

國立臺灣大學電機資訊學院電機工程學系

碩士論文

Department of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

DWA*: 基於速度空間方法與預測式驗證之室內機器

人導航演算法

DWA*: Velocity Space Approach with Look-Ahead
Verification for Indoor Robot Navigation

周執中

Chih-Chung Chou

指導教授：連豐力 博士

Advisor: Feng-Li Lian, Ph.D.

中華民國九十七年十二月

December 2008

致謝

三年前，連豐力教授親自將一臺小型機器車交付給我，那是一切的開始。

從小喜歡看機器人卡通，到了高中立志考上電機系來做機器人，經過彷彿沒有盡頭的學習與研究生涯，到今天算是踏出了一小步。這段日子很漫長，但是並不痛苦，正是因為有許多人的善意幫助。

連豐力教授並不苛求學生，但既細心又嚴格，讓我實際學習到了做研究的方法。並且時時鞭策，使我得能更上一層樓。最讓我敬佩與感謝的，是教授對學生的無私關愛，讓實驗室宛如一個大家庭。和樂的氣氛，使我能夠安心的完成研究工作。

感謝口試委員王傑智教授與李蔡彥教授，提出了許多切中要點的問題與建議；不只使這篇論文更加完善，更指引了我未來研究的方向。特別又要感謝王教授。過去我亦曾在課堂上受過王教授的指導，雖然為時不長，但在那段時間裡，王教授以他充實的學識與熱情給予了我極大的幫助，為尚在摸索學習的我打下堅實的基礎。

感謝士瑋學長、上瑋學長、育霖學長與建廷學長對我的照顧與幫助，對不起我有時會把士瑋跟瑋學長的名字叫混了。感謝同輩的冠傑、平之與一銘長久的陪伴。特別是冠傑，經常與我討論切磋，在研究工作上給了我許多協助。感謝已畢業的上上屆之前的學長姊們，在我還只是大學部專題生時，便以熱心親切的態度帶領我熟悉實驗室的生活，你們的關懷與溫情至今仍常在我心。

最後，要感謝家人在我背後給予的支持，令我不需擔心雜事，順利完成論文。我時時自省，深覺能夠走到今天這一步，並不只因自己努力，更是因為我有這份運氣，到了一個好地方，遇見了許多好人，謹以至誠感謝將此論文獻予你們。

周執中 謹誌

中華民國九十八年一月五日

DWA*: 基於速度空間方法與預測式驗證之室內機器人導航演算法

研 究 生：周執中

指導教授：連豐力 博士

國立臺灣大學 電機工程學系

摘要

為完成行動式機器人的自動導航，基於動態窗格演算法(DWA)，這篇論文提出一名為 DWA*之反應式演算法來達成高速、平順、且不受局部極小值問題影響的導航。原始的 DWA 方法只利用了少部份之環境資訊來在機器人的運動空間中搜尋適當之指令，因此，機器人易於走進局部極小值地區而無法脫出此複雜的環境。為能避開局部極小值地區，DWA*運用了區域分析技術來過濾不適當的動作指令，並使用 A*搜尋演算法進行預測式驗證，藉此決定一組最佳動作指令，其可引導機器人於數步之後得到最佳的結果。最後，這篇論文分別展示了使用聲納測距器及使用雷射測距器的機器人之模擬與實驗結果，由此可顯出 DWA*比起原始的 DWA 有著更為優越的表現。

關鍵字：

動態窗格演算法，局部反應式演算法，速度空間方法，預測式驗證。

DWA*: Velocity Space Approach with Look-Ahead Verification for Indoor Robot Navigation

Student: Chih-Chung Chou

Advisor: Dr. Feng-Li Lian

Department of Electrical Engineering
National Taiwan University

ABSTRACT

Based on the dynamic window approach (DWA) for robot navigation, this thesis presents a local reactive method, called DWA*, for mobile robots to achieve high-speed, smooth, and local-minima-free navigation. The original DWA utilizes only a small part of environmental information to search for a proper motion command in the robot's motion space. Hence, the robot can be easily driven into local-minima area and trapped in complex environment. In order to escape from the local-minima area, DWA* applies region analysis technique to filter improper commands, and uses the A* search algorithm with the look-ahead verification to determine the optimal command which can lead the robot to the best consequence after a designated number of steps. In this thesis, extensive simulation and experimental studies using sonar-based or

laser-based mobile robot are presented to illustrate the excellent performance of using the DWA* compared with that of using the original DWA.

Keywords:

