the probability should sum to less than one over the match area, that is, the robot's current map patch doesn't overlap with the old map. In order to estimate how the patch map doesn't match, the correlated response will be normalized, and then use filters to reject false positives if there is unsatisfied match.

1.4 LRGC algorithm

Fig. 8. Shows the basic scheme used for updating the map when a new scan is received from the laser range finder. A map is represented here as an undirected graph, where nodes are robot pose with associated as scans and links are constraints between pose obtained from scan matching or correlation map. The empty graph is used as the initial map.

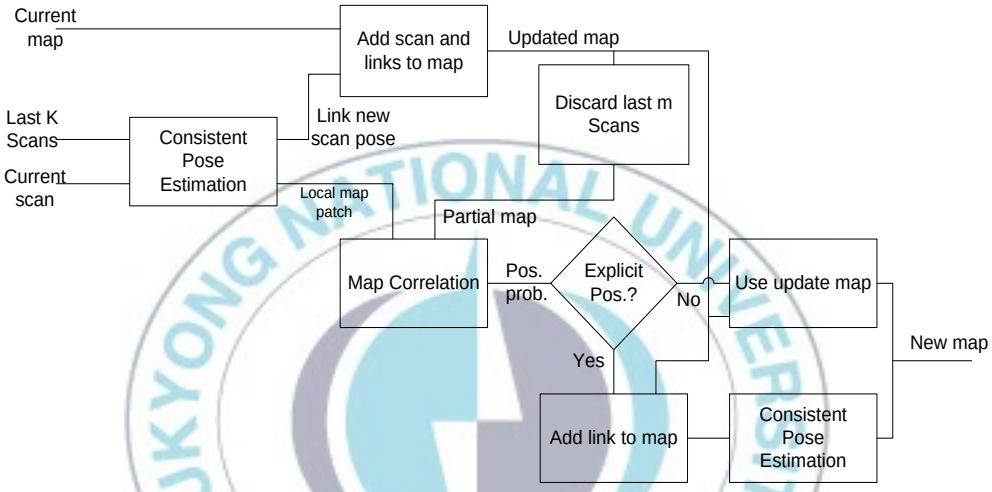


Fig. 8. Data flow of the LRGC mapping algorithm

When a new scan is added to the map, it is first registered with the last K scans to align and to improve the position estimation from the odometry. Then, the new scan pose is added together with its links to the current map. This result leads to updated map.

Loop detection is implemented in the remaining part of the flow chart. Form the updated, an old map is extracted that is assumed to be topological correct. This is done by discarding the last m scans (with m > K) to avoid recent scans are available in the old map. A local path is also created from the newest scans and correlation with the old map. The resulting position

probability distribution is examined according to the filter described in the previous section. If the highest peak passed through the filter, it can be assumed that a topological relation has been found. In this case the relation is added to the map and consistent position estimate used to close the loop, and adjust the map.

For finding topological relations, the search space is limited to an area around the current robot pose. This region grows with the pose uncertainty of the robot. The pose uncertainty is modeled with a Gaussian distribution and only test topological relations for pose that have a Mahalanobis distance to the robot pose smaller than a given threshold. Moreover, the local map size is adjusted linearly with the position uncertainty to compensate for possible ambiguities in large search spaces. Therefore, larger cycles are only closed if it exists a good evidence for topological relationship. Once a cycle has been closed, position uncertainty decreases and search space and path size fall back to small values automatically.

At the end of map building, after all the scans have been integrated, the map can be further optimized by applying the consistent pose estimation overall scans poses.

2. LOCALIZATION

Localization is the process of updating the pose of the robot in an environment based on sensor readings. The localization with respect to an internal map play an important role since the robot that cannot position itself accurately is at risk from obstacles or dangerous areas that are in the map but which cannot be easily sensed.

In order to build maps, good localization is required. Currently, a majority of mobile robots odometric information is not accurate enough for reasonable localization. As the identification of the moving objects is based on previous maps of the same region, accuracy errors to determine the exact position of the robot can lead to mistakes such as considering static parts of the environment as moving objects.

This section discusses a method for localization of a mobile robot. The assignment here is to determine the position of the robot based on a prior map and sensor data from the robot.

Generally, there are two different localization problems: global-localization and local-localization. In the global-localization, the robot adds any location and the system will be given the opportunity to observe the environment by the robot's sensors. Then, the system must decide what a possible position of the robot is by evaluating the sensor information. The process to make this decision is usually cumbersome and required depending on the size of the search space corresponding to large amount of computing time.

On the other hand, the local-localization, the approximate position of the robot is known and it is "only" a position correction can be calculated. In this case, when the robot is placed at an approximately known position and then

continuously determines its position by comparing the sensor data with environment map.

This research deals mainly with local-localization methods. Firstly, a well-known localization method, dead reckoning, presented which evaluates the odometry sensor data. This method is utilized in most robotics systems. Then, a method is discussed in more detail, scan matching and Kalman filtering, used to implement a local position determination. A sample runs in a dynamic environment which demonstrates the robustness of the scan matching methods.

3.1 Dead Reckoning

In the dead reckoning navigation, the position change of a vehicle is determined by measuring the distance traveled of one or a number of wheels. For this purpose, the wheels are mounted sensors that measure the rotational movement of the respective wheel. Additionally, a gyro may be used to determine the orientation of the robot reliably.

3.1.1 Tricycle kinematics

Fig.9. Shows a robot with three-wheel kinematics. This kinematics are often used in mobile robotics. For example, it comes in the Pioneer 3DX robots, which has been used for numerous experiments in this work. The following is discussed more detail in this kinematics. The results can also be applied to other system kinematics.

The two front wheels on Pioneer 3DX robot are the drive wheels and equipped with sensors to determine the distance traveled. The rear wheel is a freely rotating and idle wheel as it is, for example, used in office chairs.

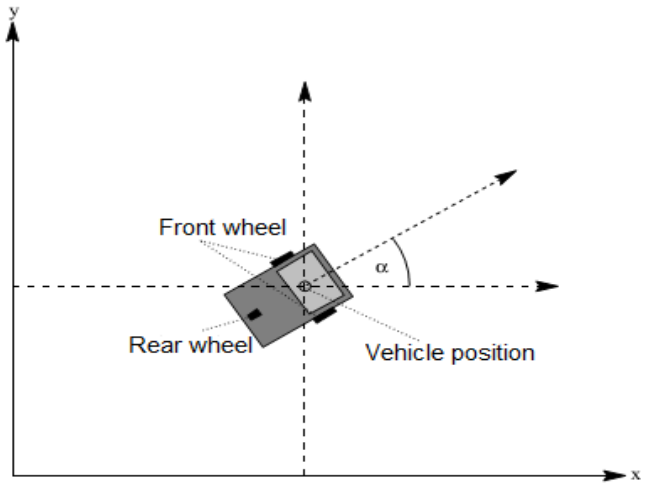


Fig. 9. Tricycle kinematics as used for example on the Pioneer 3DX Robot

Usually, this kinematic is chosen the reference point in the center of the axis of the wheels with odometry sensors. This achieves a simple calculation of the change in position of the distance which has been completed by the two wheels. The traveled distance is calculated as the average of the distances of the two wheels and the change in orientation is proportional to the difference of the two values. The following describes how the current position can be calculated from this information.

When vehicle is moving, at regular interval, the vehicle position is updated. For this purpose, the traveled distance δ and the change in the orientation α since the last time is measured and set off against from the current vehicle position. For simplicity, it is assumed that the vehicle is moving during this period almost rectilinear. Although this gives some error, this can be arbitrarily reduced by choosing a smaller time interval between measurements.

The robot position $l = (x, y, \theta)^T$ is updated by adding $a = (\delta, \alpha)^T$ using the following formula:

$$l \leftarrow F(l,a) = \begin{pmatrix} x + \delta\cos(\theta) \\ y + \delta\sin(\theta) \\ \theta + \alpha \end{pmatrix} \quad (3.1)$$

One problem of the location determination is that errors occur in the odometry. For example, the wheels slide (wheel spin), be out of round, it can be wrong due to uneven ground distances can be measured or running on the ground, does not allow accurate measurement of the distance traveled. These errors are typically quite low, for example, short distances provides dead reckoning very accurate results. Hence, the distance error is increasing without boundary limit.

Naturally, the occurring odometry error can be reduced. However, only the error will be reduced, but not the fundamental problem solved in that the position over long distances will be inaccurate. To solve this problem, further sensors may be included, for example distance measuring sensors, such as laser scanners. Nevertheless, the position error, which is generated in the dead reckoning, is modeled. The following will be discussed in more detail.

3.1.2 Position error

It is assumed that the errors of the position determination are normally distributed by dead reckoning, that is, measured distance and rotation are underlies Gaussian distribution. Furthermore, in a larger number of positioning systems (for example, the system based on Kalman filter) also modeled the robot position by a Gaussian distribution. This introduces the following representation.

$$l \sim N(\mu_l, \sum_l) \quad (3.2)$$

$$\mu_l = (\hat{x}, \hat{y}, \hat{\theta})^T \quad (3.3)$$

$$\sum_l = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 \end{pmatrix} \tag{3.4}$$

$$a \sim N(\mu_a, \sum_a) \tag{3.5}$$

$$\mu_a = (\hat{\delta}, \hat{\alpha})^T \tag{3.6}$$

$$\sum_a = \begin{pmatrix} \sigma_\delta^2 & 0 \\ 0 & \sigma_\alpha^2 \end{pmatrix} \tag{3.7}$$

At the input a, it is assumed that the distance traveled δ not correlated with the change in the orientation α, so $\sigma_{\delta\alpha} = 0$.

The parameters of the function F are considered as a vector $(x, y, \theta, \delta, \alpha)^T$, the formulas may be used directly, mean value and covariance matrix in order to determine the old position of the vehicle and the input of the new vehicle position.

3.2 Categories of localization methods

Generally speaking, localization methods fall into three basic categories: behavior-based approaches, landmark localization and dense sensor matching.

Behavior based approaches are based on the interaction of robot actions with the environment to navigate. For example, robot followed a right-hand rule to traverse an office environment and found its way back reversing the procedure. While behavior-based systems are very useful for certain tasks, their ability to localize a robot geometrically is limited since their navigation capability is implicit in their sensor and action history.

Landmark localization is based on the recognition of landmarks to keep the robot localized geometry. The landmark may be given a priori or learn from the robot system as it explore the environment [25]. While landmark

localization methods can achieve impressive results in geometric localization, they require either engineering the environment to provide a set of adequate landmarks or efficient recognition features to use as landmarks. In contrast, dense sensor data comparative methods [8, 21] attempt to use whatever the sensor information is available to determine the robot position. This is accomplished by matching the dense sensor scans against a surface map of the environment, without extracting landmark features. Hence, dense sensor data comparative method can take advantage of any existing features in the sensor data without having to explicitly decide what constitutes a landmark.

In the following, scan matching with Kalman filter which uses a Gaussian probability distribution is presented.

3.3 Scan matching localization

Scan matching localization is a process which has been successfully used to localize a mobile robot [15, 21, 26, 27]. Usually, Kalman filter is used to fuse with the pose estimate by audiometry and scan matching.

Scan matching localization using Kalman filtering represents the probability distribution of the robot position by a Gaussian distribution:

$$p(l) = N(\mu_l, \sum_l) \tag{3.8}$$

Likewise odometry errors and scan-matching correction can be modeled by Gaussian distributions. This has the advantage that robot positions can be calculated with high accuracy and an efficient fusion method can be used, namely, Kalman filtering.

On the robot moves $a = (\delta, \alpha)^T$, and then the prediction step, the new robot position is calculated using the formulas:

$$\mu_l \leftarrow F(l,a) = \begin{pmatrix} \hat{x} + \hat{\delta}\cos(\hat{\theta}) \\ \hat{y} + \hat{\delta}\sin(\hat{\theta}) \\ \hat{\theta} + \hat{\alpha} \end{pmatrix} \tag{3.9}$$

$$\sum_l \leftarrow \nabla F_l \sum_l \nabla F_l^T + \nabla F_a \sum_a \nabla F_a^T \tag{3.10}$$

From the scan matching a pose update μ_0 with an error covariance matrix Σ_0 is obtained and the robot pose and covariance is updated using the formulas:

$$\mu_l \leftarrow \left(\sum_l^{-1} + \sum_0^{-1}\right)^{-1} \left(\sum_l^{-1} \mu_l + \sum_0^{-1} \mu_0\right) \tag{3.11}$$

$$\sum_l \leftarrow \left(\sum_l^{-1} + \sum_0^{-1}\right)^{-1} \tag{3.12}$$

These equations show that Kalman filter based location can be efficiently implemented. As long as the error models are accurate, Kalman filter will provide a very good position estimate.

The success of the Kalman filter depends on the ability of scan matching. If scan matching provides a false position estimation, the Kalman filter also computed incorrectly, fused estimate. In the worst case, the robot cannot localize or provides incorrect position values.

4 PATH PLANNING AND COLLISION AVOIDANCE

4.1 Path planning

Path planning for mobile robots consists of finding a sequence of state transitions that leads robot from its initial state to some desired goal state. Typically, the states are robot locations and the transitions represent actions the robot can take, each of which has an associated cost.

Path planning in a dynamic and unknown environment is the most complicated case in robotic motion planning, and is also the most common situation that mobile robots will confront. In the real world, mobile robots often need not only to avoid static obstacles, but also to avoid colliding with the large obstacle lives. Due to the complicated and unknown environment, the robot cannot adopt one time global path planning for the environment. The global optimized is thereby difficult to be obtained. The robot has to use sensors acquiring the information about the surrounding environment and do online real-time path planning. The planning time for mobile robot should be short because the robot needs a sufficient time interval to adjust its movement in order to avoid the coming obstacle.

Given a particular goal, a robot must be able to generate a path that it will follow from its current position in the environment to the specified goal.

In the two previous sections have described how a robot can determine its position using a map of the environment, and how it can create this map from a data exploration movement. To obtain a complete navigation system in which the robot system operates autonomously in its environment, the system must also be able to path determination and motion planning. This is the subject of this section.

The robot trajectory or motion planning in the presence of moving obstacles are studied. The goal is to find an optimal robot trajectory (consisting of both path and the motion along the path) which avoids collision with moving obstacles. Some theatrical result about the complexity or trajectory planning can be found in [10,32]. Some heuristic approaches for planning a collision free path in the presence of moving obstacle are presented in [26, 29, 31].

The problem of path planning can be formulated as follows: a start position (the current position of the robot), a target position and a neighborhood are given. We look for a sequence of actions, for example a sequence of intermediate positions that the robot interference (also without colliding with obstacles) moving from the start to the target position. This problem has been studied extensively by Latombe [29]. There are many different approaches presented and complexity assessments determined.

An idea of abstracting the problem of motion planning lies in the concept of configuration space. The concept of configuration space has been widely used for solving the path planning problem. The configuration space consists of all possible positions (location and orientation), in which the robot may be in its vicinity. Through this model, the robot can be modeled as a point and the path planning is reduced to design of a point in configuration space.

For the movement in the plane of planning the configuration space is three dimensional, the robot is a cylinder and it can be rotated on the spot, that is, drive at any time in any direction, then the configuration space is reduced to 2 dimensions since the orientation of the robot is no longer important. In many processes, therefore, the robot is assumed to be circular or extended the shape of the robot to circle. In this work, this assumption can also be used to design a new grid space path planning method.

The rest of this section is organized as follows. In the former, different approaches to motion planning is presented as outlined in [29].

The second, we introduce the path planning algorithm focused on determining a path that fulfills a specified direction represented by using the focused D* search algorithm. The FD* approach divides the environment into cells of equal size is discussed in detail and show that it is suitable for large environments. The robot creates a topological route map by laser scans; it can be scheduled on the efficient paths over long distances. For local planning of individual intermediate points, a FD* approach is used with a limited search space. This results in an efficient and robust path planning system that avoids obstacle dynamic reactive recognizes impassable ways and find alternative routes. In this method, an experiment is implemented with a Pioneer 3DX robot in which the system has a number of times to plan a new path. Finally, the method is evaluated and discusses potential problems.

4.1.1 Approaches for path planning

There are a variety of procedures for planning the motion of a mobile robot. Most of these methods classify into the following categories:

Roadmap approach: the roadmap approach to path planning consists of capturing the connectivity of the robot's free space in a network of one-dimensional curves lying in the free space. Once a roadmap has been constructed, it is used as a set of standardized paths. Path planning is thus reduced to connecting the initial and goal configurations to the points and searching for a path between these points.

Potential field approach: the idea of this approach is to make the robot move as particles in an artificial potential field. In this case, the potential field is induced by the target position, which an attracting force, and the

obstacles produce a repulsive potential. The negated gradient of the total potential is treated as an artificial force applied to the robot. The position and direction vector $X = (x, y)^T$ of robot are fixed on by a composite of attractive force and repulsive force.

The attractive potential field function is given by:

$$U_{att}(X) = \frac{1}{2} k d^2(X, X_g) \tag{4.1}$$

Where:

k: positive scaling factor

X: position of the robot

X_g: goal of the robot

$d(X, X_g) = \|X_g - X\|$: Distance from robot to goal (4.2)

The attractive force F_{att} is negative grads of the attractive potential field function:

$$F_{att} = -\nabla[U_{att}(X)] = kd(X, X_g) \tag{4.3}$$

The repulsive potential field function is described by

$$U_{ref}(X) = \begin{cases} 0.5\eta\left(\dfrac{1}{d(X, X_0)} - \dfrac{1}{d}\right) & d(X, X_0) \le d_0 \\ 0 & d(X, X_0) > d_0 \end{cases} \tag{4.4}$$

η: positive scaling factor

$d(X, X_0)$: the shortest distance between the robot and obstacles

The term of constant d_0 is the distance of influence imposed by the obstacle; its value depends on the condition of the obstacle and the goal of the robot, and it usually less than half distances between the obstacles or shortest length from the destination to the obstacles.

When the robot is not at the goal, the repulsive force is

$$F_{ref}(X) = -\nabla[U_{ref}(X)]$$

$$= \begin{cases} \eta\left(\frac{1}{d(X,X_0)} - \frac{1}{d}\right)\frac{1}{d(X,X_0)} & d(X,X_0) \le d_0 \\ 0 & d(X,X_0) > d_0 \end{cases} \quad (4.5)$$

The resultant force is

$$F = F_{att} + F_{ref} \quad (4.6)$$

F navigates the movement of the robot as illustrated in Fig.10.

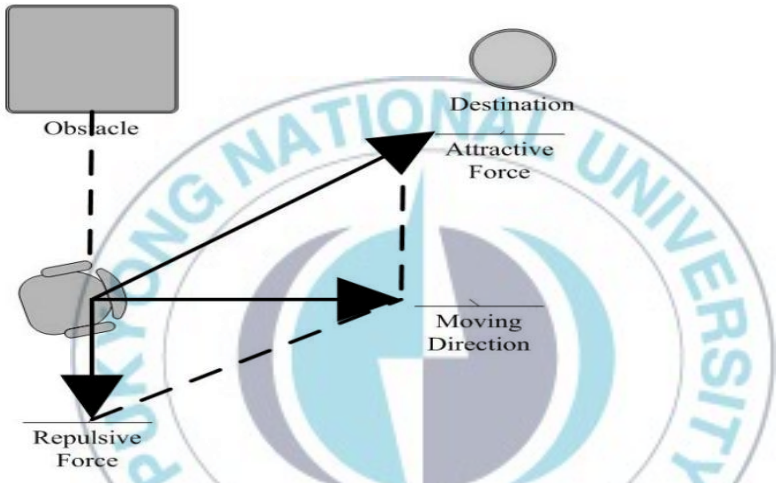


Fig. 10. Artificial Potential Field Method

The potential field methods can be very efficient to plan paths for a robot. However, it also has some drawbacks. The major problem is that robots are often trapped into a local minimum before reaching the destination. Therefore, this method is combined with many other computational methods to improve its efficiency.

A* algorithm finds a path as good as found by Dijkstra's algorithm but does it much more efficiently using an additional heuristic to guide itself to the goal. Dijkstra's algorithm uses a best first approach. It works by visiting modes in the graph starting from the start point and repeatedly examining the

closest not yet examined node until it reaches the goal. A* always first expands the node with the best cost calculated by $f(n) = g(n) + h(n)$. Where $g(n)$ represents the cost of the path from the starting point to the node n, and $h(n)$ represents the heuristic estimated cost from the node n to the goal. Usually, for calculating the heuristic cost, the Euclidean distance is used.

These methods have the serious disadvantage that the robot can operate in local minima, from which he can only get out by using additional mechanisms again. For this reason, this approach will not be further pursued.

In the following, a Focused Dynamics D* (FD*) is presented for motion planning and described problems of this approach due to its efficient use of heuristics and incremental updates.

4.1.2 FD * path planning algorithm

In this work, the FD* [32, 33] graph search algorithm is used to find the global path from the current position to the goal position. The FD* algorithm can handle increasing or decreasing arc costs and moving start states. This method uses the focusing heuristic function to estimate the estimated path cost from the current location to the goal to help the robot to minimize its search space. It plans optimal traverse in real-time by incrementally repairing paths to the robot's state as a new environment information is known, which makes it possible to greatly reduce the computational cost. When the robot gathers new information about the environment, it will re-plan new paths based on the new information and procedure a path for the robot.

4.1.2.1 Definitions and Formulation

The set of states denote robot location connected by directional arcs, each of which has an associated cost.

The term of G denotes the goal state from the robot starting at a particular state and moving across arcs (incurring the cost of traversal) to other states until reaching the goal state.

Every state X except G has a backpointer to a next state Y denoted by $b(X) = Y$. The D* use backpointer to represent the paths to the goal.

Arc cost function c (X, Y) is the cost of traversing an arc from state Y to state X. This cost is a positive number and the cost function c (X, Y) is undefined if Y does not have an arc to X. Therefore, two states X and Y are neighbors in the space if c (X, Y) is defined.

D* use an OPEN list to propagate information about changes to the arc cost function and to calculate the path cost to states in the space. Every state X has an associated tag t (X) such that:

$$t(X) = \begin{cases} New & \text{if X } \notin \text{ OPEN list} \\ OPEN & \text{if X } \in \text{ OPEN list} \\ CLOSED & \text{if X } \notin \text{ OPEN list} \end{cases}$$

For each visited state X, D* maintains an estimate of the sum of the arc cost from an X to G given by path cost function h (X). This estimate is equivalent to the optimal (minimal) cost from the state X to G.