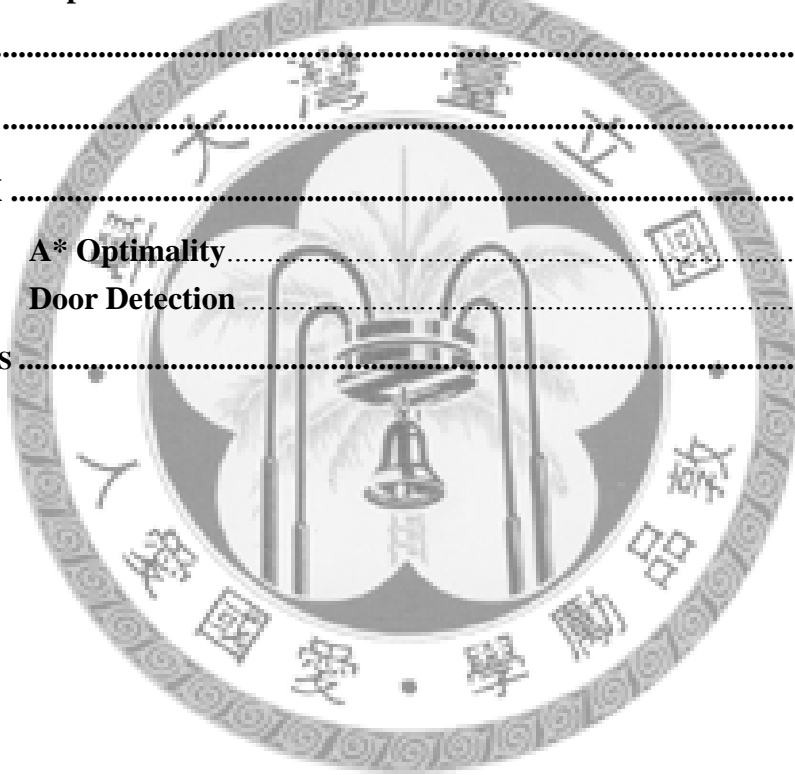
Dynamic window approach, local reactive method, velocity space approach, look-ahead verification.



Contents

摘要	I
ABSTRACT	II
CONTENTS	IV
LIST OF FIGURES	VI
CHAPTER 1	1
INTRODUCTION	1
1.1 Motivation	1
1.2 Problem Formulation	2
1.3 Contribution of the Thesis	3
1.4 Organization of the Thesis	4
CHAPTER 2	5
RELATED WORKS	5
2.1 Development of Global Path Planning Methods	5
2.2 Development of Local Reactive Methods	7
2.3 Hybrid Methods	9
2.4 Search Problem Formulation	10
CHAPTER 3	18
DYNAMIC WINDOW APPROACH	18
3.1 Procedure of DWA	18
CHAPTER 4	22
MODIFIED ALGORITHM DWA*	22
4.1 Procedure of DWA*	22
4.2 Interval Analysis	24
4.3 Region Detection	26
4.4 Set of Situations	29
4.5 In-Region DWA	31
4.6 Look-Ahead Verification	33
4.7 Implementation on Laser-Based Mobile Robots	35
4.8 Implementation on Sonar-Based Mobile Robots	36

CHAPTER 5	40
SIMULATION RESULTS.....	40
5.1 Simulation Platform	40
5.2 Simulation Results for Laser-Based Robot	41
5.3 Simulation Results for Sonar-Based Robot	50
CHAPTER 6	56
EXPERIMENTAL RESULTS.....	56
6.1 Experimental Results for Laser-Based Robot	56
6.2 Experimental Results for Sonar-Based Robot	62
CHAPTER 7	67
DISCUSSION	67
APPENDIX	70
A.1 A* Optimality.....	70
A.2 Door Detection	71
REFERENCES	73



List of Figures

FIGURE 1.1: A COMPARISON OF ROBOT NAVIGATION ALGORITHMS.	2
FIGURE 2.1: AN EXAMPLE OF SEARCH TREE.	12
FIGURE 2.2: AN EXAMPLE OF LOCAL MINIMA.	16
FIGURE 3.1: AN EXAMPLE OF DWA.....	20
FIGURE 4.1: THE PROCESS OF DWA*.....	23
FIGURE 4.2: AN EXAMPLE OF DWA*.....	24
FIGURE 4.3: AN EXAMPLE OF REGION DETECTION.	28
FIGURE 4.4: DIFFERENT TYPES OF REGIONS.	31
FIGURE 4.5: COMPARISON OF LASER AND SONAR DATA.	38
FIGURE 5.1: SIMULATION RESULT 1.....	46
FIGURE 5.2: SIMULATION RESULT 2.....	48
FIGURE 5.3: SIMULATION RESULT 3.....	50
FIGURE 5.4: SIMULATION RESULT 4.....	53
FIGURE 5.5: SIMULATION RESULT 5.....	55
FIGURE 6.1: EXPERIMENTAL RESULT 1.....	59
FIGURE 6.2: EXPERIMENTAL RESULT 2.....	61
FIGURE 6.3: EXPERIMENTAL RESULT 3.....	64
FIGURE 6.4: EXPERIMENTAL RESULT 4.....	66
FIGURE A.1. EXAMPLE OF DOOR DETECTION.	72
TABLE 2.1: NAVIGATION METHODS UNDER SEARCH PROBLEM FORMULATION.....	17

Chapter 1

Introduction

1.1 Motivation

Robot navigation is an important application research in mobile robotics. Generally speaking, robot navigation problem can be defined as follows: given a start location and a goal location, the robot is asked to move from the start to the goal.

Present robot navigation methods can be classified into two types: global path planning and local reactive methods. A simple comparison of these methods is shown in [Figure 1.1](#). Global methods assume that the environment is completely known and plan a path according to overall map information. However, they often do not consider the robot's dynamic and needs enormous computing time. On the other hand, local methods only utilize environment information of the robot's neighborhood to compute a proper motion command. Roughly speaking, global methods utilize environment information more to find a whole optimal path, while local methods care more about

robot dynamic. Though local methods often do not guarantee the path optimality, they can be recomputed fast to manage varying environments. Thus, it is better to apply local methods in practical cases.