The key function k (X) is defined to be equal to the minimum of h (X) before modification and all values assumed h (X). The key function classifies a state X on the OPEN list into one of two types: a RAISE state if $\mathrm{k(X)} < \mathrm{h(X)}$, and a LOWER state if $\mathrm{k(X)} = \mathrm{h(X)}$. D* use RAISE state to propagate information about path cost increases and LOWER state propagate information about path cost reductions. The propagation takes place through the repeated removal of the states from the OPEN list. Each time a state is removed from the list, it expanded to pass cost changes to its

neighbors. These neighbors are in turn placed on the OPEN list to continue the process.

Let $\{R_0, R_1, ..., R_N\}$ be the sequence of states occupied by the robot when states were added to OPEN list, where R_i is the robot's state at the time X was inserted on the OPEN list and a biased value $f_B(X, R_i)$.

The value of $f_B(X, R_i)$ is given by:

$$f_B(X, R_i) = f(X, R_i) + d(R_i, R_0) \tag{4.7}$$

Where:

The function $f(X, R_i)$ is the estimated robot path cost given by:

$$f(X, R_i) = h(X) + g(X, R_i) \tag{4.8}$$

The function $d(R_i, R_0)$ is the accrued bias given by:

$$d(R_i, R_0) = \begin{cases} g(R_1, R_0) + g(R_2, R_1) + ... + g(R_i, R_{i-1}) + i\varepsilon & \text{if i > 0} \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

The function $g(X, Y)$ is the focusing heuristic, representing the estimated path cost from Y to X. A vector of values (f_B, f, k) is stored with each state on the list.

The parameters $f_{\min}$ and k_{val} are defined to be minimum values for all X and be its corresponding k (X) value respectively. These parameters comprise an important threshold for D*. By processing properly-focused f values in ascending order, the algorithm ensures that for all states X, if $f(X) < f_{\min}$ or ($f(X) = f_{\min}$ and $h(X) \le k_{val}$) then $h(X)$ is optimal. Let R_{curr} be the current state on which the search is focused, initialized to the robot's start state. The parameter d_{curr} is the accrued bias from the robot's start to its current state.

$$d_{curr} = d(R_0, R_0) = 0 \quad (4.10)$$

4.1.2.1 Focus D* Algorithm

Like A*, D* operates on a cost graph. The environment with the obstacles is represented by a uniform grid map. The main idea of the method is illustrated as follows: From the initial state, the method repeatedly selects the neighbor with the minimum cost until propagates to the goal. Each small cell in the map is called a state. Each state X has the arc cost of the state X to the goal given by the path cost function $h(X)$. From the start point (start state), all neighbor states of the current state are listed on the open list. From the open list, the method calculates the arc cost of the states by $h(X)$. Then, select the state with the minimum $h(X)$, go to this state, and new neighbors are added to the open list. In the dynamic uncertain environment, when the robot detects new obstacles or the absence of expected obstacles, the cost values of the states in the area change. And the adjoining states are put on the open list for cost correction. Encountering unexpected obstacles, D* will set off a "raise" wave, a wave of increasing cost, through neighboring states. When this wave meets with the states that are able to lower path costs, these "lower" states are put on the open list to recalculate new optimal paths. When it detects the absence of an expected obstacle, the arc of the path passing through this "missing" obstacle is assigned a small cost, denoting an empty space, and the adjoining state is put on the open list as a lower state, setting off a "lower" wave, a wave of decreasing cost. D* is able to determine how far the raise and lower waves need to propagate while providing the optimal path for robot traverse continuously.

The basic D* method can use heuristic function to focus on the search in the direction of the robot and reduce the total number of the state (grid)

expansion. The focus D* method uses the focusing heuristic function to estimate the estimated path cost from the current location to the goal to help the robot to minimize its search space.

We consider the FD* algorithm which is based on a path cost function h, which represents the total cost from the current node of the search to the goal node, and a heuristic function g, which estimates but never overestimates the cheapest solution for achieving the current node from the start node in the (x,y) grid map search space. The total cost function $f = g + h$ determines the order of expanding nodes in state space. The Focus D* algorithm consists primarily of three functions: PROCESS-STATE, MODIFY-COST, and MOVE-ROBOT.

The PROCESS - STATE is used to compute optimal path costs to the goal.

MODIFY- COST is used to change the arc cost function and enter affect states on the OPEN list.

MOVE-ROBOT uses the two functions to move the robot optimally.

Initially, t (X) is set to NEW for all states, h (G) is set to zero, and G is placed on the OPEN list. The first function, PROCESS-STATE, is repeatedly called until the robot's state, X, is removed from the OPEN list. The robot then proceeds to follow the backpointers in the sequence {X} until it either reaches the goal or discovers an error in the arc cost function c. The second function, MODIFY-COST, is immediately called to correct c and place affected states on OPEN list. The robot's state is updated on it discovers an error, a possibly new sequence states has been constructed, and the robot continues to follow the backpointers in the sequence toward the goal. The last function, MOVE-ROBOT illustrates how to use Process-State and Modify-Cost to move the robot through the environment with the goal along the optimal traverse. The algorithms for these functions are presented below

along with three of more detailed functions for managing the OPEN list: INSERT, MIN-STATE, and MIN-VAL. The user provides the function GVAL(X,Y) which computes and returns the focusing heuristic g(X,Y). The embedded routines are:

MIN(a, b) returns minimum of two scalar values a and b.

COST(X) computes $f(X, R_{curr}) = h(X) + GVAL(X, R_{curr})$ and return the vector of values $(f(X, R_{curr}), h(X))$ for a state X.

DELETE(X), which deletes state X from the OPEN list and set $t(X) = CLOSED$.

PUT-STATE(X) inserts X on the OPEN list according to the vector $(f_B(X), f(X), k(X))$.

GET-STATE returns the state on the OPEN list with minimum vector value.

The INSERT function changes the value of $h(X)$ to h_{new} and inserts or repositions X on the OPEN list. The INSERT function is described as:

Function: INSERT(X,h_{new})

If t(X) = new then k(X) = h_{new}

else

If t(X) = OPEN then

$k(X) = MIN(k(X), h_{new})$;

DELETE(X);

else $k(X) = MIN(h(X), h_{new})$

$h(X) = h_{new}$; $r(X) = R_{curr}$;

$f(X) = h(X) + GVAL(X, R_{curr})$; $f_B(X) = f(X) + d_{curr}$

PUT-STATE(X)

The function MIN-STATE returns the state on the OPEN list with minimum f value. In order to do this, the function retrieves the state on the OPEN list with lower value. If the state was placed on the OPEN list when the robot was at a previous location, then it is re-inserted on the OPEN list. This operation has the effect of correcting the state's accrued bias using the robot's current state while leaving the state's h and k values unchanged. MIN-STATE continues to retrieves states from the OPEN list until it finds one that was placed on the OPEN list with the robot at its current state.

Function: MIN-STATE ()

While X = GET-STATE () = -1

if $r(X) \neq R_{curr}$ then

$h_{new} = h(X);\ h(X) = \mathrm{k}(\mathrm{X});$ DELETE(X); $INSERT(X, h_{new})$

else return X

return -1

The MIN-VAL function returns the f and k values of the state on the OPEN list with minimum f value, that is, $(f_{\min}, k_{val})$.

In the function PROCESS-STATE cost changes are propagated and new paths are computed. The state X with lowest f value is removed from OPEN list. If X is LOWER state (e.i., $k(X) = h(X)$), its path cost is optimal. Each neighbor state of X is examined to see if its path cost can be lowered. Additionally, the neighbor state receives an initial path cost value, and cost changes are propagated to each neighbor that has a backpointer X, regardless of whether the new cost is greater than or less than the old. Because these states are descendants of X, any change to the path cost of X affects their path costs as well. All neighbors that receive a new path cost are placed on the OPEN list, so that they will propagate the cost changes to their neighbor.

If X is a RAISE state, its path cost may not be optimal. Before the X propagate cost change to its neighbor, its optimal neighbors are examined to see if $h(X)$ can be reduced. cost changes are propagated to NEW state and immediate descendants in the same way as for LOWER states. If X is able to lower the path cost of a state that is not an immediate descendant, X is placed back on the OPEN list for future expansion. This action is required to avoid creating a closed loop in the backpointer. Thus, the update is postponed until the neighbor has an optimal path cost.

Function: PROCESS-STATE

X = MIN-STATE()

if X = NULL then return -1

$val = (f(X), k(X))$; $k(X) = k_{val}$; $DELETE(X)$;

if $k_{val} < h(X)$ then

If $t(Y) \neq NEW$ & $h(X) > h(Y) + c(Y, X)$ then

$b(X) = Y$; $h(X) = h(Y) + c(Y, X)$;

if $k_{val} = h(X)$ then

If $t(Y) \neq NEW$ || ($b(Y) = X$ & $h(Y) \neq h(X) + c(Y, X)$) ||($b(Y) \neq X$ & $h(Y) > h(X) + c(Y, X)$) then

$b(Y) = X$; $INSERT(Y, h(X) + c(Y, X))$;

else

if $t(Y) = NEW$ || ($b(Y) = X$ & $h(Y) \neq h(X) + c(Y, X)$) then

$b(Y) = X$; $INSERT(Y, h(X) + c(Y, X))$;

else

If $b(Y) \neq X$ & $h(Y) > h(X) + c(Y, X)$ & $t(Y) = CLOSED$ then

$INSERT(X, h(X))$;

else

if $b(Y) \neq X$ & $h(Y) > h(X) + c(Y, X)$ & $t(Y) = CLOSED$ & $h(Y) > k_{val}$ then

$INSERT(Y, h(Y))$;

return MIN-VAL()

In function MODIFY-COST, the arc cost function is updated with changed value. Since the path cost for state Y state will change. When X is expanded via PROCESS-STATE, it computes a new $h(Y) > h(X) + c(Y, X)$ and places Y on the OPEN list. Additional state expansions propagate the cost to the descendants of Y.

Function: MODIFY-COST (X,Y,c_{val})

$c(X, Y) = c_{val}$

If $t(Y) = CLOSED$ then $INSERT(X, h(X))$

Return MIN-VAL ()

The function MOVE-ROBOT illustrates how to use PROCESS-STATE and MODIFY COST to move the robot from S through the environment to G along an optimal transverse. For all states, t is set to NEW, h(G) is set to zero. PROCESS-STATE is called repeatedly until either an initial path is computed to the robot's state or it is determined that no path exists. The robot then proceeds to follow the backpointer until it either reaches the goal or discover a discrepancy between the sensor measurements of an arc cost s and stored arc cost c due to a detected obstacle. If the robot moved since the last time discrepancies were discovered, then its state R is saved as the new focal point and the accrued bias d_{curr} is updated. MODIFY-COST is called to correct c and place affected state on the OPEN list then called repeatedly to propagate costs and compute a new path to the goal. The robot continues to follow the

backpointers toward the goal. The function return GOAL-REACHED of the goal is found and NO-PATH if it is unreachable.