However, sometimes local methods can make undesirable decisions because that they only utilize part of known information. For this reason, it is hoped to design a local method which can utilize the environment information efficiently.

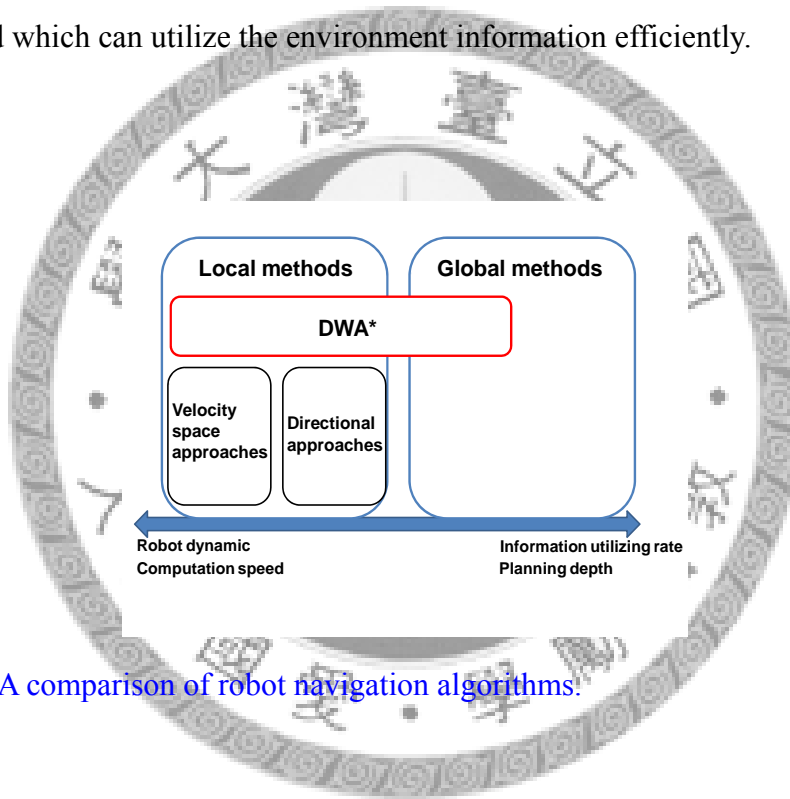


Figure 1.1: A comparison of robot navigation algorithms.

1.2 Problem Formulation

In this thesis, the robot is assumed to operate in an unknown static environment. The term ‘unknown’ means that the robot does not have a built-in world model; it has to complete the navigation task using sensory information. ‘Static’ means that the environment does not change or changes slowly. Another assumption is that the robot

is equipped with dead-reckoning system so that it always knows the goal location related to its present position. The robot is assumed to be a circular wheeled mobile robot. It has two degrees of freedom: translational velocity and rotational velocity. Therefore, the robot trajectories can be represented as circular lines.

1.3 Contribution of the Thesis

A new navigation method DWA* is presented in this thesis. DWA* is a local reactive method and based on a velocity space approach DWA [4: Fox et al. 1997], which directly searches an optimal velocity command from the set of all executable velocities for the robot. By adjusting the range of search space, DWA can be applied for various types of wheeled robots. However, in the searching of DWA, the intention of goal-directing and obstacle-avoidance are mixed. For this reason, sometimes DWA may select an evidently improper velocity. Different from original DWA, DWA* preserves several candidate velocities and then makes a selection by look-ahead verification [8: Ulrich & Borenstein 2000].

Most local reactive methods can easily get stuck in trap situations like U-shape obstacles. It is because that they only utilize a little part of known information. This problem can be solved by look-ahead verification, which predicts the robot's state after

several steps and evaluates candidate motion commands according to the consequence.

By adding look-ahead verification, DWA* can utilize the environment information completely to achieve high-speed, smooth and local-minima-free navigation.

1.4 Organization of the Thesis

In this thesis, related works is discussed in [Chapter 2](#), and the original DWA is briefly introduced in [Chapter 3](#). The details of DWA* is presented in [Chapter 4](#). In [Chapter 5](#), Simulation results are discussed to demonstrate the characteristics of DWA* and several experimental results of DWA* on real mobile robots are shown in [Chapter 6](#). Finally, the limitations and future works of DWA* are discussed in [Chapter 7](#).

Chapter 2

Related Works

In this chapter, related researches of robot navigation techniques are introduced. In [Section 2.1](#) and [Section 2.2](#), the development of global path planning methods and local reactive methods are introduced. In [Section 2.3](#), hybrid methods which integrate global and local methods are presented. In [Section 2.4](#), robot navigation problem is re-defined as a search problem. Under the search problem formulation, the differences of above methods are discussed and the concept of DWA* is illustrated.

2.1 Development of Global Path Planning Methods

The goal of global path planning methods is to find a path connecting the robot's current position and the goal. The earliest global path planning technique is *visibility graph*. In this method, obstacles are assumed as polygons and their vertices are saved in a vertex set V , and an edge set E is constructed by connecting each pair of vertices in

V. The robot path is then searched by topological methods. Although the visibility graph method can find a path from the start to the goal, it is obvious that a path passing the vertices of obstacles is not good enough.