Function: MOVE-ROBOT(S,G)

$t(Y) = NEW$

$d_{curr} = 0;\ R_{curr} = S;$

$INSERT(G, 0);\ val = (0, 0)$

while $t(S) \neq CLOSED \,\&\, val \neq -1$

$val = PROCESS - STATE()$

if $t(S) = NEW$ then return NO-PATH

$R = S$

while $R \neq G$

if $s(Y, X) = c(Y, X)$ then

if $R_{curr} \neq R$

$d_{curr} = d_{curr} + GVAL(R, R_{curr}) + \varepsilon$

$R_{curr} = R$

$val = MODIFY - COST(X, Y, s(X, Y))$

while $val \neq No - PATH$

$val = PROCESS - STATE();\ R = b(R)$

return GOAL-REACHED

FD* search fans out from the goal node, expanding neighbor nodes within the contours of increasing f value until the start node is reached or all possible obstacle free neighbors are exhausted upon which the algorithm declares no path is found. Initial search by FD* algorithm sets pointer from each state in the searched area to the next state and optimal paths to the goal from every state in the expanded area of the environment are computed simply following the pointers. Any discrepancy that is discovered from the

earlier sensory information about the vicinity of the robot environment initiates algorithm on-line execution. The new path is then determined redirecting the pointers locally. The number of expanded nodes is minimal and consequently the time of execution.

4.2 Collision Avoidance Behavior

To operate successfully in populated environments, mobile robots must be able perceive their environment and react to unforeseen circumstances and re-plan dynamically in order to achieve their missions. To ensure safe navigations, most of existing robot systems rely on reactive collision avoidance modules to control the robot. The predominant hypothesis of these approaches to collision avoidance is strictly sensor-based: Sensor readings are continuously analyzed to determine collision-free motion.

In this section, we focus on the reactive avoidance of collisions with obstacles. The Dynamic Window Approach [31] proposed in this work is especially designed to deal with the constraints improved by limited velocities and accelerations, because it is derived directly from the motion dynamics of mobile robots. The DWA generates actuator command such that the robot does not collide with obstacles because this method considers periodically only a short time interval when computing the next steering command to avoid the enormous complexity of the general motion planning approach. The approximation of trajectories during such a time interval by circular arcs results in a 2D search space of a translational and rotational space. This search space is reduced to the admissible velocities allowing the robot to stop safely. In various tests, the DWA figured out to be reliable even in highly crowded areas. The advantage of the DWA over other low-level colliding avoidance algorithms is the low complexity even at high speeds.

4.2.1 Modeling of Wheeled Mobile Robot (WMR)

The kinematical scheme of a mobile robot, where v is the velocity of the robot's centroid. v_L is the velocity of the left wheel, v_R is the velocity of the right wheel, b is the bias of the WMR (distance between the planes of the drive wheels), r is the radius of the drive wheels, (x,y) is the position of the robot, and the orientation of the robot is shown in Fig. 11.

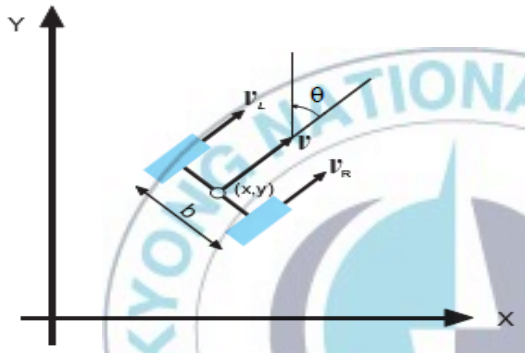


Fig. 11. Kinematic variables of the WMR

According to the motion principle of the rigid body kinematics, the motion of a mobile robot can be described using equations (4.2.1.1) and (4.2.1.2), where ω_L and ω_R are angular velocities of the left and right wheels respectively, and ω is the angular velocity of the centroid as (4.2.1.3).

$$v_R = r\omega_R \quad , \; v_L = r\omega_L \tag{4.1}$$

$$\omega = \frac{v_R - v_L}{b} \quad , \; v = \frac{v_R + v_L}{2} \tag{4.12}$$

Substituting equation (4.2.1.2) into (4.2.1.2) which yields

$$\omega = \frac{r}{b}(\omega_R - \omega_L) \quad , \; v = \frac{r}{2}(\omega_R + \omega_L) \tag{4.13}$$

Moreover, the dynamic function of the robots gives

$$\dot{x} = v\cos\theta \quad , \dot{y} = v\sin\theta \quad , \dot{\theta} = \omega \tag{4.14}$$

Substituting equation (4.2.1.4) in (4.2.1.3) we obtained:

$$\omega_R = \frac{1}{b}v + \frac{b}{2r}\omega \quad , \omega_L = \frac{1}{b}v - \frac{b}{2r}\omega \tag{4.15}$$

The kinematic equation and its inverse are given:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2}(\omega_R + \omega_L) \\ \frac{r}{b}(\omega_R - \omega_L) \end{bmatrix} \tag{4.16}$$

$$\begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} = \begin{bmatrix} \frac{1}{b}v + \frac{b}{2r}\omega \\ \frac{1}{b}v - \frac{b}{2r}\omega \end{bmatrix} \tag{4.17}$$

4.2.2 General Motion equations

This section describes the fundamental motion of a mobile robot. The derivation begins with the correct dynamic laws, assuming that the robots translational and rotational velocity can be controlled independently. To make the equations more practical, we derive an approximation that models velocity as a piecewise constant function in time.

The use of a global reference frame allows the decoupling of the two translational axes. The equation of motion of the x- axis can be expressed as follows:

$$\begin{aligned} x(t_i) &= x(0) + v_x t_i + \int_0^{t_i} a_x t dt \\ &= x(0) + v_x t_i + \frac{1}{2} a_x t_i^2 \end{aligned} \tag{4.18}$$

Where the term of a_x and v_x are constants acceleration and velocity respectively.

Similarly for y-axis:

$$\begin{aligned} y(t_i) &= y(0) + v_y t_i + \int_0^{t_i} a_y t dt \\ &= y(0) + v_y t_i + \frac{1}{2} a_y t_i^2 \end{aligned} \tag{4.19}$$

These equations show that when accelerating from a constant velocity to achieve a given velocity command the robot describes a quadratic curve until the desired velocity is attained. The curvature of those curves depends on the magnitude of the acceleration. In order to achieve curves with low curvature the two-dimensional search space is searched for different accelerations. If the accelerations are chosen such that $a_{xy} = a_x = a_y$, a motion command will be defined as:

$$\vec{v} = (v_x, v_y) \tag{4.20}$$

$$\vec{a} = (a_x, a_y) \tag{4.21}$$

To determine if a motion command is admissible the length of the resulting trajectory has to be determined. Simulation of the base motion according to the equations of motion will determine the duration time t_i of the trajectory until hitting an obstacle. The length of the trajectory can then be computed analytically:

$$\begin{aligned} l &= \int_0^{t_i} v_{t_i} dt \\ &= \int_0^{t_i} \sqrt{(v_x + a_x t)^2 + (v_y + a_y t)^2} dt \end{aligned} \tag{4.22}$$

If the length of the trajectory permits the robot come to a halt after moving for the duration of one servo tick, the motion command is considered admissible.

4.2.3 Dynamic Window Approaches

The dynamic window approach is a velocity space local reactive avoidance technique where search for commands controlling the robot is carried out directly in the space of velocities. The dynamics window approach desire to collision avoidance need to following properties:

- ❖ The robot must travel safely even with high speed. It therefore must take the dynamic constraints into account.
- ❖ The robot should react adequately and rapidly to unforeseen circumstances. This requires fast techniques for the detection of obstacles and the selection of appropriate steering commands.
- ❖ The robot should make maximum progress towards the goal. This implies that whenever advantageous, the robot should modify its travel direction to stay away from obstacles.

This method takes into account the kinematic and dynamic constraints of the robot. The search space is the set of current velocity vector (v_c, ω_c), where v_c and ω_c denote the translation velocity and rotational velocity respectively that can be reached within the next sampling interval. Among all velocity tuples those are selected that allow the robot to come to a stop before hitting an obstacle, given the current position, the current velocity, and the acceleration capabilities of the robot. These velocities are called the admissible velocities. To determine the next motion command all admissible velocities within the dynamic window are considered. Among those a velocity is chosen that maximizes the alignment of the robot with the target and the length of the trajectory until an obstacle is reached. Then the dynamic window V_d is defined as [31]:

$$V_d = \left\{ \begin{array}{l} (v,\omega) \mid v \in [v_a - \dot{v}_c \Delta t, v_a + \dot{v}_c \Delta t] \wedge \\ \qquad \omega \in [\omega_a - \dot{\omega}_c \Delta t, \omega_a + \dot{\omega}_c \Delta t] \end{array} \right\} \qquad (4.23)$$

where the term of $\dot{v}_a$ and $\dot{\omega}_a$ represent maximal translational and rotational accelerations of actual.

A velocity vector (v, ω) is considered safe if the robot is able to stop along the trajectory defined by (v, ω) before crashing into any object that may be encountered along that path. The set V_a of admissible velocities can be obtained as:

$$V_a = \left\{ v, \omega \le \sqrt{2 d_{\min}(v,\omega) \dot{v}_b} \wedge \sqrt{2 d_{\min}(v,\omega) \dot{v}_b} \right\} \qquad (4.24)$$

Where

The term $d_{\min}(v, \omega)$ denotes the distance to the nearest obstacle on the corresponding curvature.

The term of $\dot{v}_b$ and $\dot{\omega}_b$ are maximal translational and rotational accelerations breakage decelerations.

Let V_s be the limitation of maximum translation and rotation velocity in the search space, then the result search space V_r can be described as the intersection of the restricted areas:

$$V_r = V_s \cap V_a \cap V_d \qquad (4.25)$$

In order to produce the search for velocities feasible and appropriate for fast reactive response, the dynamic window approach considers exclusively the first sampling interval to the collect the optimal velocity vector. The velocity in the remaining n-1 sampling intervals is constant. The search is repeated after each sampling interval and the velocities stay automatically constant if does not exit commands are given.

The velocity maximizing a certain objective function $F(v,\omega)$ is selected from the reduce set of velocities V_r as [31]. The object function includes a measure of progress towards a goal location, the forward velocity of the robot, and the distance to the next obstacle on the trajectory. It is expressed as weighted sum of the criteria target heading, clearance and velocity.

$$F(v,\omega) = k_1 head(v,\omega) + k_2 dist(v,\omega) + k_3 vel(v,\omega) \tag{4.26}$$

where:

The target heading $head(v,\omega)$ measure the alignment of the robot with the target direction. The function $head(v,\omega) = 1 - |\theta| / \pi$ where θ is the angle between the direction of motion and goal heading, result in large values for good alignment with the goal heading. The goal heading is modified if the robot's lateral distance to an obstacle becomes too small.

The function $dist(v,\omega)$ is the distance to the closest obstacle. If there is no obstacle existing on the curvature, this value is set to a large constant.

The function $vel(v,\omega)$ is used to evaluate the progress of the robot on the corresponding trajectory. It can be defined as follows:

$$vel(v,\omega) = \begin{cases} \dfrac{\|(v,\omega)\|}{\max(v,\omega)} & \text{if robot is far from goal} \\ 1 - \dfrac{\|(v,\omega)\|}{\max(v,\omega)} & \text{if robot is close to goal} \end{cases} \tag{4.27}$$

where $\max(v,\omega)$ the maximum velocity of the robot can be achieved. It will favor high velocities if the robot is far from the goal and low velocities when it is close. If the trajectory that results from the motion command passes through the goal region. The parameters k_1, k_2, and k_3 can be adjust to modify the behavior of the robot.

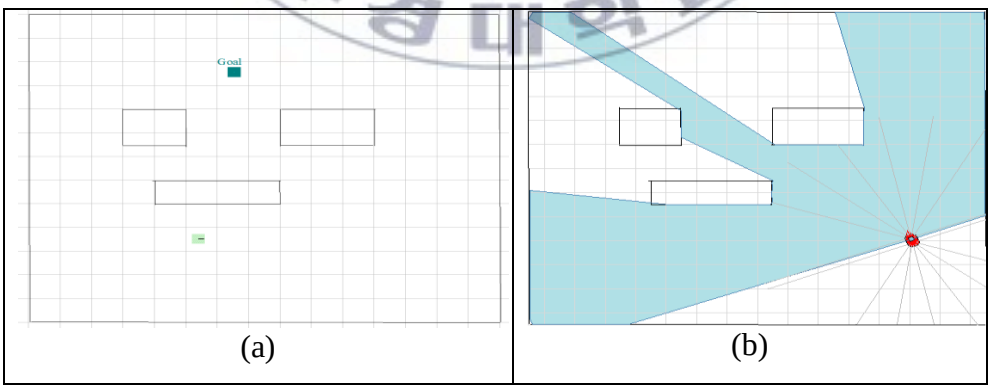
5 EXPERIMENTAL RESULTS

5.1 Simulation results

The mapping algorithm outline in the second section has been tested in various environments. The results obtained in this section come from the mapping algorithm running on simulated world data.

Fig.12 and Fig. 13 show the experiments have been carried out by running robot around the virtual environment based on MobileSim platform [44].

The proposed motion control algorithms have been implemented in Aria programming [31] and tested on the virtual robot is equipped with SICK that are used to detect dynamic obstacles and to update occupancy grid map information. Obstacles considered by the dynamic window were represented as point object. Partially known information about the state of the environment is given a prior. The simulation results were taken and monitor under MobileEyes software [44]. We tested a large number of various situations and our motion control methods provided safe and efficient robot motion in all of them. The results of a test are presented in Fig. 14



(a) (b)

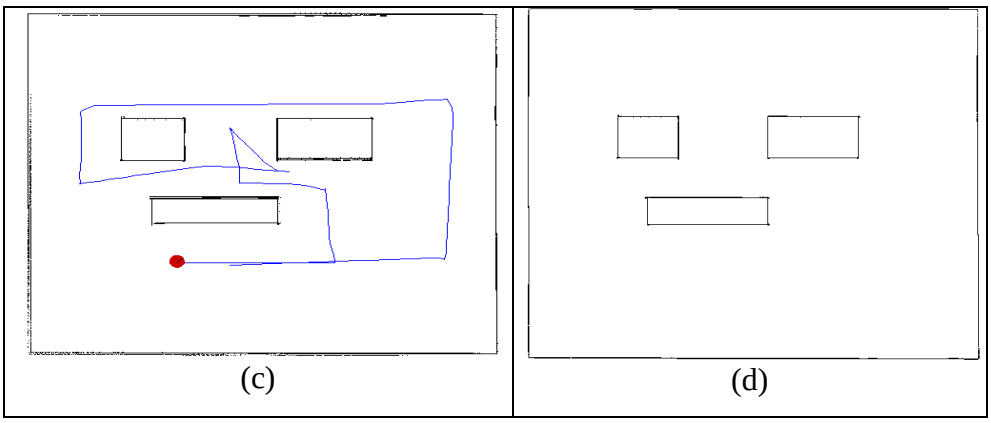


Fig. 12. Mapping simulation (a) Simulation environment constructed with a number of objects, (b) Start scanning, (c) Final map.

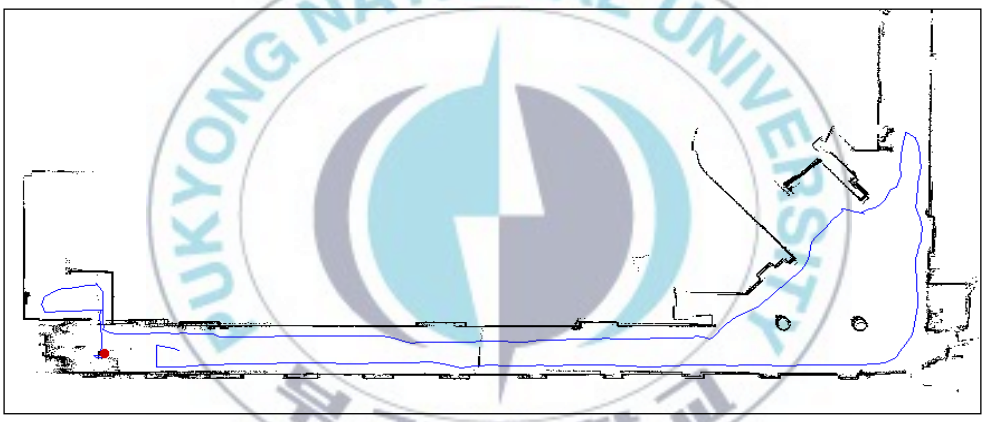
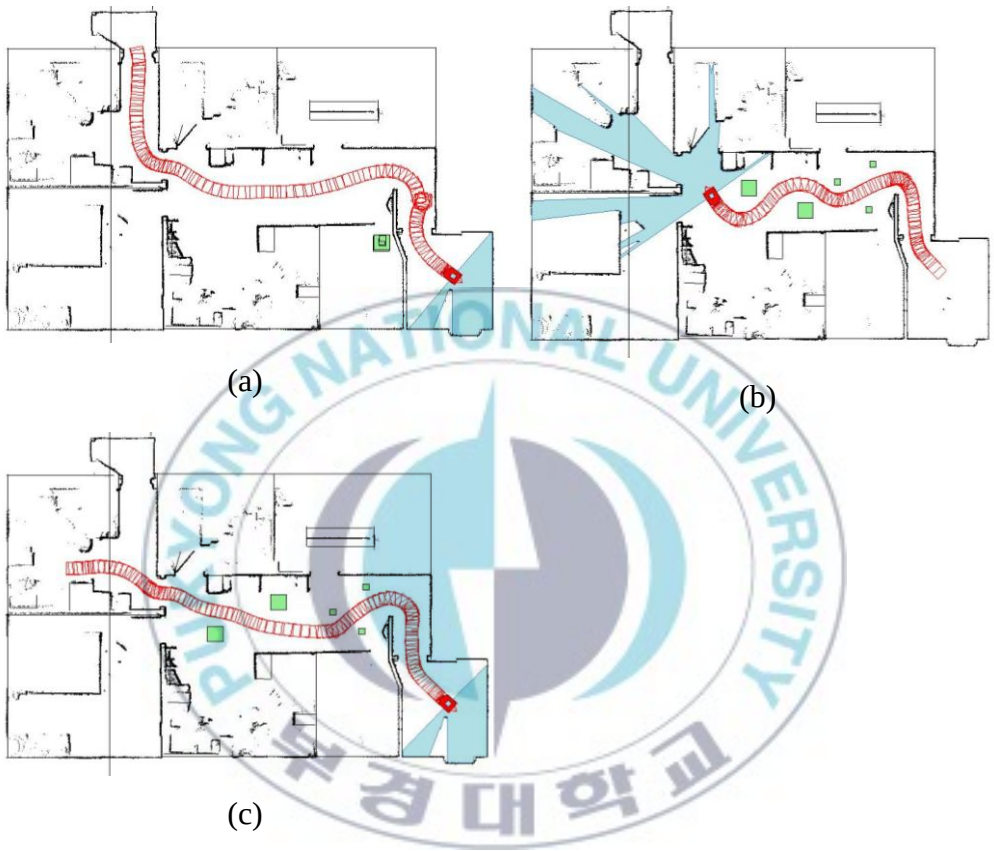


Fig. 13. Sketch of the larger environment including the trajectories of the robot. Lines of the walls correspond to the on-line detected obstacles which are incrementally incorporated into the grid. The smooth curvature represents the true robot trajectory. The robot was able to follow the planned optimal path well.

In order to illustrate the functionality of the proposed path planning and obstacle avoidance algorithms, the results of the test are presented in Fig. 14 where the robot moved across its path to the goal position. In the simulation

environments, we put a number of objects on the map. These objects are used to test the colliding avoidance method. The robot motion obtained by the presence of obstacles. Fig .14 shows the shortest global path in complex environments.



(a) (b)

(c)

Fig. 14. Motion control. (a) Path planning without obstacles, (b) and (c) Robot traveled from the start position until reaching the goal and avoid colliding with unexpected obstacles.

5.2 Experimental results

The map building algorithm outlined in the section 2 has been implemented and tested using real robots and datasets gathered with real

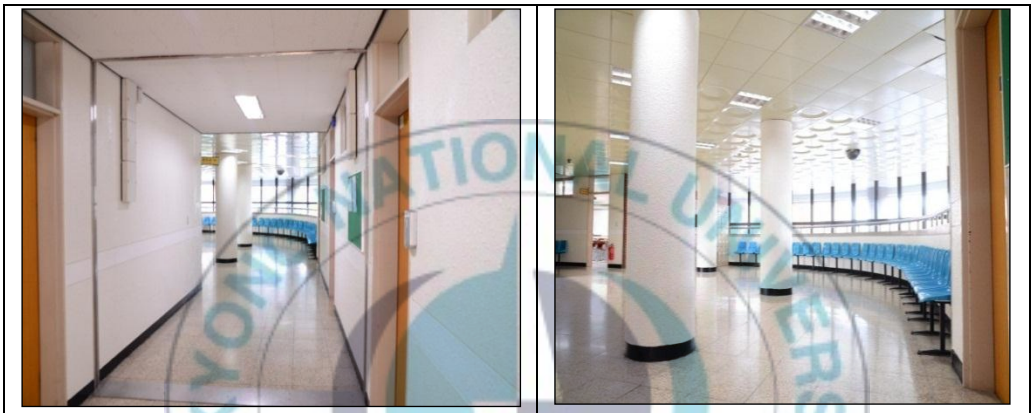
robot. Our implementation runs online on the ActivMedia Pioneer 3DX platform. The Pioneer 3DX robot is equipped with a SICK laser range finder mounted on the robot platform. The laser sensor has a maximum viewing angle of 180 degrees and an accuracy of the range-finder is 1 centimeter. The experiments carried out in a variety of environments have shown the effectiveness of our approach in indoor and outdoor environments.

In the first experiment, we use the Pioneer 3DX robot to build the map in the Intelligent Control Lab (ICLab) (as depicted in Fig.15) and in a populated corridor at Department of Mechatronics Engineering, Pukyong National University. The maps of these environments are depicted in Fig.16 and Fig.17. The datasets have been recorded with a Pioneer 3DX robot equipped with a SICK sensor. As can be seen in the right image of Fig. 17, the quality of the final map is so high without any significant errors or inconsistencies.

Fig. 15. The Intelligent Control Lab: The robot start in the lower left side and runs around the loop. The right image depicts the resulting map generated.