For finding the optimal path, the cell decomposition and graph search methods are used. In these methods, the map is separated into many grids and a path can be represented as a list of grids on the map. Navigation function *NF1* [6: Brock & Khatib 1999] is one of the simplest methods. It is a wave-front planner [15: Choset et al. 2005] which affords the simplest solution to local minima problem. NF1 starts at goal and uses wave-propagation concept to label all grids with the distance to the goal. As a result, a local minima-free path can be determined via gradient descent from the start grid.

Although NF1 can find a local minima-free path, it is a breadth-first search method. In many cases, it is wished to find the optimal path without visiting every grid.

A algorithm* [14: Russell & Norvig 2003] can solve this problem perfectly. A* is the most popular global method today, it evaluates each grid node by a cost function:

$$f(n) = g(start, n) + h(n, goal) \quad (1.1)$$

where $g(start, n)$ is the true path cost from the start node to node n , and $h(n, goal)$ is a heuristic function, which represents the estimated cost of the shortest path from n to the goal. Different from NF1, A* is a best-first search (BFS) algorithm, that is, the

node with minimum cost will be selected for expansion at first. The optimality of A* is guaranteed if the function $h(n_1, n_2)$ satisfies some conditions. Dynamic A* (D*) [13: Stentz 1994] is a modified version of A*. It is developed to solve the problem of re-planning in changing environment. D* utilizes the computation result obtained from last iteration and updates only the parts influenced by changed nodes. Therefore, D* performs much more efficient than A* in the re-planning stage.

2.2 Development of Local Reactive Methods

The most popular local reactive methods today can be classified as *directional approaches* and *velocity space approaches*. Directional approaches divide the navigation problem into two parts. First, sensory information is analyzed for finding a proper direction for the robot. Second, the robot is controlled to move toward the direction. However, because of the kinematic and dynamic constraints, the robot may be not able to move toward the determined direction smoothly. Hence, velocity space approaches are developed to solve the problem. Instead of finding a direction, velocity space approaches directly search for a proper velocity vector, containing transitional and rotational velocity. By limiting the search space, the problem induced by the kinematic and dynamic constraints can be easily solved.

The earliest directional approach is *virtual force field* (VFF) [1: Borenstein & Koren 1990]. VFF draws a local grid map according to sensory information and then uses potential field method to avoid obstacles. After VFF, *vector field histogram* (VFH) was developed [2: Borenstein & Koren 1991]. VFH transfers a two-dimensional grid map into a one-dimensional polar histogram. VFH first analyzes the histogram to find several navigable open areas and is able to avoid local minima problem suffered by VFF. VFH+ reduces some of the parameter tuning in VFH and approximates the robot trajectory better. Hence, it can drive robots smoother and safer [7: Ulrich & Borenstein 1998]. *Nearness diagram* (ND) [9: Minguez & Montano 2004] inherits the concept of VFH and classifies the state of robot's neighborhood into different scenarios. By applying different control methods in different scenarios, ND can drive robot safely in narrow and complex environments.

The earliest velocity approach is *curvature velocity method* (CVM) [3: Simmons 1996]. After that, *dynamic window approach* (DWA) was developed [4: Fox et al. 1997]. DWA inherits the concept of CVM and becomes one of the most popular velocity space approaches today. Model-based dynamic window approach (μ DWA) was then developed [5: Fox et al. 1998]. μ DWA integrates real and “virtual” sensor data, which is derived from a map of the environment, to compensate for the problem of sensor limitation.

Compared with velocity space approaches, directional approaches like VFH and ND are better at solving local minima problem. The reason is that the process of ‘finding a proper direction’ can be seen as ‘planning for next several steps.’ Therefore, more environment information can be utilized in directional approaches than in velocity space approaches. On the other hand, velocity space approaches can drive robots faster and smoother than the directional approach because of their attention to robot dynamic constraints.

2.3 Hybrid Methods

For completing the robot navigation, both global methods and local methods are needed. There are some researches about the integration of global and local methods. In [6: Brock & Khatib 1999], NF1 is integrated with DWA. In [10: Seder et al. 2005], the integration of DWA and D* algorithm is presented, the authors then developed the function of moving obstacle avoidance in [11:Seder & Petrović, 2007]. In these methods, a whole path is first computed by global methods and local methods are applied to track the path. Therefore, the robot can move along the computed path with real-time obstacle avoidance. However, since the motion definition of the global method does not correspond with that of the local method, the tracking performance

may be poor.

VFH* [8:Ulrich & Borenstein 2000] is different from above methods. Rather than searching paths by global methods, VFH* uses *look-ahead verification* to consider the information beyond the immediate surroundings of the robot. The concept of look-ahead verification is to recursively re-compute the local method to predict the robot's new state after several steps. Robot motion is determined by predicting the consequences after several steps of executing present candidate motion commands. With the help of look-ahead verification, VFH* can deal problematic situations better than pure local methods. The concept of look-ahead verification is used in DWA* to solve local minima problem.

2.4 Search Problem Formulation

For realizing and comparing the characteristic of above methods better, it is needed to find a generalized formulation of these methods. Based on the literature survey, the robot navigation problem is defined as a search problem.

Figure 2.1 shows a search problem formulation for robot navigation problem, the root node depicts the present position of robot. The target of a navigation method is to determine the optimal motion which can lead robot to the goal with minimum cost.

Under this definition, each navigation method has four key factors:

- **Robot state space** $X = \{x_k^n\}$: Each node in [Figure 2.1](#) represents a reachable robot configuration. Where k depicts depth and n depicts the sequence number of this node. The root node depicts the start position of robot.
- **Robot motion space** $U_{n,k} = \{u_k^m : x_{k+1}^m = f(x_k^n, u_k^m)\}$: For each state x_k^n , $U_{n,k}$ depicts the union of all executable motion for the robot. $x_{k+1}^m = f(x_k^n, u_k^m)$ is the robot's dynamic equation.
- **Cost function** $Cost(x_{k+1}^m, u_k^m)$: This function reflects the *cost* of choosing motion u_k^m to change the robot state from x_k^n to x_{k+1}^m , the motion with minimum cost in $U_{n,k}$ is chosen.
- **Maximum search depth** n_d : The maximum level of the search tree. In global methods, robots search continuously until the goal node is expanded. Hence, n_d equals the depth of goal node. While in local methods n_d often equals 1.

[Table 2.1](#) shows the differences of navigation methods introduced in the last section.

Motion space definition and search depth are the two key differences of global and local methods.

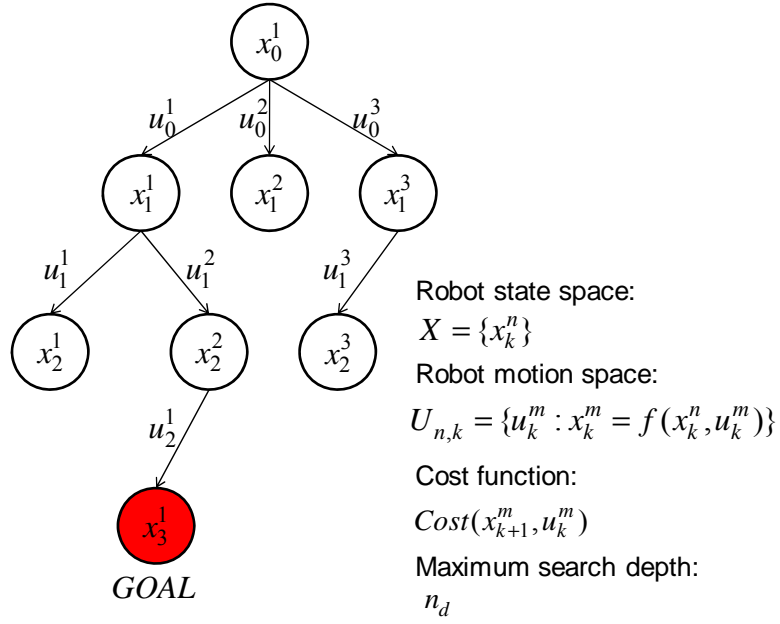


Figure 2.1: An example of search tree.

In global methods, for solving the problem easier, motions are often defined without considering robot dynamic. For example, in A* and NF1, robot motion is defined as “moving to one of the 8 neighbor grids.” However, it is impossible for a 4-wheeled robot to move right or left straightly.

On the other hand, although local methods design motion space better, the path optimality is not guaranteed because of limited search depth and local minima problem.

Under the search problem formulation, the local minima are as follows:

- **Local minimum:** When the robot is in state x_k^n , if it satisfies $Cost(x_k^n, 0) \geq Cost(x_{k+1}^m, u_k^m)$ for all $u_k^m \in U_{n,k}$ and $x_k^n \neq goal$, then we say x_k^n is a **local minimum**.

Here $Cost(x_k^n, 0)$ means the cost of doing nothing but keeping in present state x_k^n .

Obviously, if robot gets stuck in local minima, it can never reach the goal. In many cases, even if the robot does not stop on the local minima, it may keep moving around the neighborhood of local minima and it may not be able to complete navigation task.

For realizing the local minima problem better, a simple test algorithm is designed and used to observe the performance. The algorithm **TEST** is defined as follows:

- **Robot state space:** The two-dimensional position of robot.
- **Robot motion space:** Assume that the robot can move straightly on 181 directions, from -90 degrees to 90 degrees. It can move at most 0.5m in one step.
- **Cost function:** The cost function of TEST1 is the weighting sum of two sub-functions: **Cost1** and **Cost2**. **Cost1** represents the distance between the new state and the goal, **Cost2** depicts the maximum distance where the robot can move before collisions happen. **Cost1** and **Cost2** are both linearly projected onto the interval [0 1]. The final cost function equals $0.8*Cost1 + 0.2*Cost2$.
- **Maximum search depth:** In TEST, the depth equals 1.

Figure 2.2(a) shows a situation which can lead the robot to local minima state. In this situation, the robot fronts a U-shape obstacle and there is a local minima state at the bottom of this obstacle. After comparing all executable motions, TEST algorithm

determined to move forward straightly.