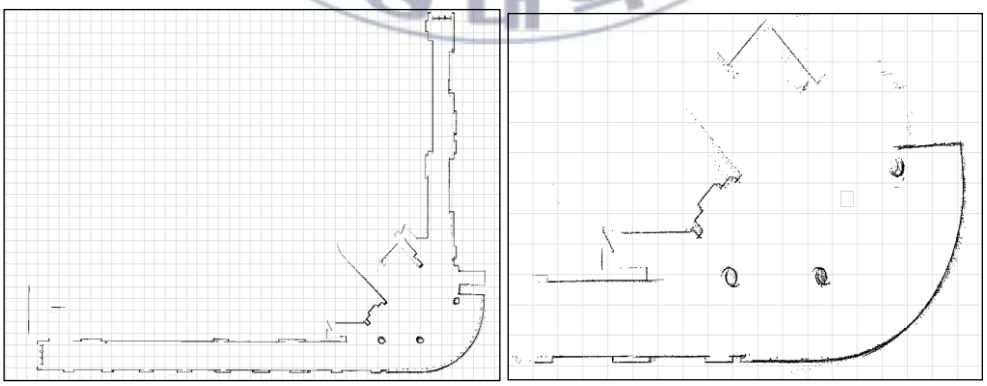
In the second experiment, we created a 2-D graphic map of the outdoors on Yongdang campus. From a bird's eyes view, walls, trees and obstruction of

the environment to be navigated as illustrated in Fig. 19 while the real world environment is delineated in Fig. 18. Note that this environment partly violates the assumptions that the environment is planar. Additionally, there were objects like bushes and grass which are hard to be mapped with a laser range finder. Furthermore, there were moving objects like cars and people. Despite the resulting spurious measurements, our algorithm was able to generate an accurate map.



(a) (b)

Fig. 16. Corridors on the second floor of Department of Mechatronics at Pukyong National University. (a) The corridor. (b) Detail of the corridor



(a) (b)

Fig. 17. 2-D environment map of a corridor in Mechatronics Building. (a) The whole environment. (b) A detail showing different artifacts

(a) (b)

Fig. 18. Outdoor area near the Mechatronics Department building

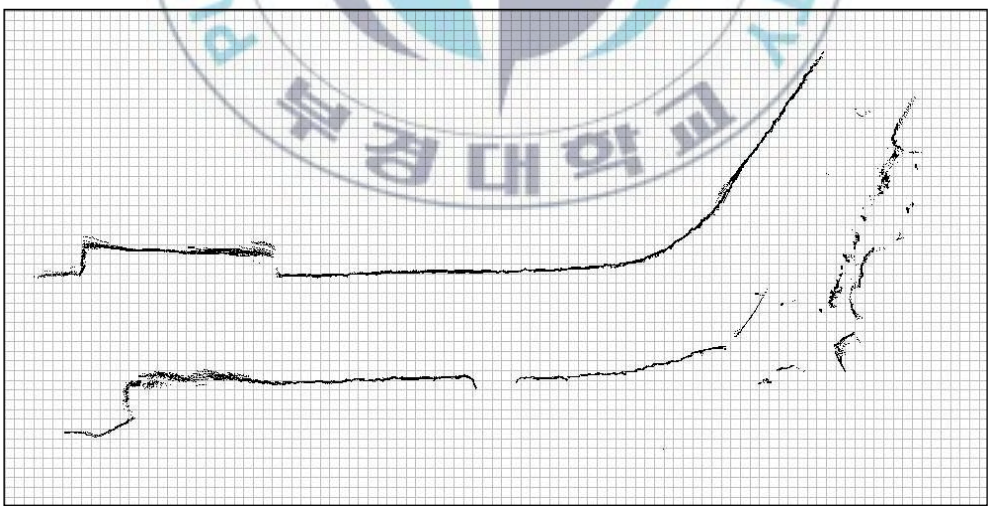


Fig. 19. Map of the path in the outdoor environment.

In the third experiment, we reconstructed the map of a partial Yondang

Campus. The test environment was implemented corresponds to an semi-outdoor area around several building such as the Chemical Engineering building (B4), school bus stop, Mechanical Engineering building (B6) and Mechatronics Department building (B9). The term semi-outdoor is used to define an outdoor area which is close to a building and, thus, has both structured and unstructured elements. Table 1 provides some general details of the collected data. The environment consists of narrow, tiled, bushes and the wall sides of building. Fig. 120. Shows the laser readings gathered in environment.

Number of readings	72424
Number of lines	181
Average speed	25 cm/s

Table 1. Some details of the data gathered in the semi-outdoor environment

Fig. 20. Map of partial Yongdang Campus reconstructed by laser reading gathered in environment.

The path planning method has been implemented and extensively tested on our Pioneer 3DX mobile robot. This robot is equipped with SICK laser range finder that is used to detect dynamic obstacles.

Fig. 21. Pioneer 3DX robot driving around an expected object.

The first experiment was carried out using the robot in our department environment at Pukyong National University. In order to test the capacities of our system to deal with unexpected obstacles we installed a number of objects in the corridor and change their position frequently. Additionally, people were walking in the environment. In the experiment, we did not observe a single collision during which the robot traveled over 40 m with average speeds of over 25 cm/s. Fig.21 show the typical situation during these experiments. Here the Pioneer 3DX robot is moving around unexpected obstacle in the corridor. During the experiment we found that the generated

paths were very smooth and that the overall behavior was quite efficient.

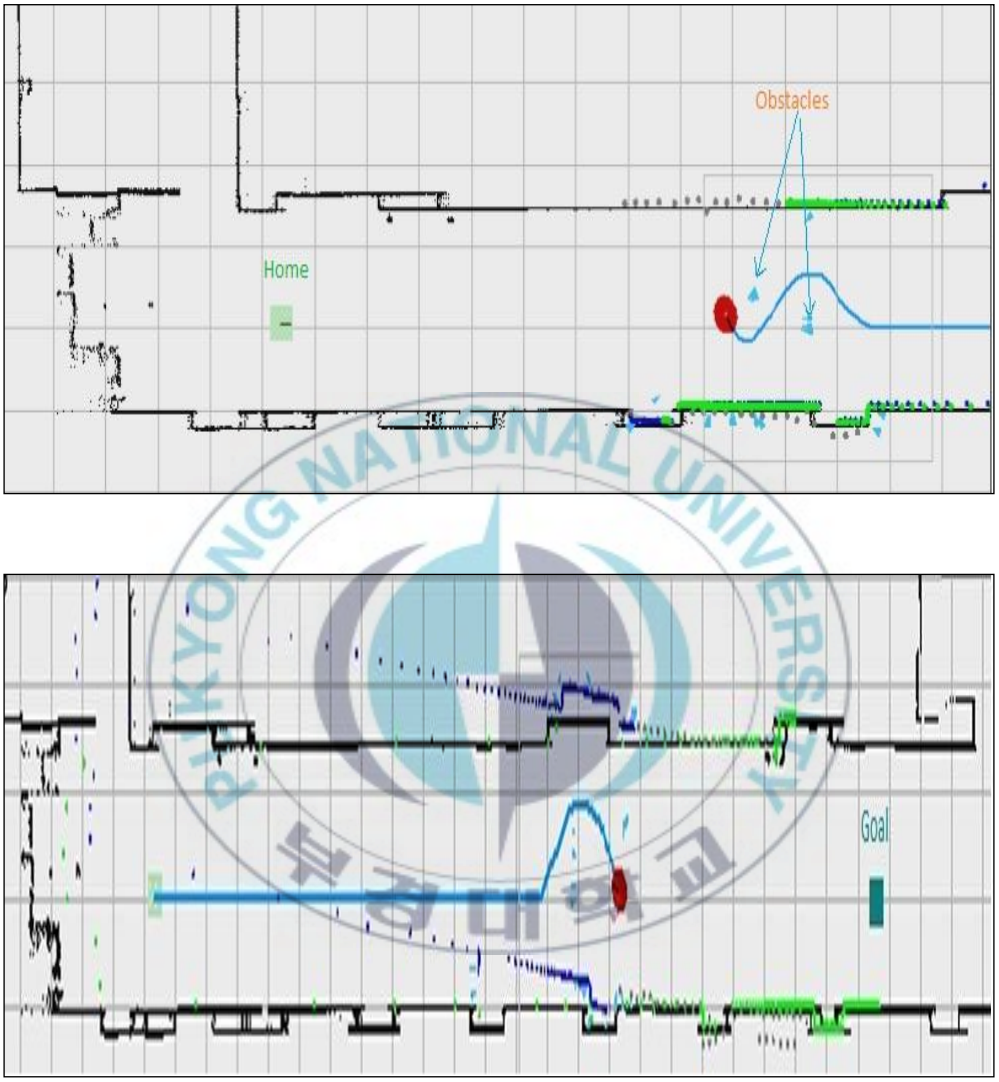


Fig. 22. Typical trajectory taken by Pioneer 3DX when reaching the target position.

The robot travel along the corridor of our department environment and it must collide avoidance is illustrated in Fig. 22. The image contains the trajectory generated and unexpected obstacle position.

Fig. 23. Experimental run in the campus at Pukyong National University

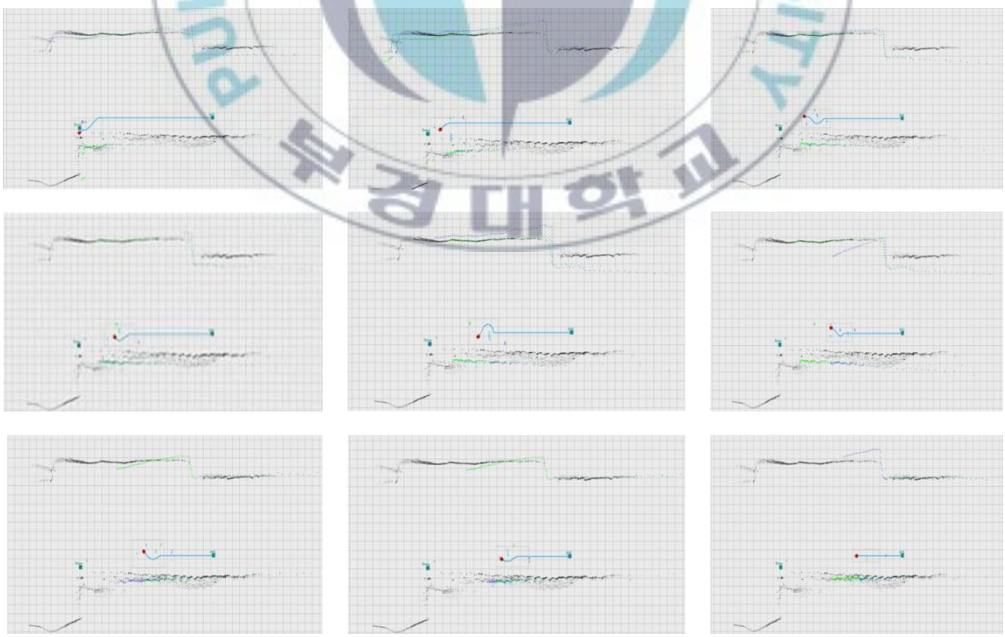


Fig. 24 Trajectory of Pioneer 3DX travelling along the path shown in Fig. 22

Another experiment carried out in an outdoor environment is depicted in Fig.23. Here we installed a large number of obstacles in front of a road to increase the difficulty of reaching the goal position. As can be seen in the image, the robot found the shortest path to reach the goal while colliding object located on the path. Fig. 24 shows a sequence of images illustrating the robot executing the steering commands.