Figure 2.2(a) is also a typical situation which can cause local minima problem. It is because that majority of local methods consider the distance to goal as the main part of their cost function, just like TEST. However, in this situation, the robot must once move away from the goal to avoid the obstacle. If the obstacle is wide enough, the robot may consider that it is too expensive to detour and get stuck in front of the obstacle, as TEST does.

For solving the local minima problem, there are mainly four approaches:

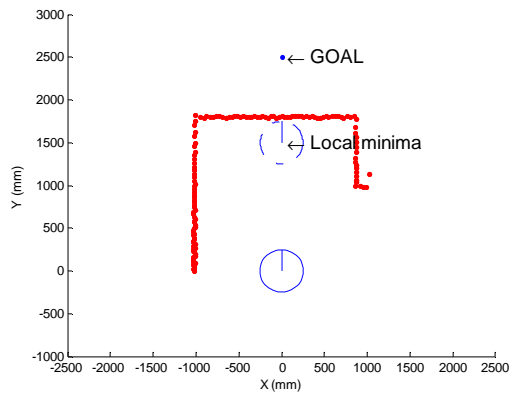
- **Limited motion space:** The environmental information is analyzed to filter improper motions. For example, ND applies region analysis technique to remove motions which are possible to lead the robot to local minima states in advance.
- **Multi-policy:** When the robot gets stuck in local minima, it will switch to another navigation method. For example, in DWA, when the robot finds that there are no admissible trajectories to translate, it will rotate away the obstacles until it is able to translate again.
- **Combined with global methods:** In these methods, a path is found by global methods at first and the cost functions are designed for tracking the optimal path.

For example, NF1+DWA [6: Brock & Khatib 1999] or A*+DWA [10: Seder & Petrović 2005] [11: Seder & Petrović 2007]. Nevertheless, the tracking

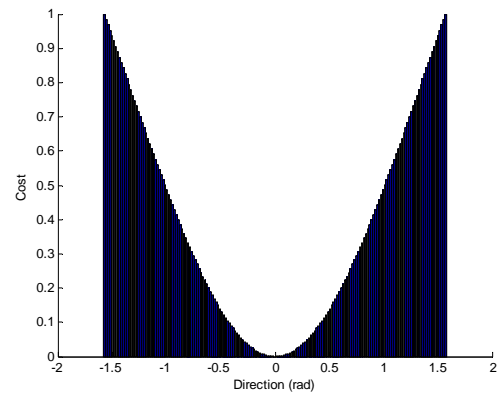
performance may be poor because the motion definition of the global method does not correspond with that of the local method.

- ***Look-ahead verification***: In these methods, the robot predicts its new position after several steps to evaluate the consequence, such as VFH* [8: Ulrich & Borenstein 2000] and DWA*. In fact, adding look-ahead verification can be seen as increasing maximum search depth of the algorithm. Compared with the last group of methods, these methods can track the predicted path better because they use the same approach to drive robot and predict new positions. However, these methods spend much more computing time to find a path to the goal. Therefore, in most practical cases, the maximum search depth is finite and the robot trajectory may not be optimal.

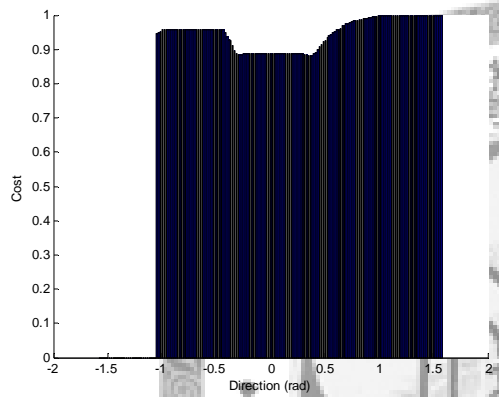
The objective of the approach presented in the thesis is a high-speed, smooth and local-minima-free navigation method. For this reason, DWA* is based on a velocity space approach DWA and integrates with look-ahead verification technique.



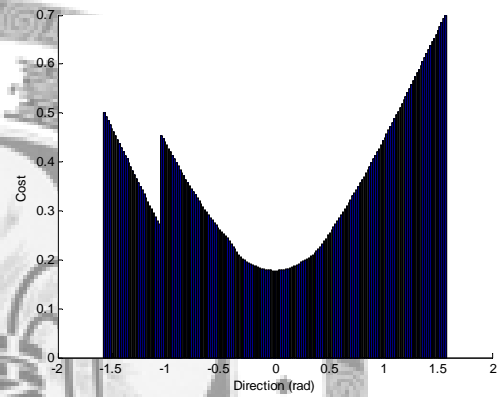
(a)



(b)



(c)



(d)

Figure 2.2: An example of local minima.

(a) U-shape obstacle. The solid-line circle depicts the present position of robot; the dot-line circle depicts the local minima state under *TEST*.

(b) *Cost1*, the horizontal axis depicts the directions of robot motion.

(c) *Cost2*.

(d) Cost function, $0.8*Cost1 + 0.2*Cost2$.

Table 2.1: Navigation methods under search problem formulation.

Algorithm	State definition	Motion definition	Cost function definition	Maximum search depth
Global path planning methods				
Visibility graph	Positions of obstacle vertices	Moving along edges	Distance to the goal	Infinite
NF1	Positions of map grids	Moving to neighbor grids	Distance to the goal	Infinite
A*	Positions of map grids	Moving to neighbor grids	Estimated total cost of getting to goal through the grid.	Infinite
Local reactive methods				
VFH+	Robot positions	Moving toward the selected direction	Determined by objective functions	1
VFH*	Robot positions	Moving toward the selected direction	Estimated total cost of getting to goal through the direction	≥ 1
ND	Robot positions	Moving toward the selected direction	Distance to the goal	1
DWA	Robot positions	Moving on the selected velocity	Determined by objective functions	1

Chapter 3

Dynamic Window Approach

Dynamic window approach (DWA) is introduced in this chapter. DWA is one of the most popular velocity space approaches. Instead of computing a proper velocity according to the obstacle configuration, DWA searches all possible velocities for the robot to find the optimal solution. Hence, DWA can achieve high-speed and smooth navigation for various types of mobile robots.

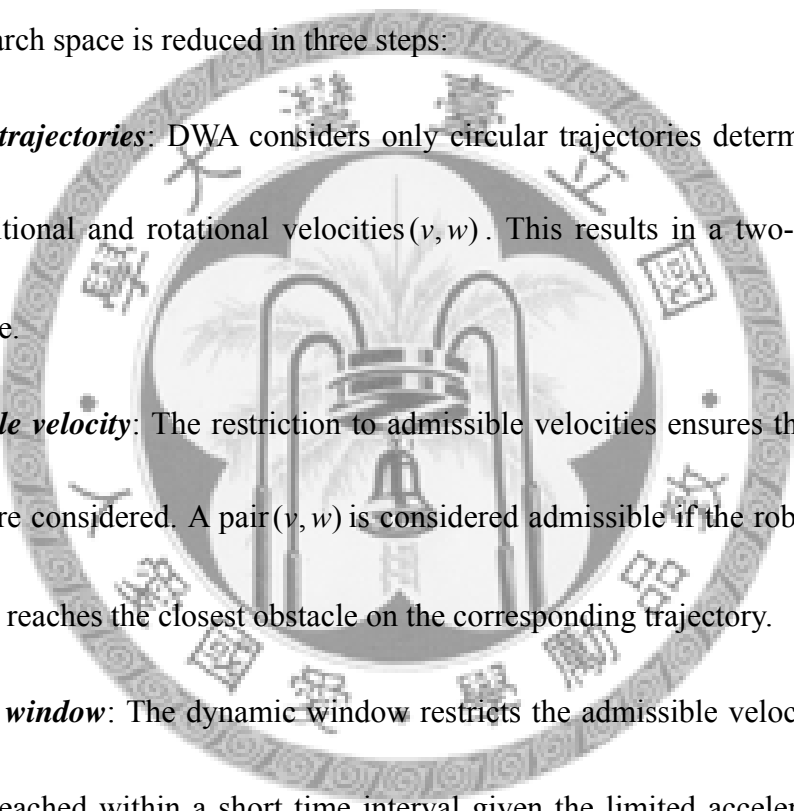
3.1 Procedure of DWA

Dynamic window approach (DWA) is one of the most popular velocity space approaches. Instead of computing a proper velocity according to the obstacle configuration, DWA searches all possible velocities for the robot to find the optimal solution.

DWA is originally presented in [4: Fox et al. 1997] and briefly introduced in the

section. In DWA, the trajectory of a robot can be described as a sequence of circular and straight line arcs. The robot then searches all the possible values in velocity space (v, w) that are free of collision and can be reached within the next sampling time interval Δt . Figure 3.1(a) shows an example where several collision-free trajectories for the robot are illustrated.

The search space is reduced in three steps:

- 
- (a) **Circular trajectories**: DWA considers only circular trajectories determined by the pair of transitional and rotational velocities (v, w) . This results in a two-dimensional velocity space.
 - (b) **Admissible velocity**: The restriction to admissible velocities ensures that only safe trajectories are considered. A pair (v, w) is considered admissible if the robot is able to stop before it reaches the closest obstacle on the corresponding trajectory.
 - (c) **Dynamic window**: The dynamic window restricts the admissible velocities to those that can be reached within a short time interval given the limited accelerations of the robot.