6 CONCLUSIONS

This section summarizes the Thesis, and discusses the research limitations of the project. Future research on navigation for mobile robot will also be discussed.

6.1 Summary

We studied the problem of robot navigation in unknown and dynamic environment using a laser range finder sensor. This thesis presented the implementation of different types of high accuracy map building and navigation for both outdoor and indoor applications. We performed the arbitrary probability distributions across a grid of robot poses approach as a robust scan matching localization technique to estimate the position of a mobile robot. The scan matching approaches do not reply on the distinguishable features in the environment. Hence, we avoid the difficult process of feature detection and feature correspondences. We use most of the sensor data in the matching process. This suggests that scan matching methods are robust.

The map is generated by matching a scanned directly to another scan

without requiring an a priori world environment. Thus, these methods can be used for exploration and map building in unknown environments. In additionally; this method can work in general curved environments.

The results of localization by using scan matching localization approach shows that it potentially can keep track of the robot's position in an arbitrary probabilistic configuration.

We presented an integrated approach to sensor based collision avoidance and path planning for mobile robots in dynamic environments. Our algorithm includes several techniques to deal with the complexity of the induced problem during re-planning. The motion of an obstacle is regarded as the motion of the occupied moving cells in a grid map. The predicted trajectory of each moving cell is used for the collision calculation with the possible robot trajectories. The path planning is proposed based on the FD* graph search algorithm. The algorithm for producing the path is proved to be the shortest path in the geometry space. The possible robot trajectories are generated by using the dynamic window approach. The overall system is highly efficient and can be run on a standard PC. It automatically adapts itself to the performance of the underlying processor and to the complexity of the search problem.

The algorithm has been experimentally implemented and tested on a Pioneer 3DX mobile robot using a laser range finder. In all experiments our method was able to generate safe trajectories. The experiments confirmed that our algorithm yields efficient trajectories of the navigation system.

6.2 Research limitations

The project successfully implements the experiment mapping, localization and path planning for mobile robots in dynamic environments

based on several methods as discussed in the previous sections. However, there are still some research limitations in the project. The limitations of the project are described as follows:

We comprehensively studied the techniques for matching a range scan for deriving the relative position and heading of the robot. The difficulties are issues in this problem are that the scans are noisy, discontinuous, not necessarily linear, and two scans taken at different positions may not completed overlap because of occlusion.

In the study of optimal registration of multiple range scans for mapping an unknown environment, new sensor data are merged to a cumulative model based on local registrations, and this may cause inconsistency in the model.

6.3 Direction for future research

To conclude our thesis, we point out several possible directions for future research.

We have proven that it is sufficient to use a two-dimensional laser range finder for robot localization in structured indoor environments where the world contour is typically formed by long smooth walls. However, it will be much more difficult to do this in a cluttered industrial workshop where the nice wall structures are hidden by irregular, three-dimensional objects such as pipes, racks, machines, moving objects such as people, cars, etc. it will be interesting to examine if the laser range sensing is still capable of providing localization for the robot. A possible approach is to use 3D range scans to reveal more structures in the environment. Registration of 3D scans is considerably more difficult than registration of 2D scans because the range measurements are spares in 3D space. One possibility is to use an active approach in collecting the measurements (to point the laser beam in the directions where objects are observed from previous scans).

More robust localization algorithms should be investigated. Global Positioning System (GPS) can serve to improve localization; however, GPS currently is only precise to within several feet and is less precise indoors than outdoors. For tasks that require more precision and accuracy, GPS cannot be the sole solution. Software-based solutions such as Monte Carlo Localization are also promising. Sonar may be used in conjunction with other sensing modalities such as vision and laser to create more robust collision avoidance systems in more unstructured environments.

Furthermore, building maps from sensor data is typically done only for a limited period of time. After the robot has acquired a map, it uses this model for a variety of different tasks. An interesting aspect in the context of map learning is the lifelong map learning problem where the robot has to update and maintain its model of the environment for a long period of time. The longer the robot integrates observations obtained in the environment into grid map, the more the map gets blurred. The reasons for this are small errors in the observations, ambiguous situation for the scan-matcher, as well as the sampling process for drawing the next generation of samples. One possibility to overcome this problem is to abort the map update process and focus on localizing the vehicle. Whenever the robot detects changes in the environment it would have to updating the map model appropriately.

Though the integrated Focused D* and DWA solves the issue of obtaining the optimal path and colliding avoidance in a dynamic environment. The FD* method uses the focusing heuristic function to predict the estimated path cost from the current location to the goal to help the robot to minimize its search space. To enhance the performance of FD* algorithm, Field D* [43] method was proposed. Field D* extends the standards D* algorithm by using linear interpolation to derive the path cost of the points between grid intersections.

APPENDIX

A1. Notation

The following notations are used in this work.

The position of the robot is given by a row vector $(x, y, \theta)^T$ which contains location and orientation. For a matrix M and M^T specifies the transposed matrix. M is square and regular, then M^{-1} denotes the inverse of matrix M and det(M) its determinant.

For an angle α defined $R_2(\alpha)$ and $R_3(\alpha)$ associated rotation matrices

$$R_2(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$$

$$R_3(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A2. Normal Distribution

In the range of the sensor data processing is often assumed that the probability of occurrence is based on measured values on a normal distribution. One advantage of the normal distribution is that you can define them completely by specifying the expected value (mean value) and variance.

The density function of the normal distribution for the scalar case with mean μ_x and variance σ_x^2 is defined as follows:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-\frac{1}{2}\left(\frac{x-\mu_x}{\sigma_x}\right)^2}$$

Notation: $x \sim N(\mu_x, \sigma_x^2)$

In n-dimensional space, the above equation can be rewritten as:

$$p(x) = \frac{1}{\sqrt{(2\pi)^2 \det(\Sigma_x)}} e^{-\frac{1}{2}(x-\mu_x)^T \Sigma_x^{-1} (x-\mu_x)}$$

Notation: $x \sim N(\mu_x, \sum_x)$, where μ_x in a n-dimensional is expectation and $\sum_x$ is a n × n-dimensional covariance matrix.

A3. Mahalanobis distance

For the presentation of normally distributed density functions in the multidimensional case point sets with equal probability density are interesting. To determine this, set the density function p(x) is constant and determined x

$$\frac{1}{\sqrt{(2\pi)^2 \det(\sum_x)}} e^{-\frac{1}{2}(x-\mu_x)^T \sum_x^{-1}(x-\mu_x)} = \text{constant}$$

$$(x-\mu_x)^T \sum_x^{-1}(x-\mu_x) = \text{constant} = r^2$$

In the two-dimensional case this equation describes a contour line of the three-dimensional graph of p (x). This contour has the shape of an ellipse, the value of the formula $(x-\mu_x)^T \sum_x^{-1}(x-\mu_x)$ is the squared Mahalanobis distance between the vector x and the mean value μ_x. The importance of this measure can be illustrated most simply in the scalar case with r = 1. The equation then reduces to $(x-\mu_x)^2 \sigma_x^{-2} = 1 \text{or} (x-\mu_x)^2 = \sigma_x^2$. This means that the Mahalanobis distance of x to the mean value μ_x is closer to 1.

A4. Transformation of density functions

In the sensor data processing the problem is often given to transform observations of a sensor in another uniform observation system. There are the measured data from a normal distribution with a known mean and a known covariance matrix. The question now is what probability distribution is after the transformation.

Formally, a m-dimensional variable u, $u \sim N(\mu_u, \sum_u)$ and a transformation F: $\Re^m \to \Re^n$ with F (u) = x, $x \sim N(\mu_x, \sum_x)$ and given the mean value μ_x, and searched the covariance matrix $\sum_x$.

To determine the new normal distribution, a distinction between the two cases, if F is linear or nonlinear.

Linear Transformation

In the linear case, F can be represented as F (u) = Au + b, where A is a constant and b is an nxm matrix of n-dimensional column vector. For the expectation μ_x and covariance matrix $\sum_x$ is given by:

$$\begin{aligned}\mu_x &= E(x) \\ &= E(Au+b) \\ &= AE(u)+b \\ &= A\mu_u + b\end{aligned}$$

$$\begin{aligned}\sum_x &= E((x-E(x))(x=E(x))^T) \\ &= E((Au+b-AE((u)-b)(Au+b-AE(u)-b)^T) \\ &= E((A(u-E(u)))(A(u-E(u)))^T) \\ &= AE((u-E(u))(u-E(u)^T)A^T \\ &= A\sum_u A^T\end{aligned}$$

Nonlinear transformation

If F is non-linear, so approaching it through a Taylor polynomial in an appropriate reference point $\hat{u}$ and omits higher order terms:

$$F(u) \approx F(\hat{u}) + \nabla F(\hat{u})(u-\hat{u})$$

Here $\nabla F(\hat{u}) = \frac{\partial F}{\partial u}(\hat{u})$ is an nxn matrix with the partial derivatives of F at the point $\hat{u}$. This matrix is called the Jacobian.

F is linearized at a suitable reference point $\hat{u}$. For the reference point is referring to the mean value of u, because we are only interested in the points

near the expected value $\hat{u} = \mu_u$. Now the function F becomes to the linear case, and then it follows the matrix A and the vector b:

$$A = \nabla F(\mu_u)$$

$$b = F(\mu_u) - \nabla F(\mu_u)\mu_u$$

With the assistance of the results from linear case, the mean μ_x and covariance matrix μ_x is obtained:

$$\begin{aligned}\mu_x &= A\mu_u + b \\ &= \nabla F(\mu_u)\mu_u + F(\mu_u) - \nabla F(\mu_u)\mu_u \\ &= F(\mu_u)\end{aligned}$$

$$\begin{aligned}\sum_x &= A\sum_u A^T \\ &= \nabla F(\mu_u)\sum_u \nabla F(\mu_u)^T\end{aligned}$$

Example of transformation

Considering the measurement process of a 2d laser laser scanner, it is found that the range measurements and the angle at which the measurements take place are associated with an error. For this error, we assume that they are normally distributed and independently.

Let d is the distance measured and α is the angle at which the measurement takes place, then d and α normally distributed with $d \sim N(\mu_d, \sigma_d^2)$ and $\alpha \sim N(\mu_\alpha, \sigma_\alpha^2)$, where μ_d is the expected value (mean value) of the actual distance and μ_α is the mean value of the actual angle corresponds.

A measure $(d, \alpha)^T$ is now converted to Cartesian coordinates by applying the transformation F (see Fig…..)

$$F\begin{pmatrix} d \\ \alpha \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} d\cos\alpha \\ d\sin\alpha \end{pmatrix}$$

The transformation results for the Cartesian coordinates is determined by a new distribution

$$\begin{pmatrix} x \\ y \end{pmatrix} \sim N(\begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \Sigma_{xy})$$

where:

$$\begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} = F(\begin{pmatrix} d \\ \alpha \end{pmatrix}) = \begin{pmatrix} d\cos\alpha \\ d\sin\alpha \end{pmatrix}$$

$$\Sigma_{xy} = \nabla F_{d\alpha} \Sigma_{d\alpha} \nabla F_{d\alpha}^T$$

$$\nabla F_{d\alpha} = \begin{pmatrix} \cos\alpha & -d\sin\alpha \\ \sin\alpha & d\cos\alpha \end{pmatrix}$$

$$\Sigma_{d\alpha} = \begin{pmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_\alpha^2 \end{pmatrix}$$

Fig. 25**.** Transformation of a measurement in Cartesian coordinates

A5 Kalman Filter

Usually, the situation is given with several measurements from different sensors on the same facts. Each of these measurements provides a normally distributed observations with mean and variance values, the question is how the measurements of the sensors are linked together to get a final result that is

more accurate and has a smaller error.

For this, the Kalman filter is used, on condition that all estimates with normally distributed errors are present. The Kalman filter estimates the fused then turn a new estimate with a new error is normally distributed. As an optimality criterion for the selection of the new estimate is the criterion of minimum variance. The new estimate is selected so that the resulting error is minimized.

A5.1 One-dimensional case

The procedure and effect of the Kalman filter can be best shown an example of the scalar case. Suppose two sensors for measuring distance are set up so that they have the same distance to the object in front of them, e.g. a wall, possess. The sensors must, therefore, measure the same distance value. Furthermore, it is assumed that the measurements of the sensors are affected by an error. The measurements l_1 from sensor 1 and l2 from sensor 2 are distributed as follows:

$$l_1 \sim N(\mu_1, \sigma_1^2)$$

$$l_2 \sim N(\mu_2, \sigma_2^2)$$

Here μ_1 and μ_2 correspond to real distances and σ_1 and σ_2 are the respective standard deviations, which can be determined by a series of measurements. The problem now is how to link the data of the two sensors in order to calculate the present distance from a respective one of the two sensors measuring, which takes account of the properties of both sensors. It is obvious, in this case to use a weighted arithmetic mean of the two distance measurements. The respective weights are inversely proportional to the associated variance.

l1 and l2 are the measurements of the two sensors; new distance value l

with variance σ^2 is calculated by Kalman filtering

$$l = \frac{1}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}} \left(\frac{1}{\sigma_1^2} l_1 + \frac{1}{\sigma_2^2} l_2 \right)$$

$$\frac{1}{\sigma^2} = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}$$

In the case of $\sigma_1 = \sigma_2$ (when using two identical instruments) the equations are reduced as

$$l = \frac{1}{2}(l_1 + l_2)$$

$$\sigma^2 = \frac{\sigma_1^2}{2}$$

A5.2 Multidimensional case

In the n-dimensional case, there are two n-dimensional vectors m1 and m2.

$$m_1 \sim N(\mu_1, \Sigma_1)$$

$$m_2 \sim N(\mu_2, \Sigma_2)$$

The calculation by Kalman filtering improved estimate m with covariance matrix Σ is then given by:

$$\mu = \left(\Sigma_l^{-1} + \Sigma_2^{-1} \right)^{-1} \left(\Sigma_l^{-1} \mu_l + \Sigma_2^{-1} \mu_2 \right)$$

$$\Sigma = \left(\Sigma_l^{-1} + \Sigma_2^{-1} \right)^{-1}$$

At this point it should be noted that the Kalman filter is usually applied in such a way that initially a prediction based on the current state of measurements, and compared with the prediction of the actual sensor measurement. From this, an update to the system state is computed.

REFERENCES

1. J. A. Castellanos, J. M. Martinez, J. Neira and J. D. Tardos, Simultaneous map building and localization for mobile robots: a multisensor fusion approach, Proc. of the 1998 IEEE International Conference on Robotics & Automation, pp. 1244-1249, 1998.
2. R. Madhavan; M. Dissanayake, H.F. Durrant-Whyte, Autonomous underground navigation of an LHD using a combined ICP-EKF approach, Proc. Of the IEEE Conference on Robotics and Automation, pp.3703-3708, 1998.
3. P. Jensfelt, H. Christensen, Laser based poses tracking, Proc. of the IEEE Conference on Robotics and Automation, pp. 2994-2998, 1999.
4. J. Castellanos, J. Martinez, J. Neira, J. Tardos, Simultaneous map building and localization for mobile robots: a multisensor fusion approach, Proc. of the IEEE Conference on Robotics and Automation, pp 1244-1249, 1998.
5. D. F. Wolf and G. S. Sukhatme, Mobile robot simultaneous localization and mapping in dynamic environments, Trans. On Autonomous Robots, Vol. 19, No. 1, pp.53-65, 2005.
6. A. Hernández, J. Ureña, M. Mazo, J.J. García, A. Jiménez, J.A. Jiménez, M. C. Pérez, F.J. Álvarez, C. De Marziani, J.P. Dérutin, J. Sérot, Advanced adaptive sonar for mapping applications, Journal of Intelligent & Robotic Systems, Vol. 55, No. 1, pp. 81-106, 2009.
7. D. Hahnel, R. Triebel, W. Burgard and S. Thrun, Map building with mobile robots in dynamic environment, Proc. of the IEEE International Conference on Robotics and Automation , pp.1557-1563, 2003.
8. J. S. Gutmann, Markov Kalman localization for mobile robots, Trans. On

Pattern Recognition, Vol.2, pp. 601-604, 2002.

9. S. Thrun, J.S. Gutmann, D. Fox, W. Burgard and B. J. Kuipers, Integrating topological and metric maps for mobile robot navigation: a statistical approach, AAAI Proceedings, 1998.
10. D. Saitov and S. G. Lee, Mobile Robot Navigation Based on EWA with Adaption of Particle Filter and Map Merging Algorithms for Localization and Mapping, International Journal of Precision Engineering and Manufacturing, Vol. 12, No. 3, pp. 451-459, 2011.
11. J. O. Wallgrün, Voronoi Graph Matching for Robot Localization and Mapping, Trans. on Computational Science IX, Lecture Notes in Computer Science, Vol6290, pp. 76-108, 2010.
12. S. Se, D. Lowe and J. Little, Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks, International Journal of Robotics Research, Vol. 21, No. 8, pp. 735-758, 2002.
13. S. Se, D. Lowe and J. Little, Vision-Based global localization and mapping for mobile robot, Trans. on Robotics, Vol. 21, Vo. 3, pp. 364-375, 2005.
14. J. A. Castellanos, J. M. Martinez, J. Neira and J. D. Tardos, Simultaneous map building and localization for mobile robots: a multisensor fusion approach, I Proceedings of the 1998 IEEE International Conference on Robotics & Automation, pp. 1244-1249, 1998.
15. F. Lu and E. Milios, Globally consistent range scans alignment for environment mapping. Trans. on Autonomous Robots, Vol. 4, pp. 333–349, 1997.
16. R. E. Kalman, A new approach to linear filtering and prediction problems. Trans. on the ASME–Journal of Basic Engineering, Vol. 82, pp. 35–45, 1960.

17. J. J. Leonard and H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. Trans. on Robotics and Automation, Vol. 7, No. 3, pp. 376–382, 1991.
18. R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics, Trans. on Autonomous Robot Vehicles, pp. 167-193, 1990.
19. C. Stachniss, G. Grisetti, D. Haehnel, and W. Burgard. Improved rao-blackwellized mapping by adaptive sampling and active loop closure. Work-shop on Self-Organization of Adaptive, 2004
20. J. S Gutmann and K. Konolige, Incremental mapping of large cyclic environment, in Computational Intelligence in Robotics and Automation, pp. 318-479, 1999.
21. F. Lu and E. E. Milos, Robot pose estimation in unknown environments by matching 2D range scans, Journal of Intelligent and Robotic Systems, Vol.18, No. 3, pp.249–275, 1997.
22. W. Burgard, D. Fox and S. Thrun, Active mobile robot localization, Proc. of the International Joint Conferences on Artificial Intelligence, pp.1346-1352, 1997.
23. K. Konolige and K. Chou, Markov localization using correlation, Proc. of the International Joint Conference on Artificial Intelligence, 1999.
24. P. H´ebert, S. Betg´e-Brezetz, and R. Chatila, Probabilistic map learning: necessity and difficulties, Proc. of the International Workshop on Reasoning with Uncertainty in Robotics, Amsterdam, 1995.
25. S. Se, D. Lowe and J. Little, Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks, International Journal of Robotics Research, Vol. 21, No. 8, pp. 735-758, 2002.
26. I. J. Cox and G.T. Wilfong, Autonomous Robot Vehicles, Springer-Verlag,

1990.
27. J. Cox, Blanche: Position estimation for an autonomous robot vehicle, In Cox and Wilfong, pp. 221-228, 1990.
28. F. Lu, Shape registration using optimization for mobile robot navigation, PhD thesis.
29. J. C. Latombe, Robot Motion Planning, Kluwer Academic Pulishers, 1991.
30. M. Seder and I. Petrovic, Dynamic window based approach to mobile robot motion control in the presence of moving obstacles, Proc. of the International Conference on Robotics and Automation, pp. 1986-1991, 2007.
31. D. Fox, W. Burgard and S. Thrun, The dynamic window approaches to collision avoidance, IEEE Robotics & Automation Magazine, Vol.3, No. 1, pp.23-33, Mar 1997.
32. A. Stentz, Optimal and efficient path planning for partially-known environment, Proc. of the International Conference on Robotics and Automation, vol.4, pp.3310-3317, May 1994.
33. A. Stenz, The focused D* algorithm for real-time replanning, Proc. of the International Joint Conference on Artificial Intelligence, vol. 2, pp. 1652-1659, August 1995.
34. N. Avaehe and O. D. Faugeras, Maintaining representations of the environment of a mobile robot, Trans. on Robotics and Automation, Vol. 5, No. 6, pp. 804-819, 1989.
35. S. Borthwick and H. Durrant-Whyte, Simultaneous localisation and map building for autonomous guided vehicles. Proc. of the International Conference on Intelligent Robots and Systems, pp. 761-768, 1994.
36. J. Gonzalez, A. Eeina and A. Ollero, Map building for a robot equipped

with a 2D laser rangefinder. Proc. of the International Conference on Robotics and Automation, 1994.

37. J. L. Crowley, World modelling and position estimation for a mobile robot using ultrasonic ranging. Proc. of the International Conference on Robotics and Automation, pp. 674-680, 1989.
38. Y. D. Kwon and J. S. Lee, A stochastic environment modelling method for mobile robot by using 2-D laser scanner. Proc. of the International Conference on Robotics and Automation, pp. 1688-1693, April 1997.
39. J. Leonard, H. Durrant-Whyte and I. J. Cox, Dynamic map building for an autonomous mobile robot. Proc. of the International Conference on Intelligent Robots and Systems, pp. 89-95, 1990.
40. H. F. Durrant-Whvte, Consistent integration and propagation of disparate sensor observations, International Journal of Robotics Research, Vol. 6, No. 3, pp. 3 24, 1987.
41. E. Smith, M, Self, and P. Cheeseman, Estimating uncertain spatial relationships in robotics, In Cox and Wilfong, pp 167-193, 1990.
42. J. A. Castellanos, J. M. M. Montiel, J. Neira, and J. D. Tardos, The SPmap: A probabilistic framework for simultaneous localization and map building, Trans. on Robotics and Automation, Vol. 15, No. 5, pp. 948-952, Oct.1999.
43. D. Ferguson and A. Stentz, Filed*: An Interpolation-based Path Planner and Replanner, International Symposium on Robotics Research, Vol. 8, pp. 239-253, 2005.
44. robots.mobilerobots.com
45. www.intelligentcontrol.es/diego/projects_mobilerobots.html
46. www.robotis.com