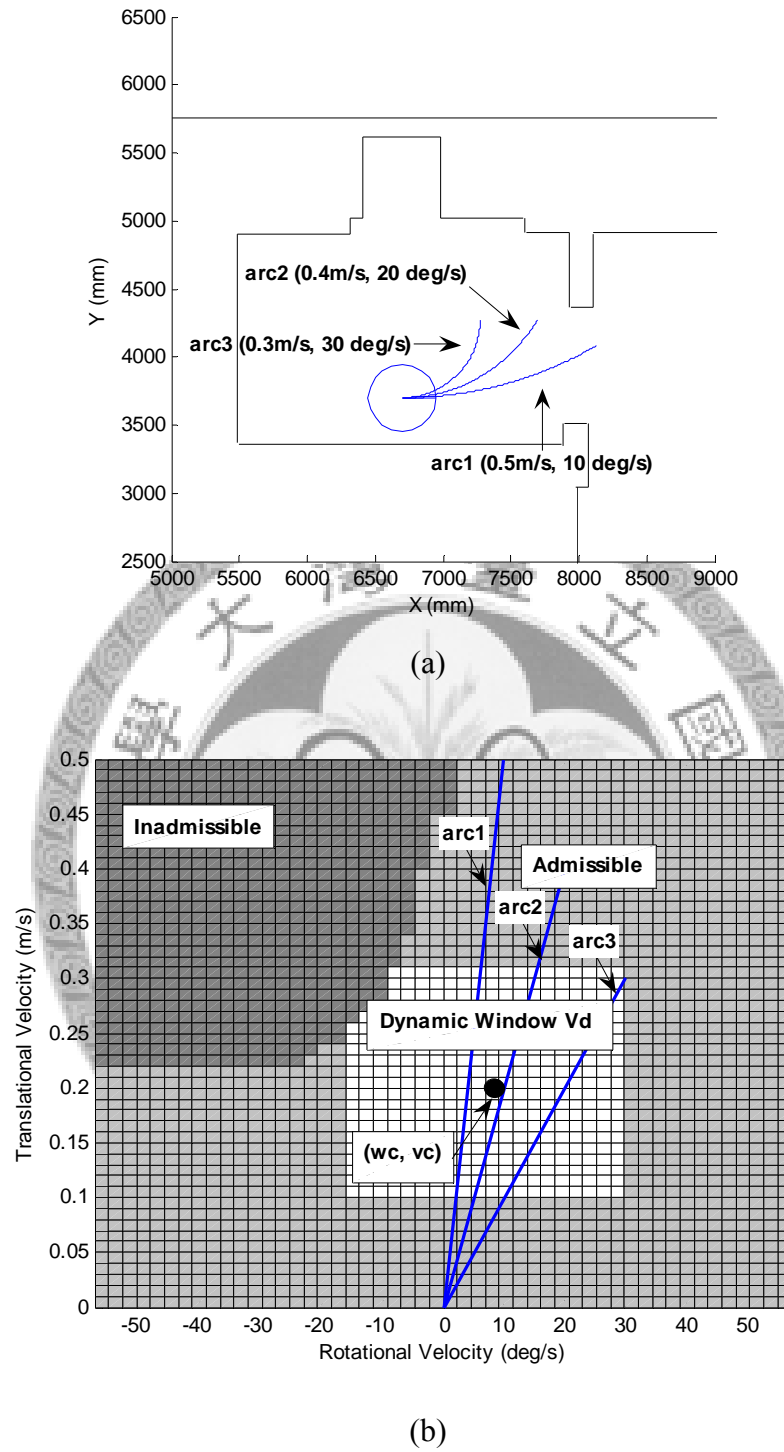


Figure 3.1: An example of DWA.

(a) Robot in physical space. The circle represents robot. These curves from the robot are trajectories generated by different velocities (m/s, rad/s) in the next 3 seconds.

(b) The restricted search space of DWA. Each point in the space represents a velocity vector (v, w) . The horizontal axis represents rotational velocity, and the vertical axis represents transitional velocity;

The restricted search space is shown in Figure 3.1(b). Each point in the space represents a velocity vector (v, w) , and only the velocity vectors in the white region are considered. In the search space, if there are two points locating on the same line passing the origin, it means that the two velocity vectors drive the robot into the same arc trajectory. The blue lines in Figure 3.1(b) correspond to the arc trajectory in Figure 3.1(a).

DWA uses some *objective functions* to find the optimal velocity vectors. In [4: Fox et al. 1997], the following three objective functions are used:

- (a) **Target heading (heading)**: *heading* is a measure of progress towards the goal location. It is maximal if the robot moves directly towards the target.
- (b) **Clearance (dist)**: *dist* is the distance to the closest obstacle on the trajectory. The smaller the distance to an obstacle the higher is the robot's desire to move around it
- (c) **Velocity (vel)**: *vel* is the forward velocity of the robot and supports fast movements.

DWA chooses the velocity vector which maximizes the weighting sum of these objective functions:

$$\alpha(1) \cdot \text{heading}(v, w) + \alpha(2) \cdot \text{dist}(v, w) + \alpha(3) \cdot \text{vel}(v, w) \quad (3.1)$$

Chapter 4

Modified Algorithm DWA*

In this Chapter, details of DWA* are presented. [Section 4.1](#) shows the procedure of DWA*. The local reactive method part of DWA* is introduced in [Section 4.2](#) – [Section 4.5](#). Look-ahead verification part is shown in [Section 4.6](#). [Section 4.7](#) and [Section 4.8](#) are implementation details of DWA* on laser-based robots and sonar-based robots.

4.1 Procedure of DWA*

The procedure of DWA* is shown in [Figure 4.1](#). In DWA*, a search tree is maintained, and each tree node represents a robot position. The right side of [Figure 4.1](#) shows the procedure of node expansion. First, the environment information is realized as interval configuration. Second, the intervals are analyzed to find navigable regions. Third, for each region, a candidate velocity is found by in-region DWA. Finally, for

each candidate velocity, a new robot position is computed as a new node. New nodes are expanded continuously until the tree depth exceeds a constant.

After the expansion stops, the deepest node is determined as goal state (it is because of the optimality of A* search, which is discussed in *appendix*), and the present candidate velocity which can lead the robot to the goal state is selected. Figure 4.2 is an example. In this case, robot has three candidate motion commands to choose: (1) moving left-forward, (2) moving forward and (3) turn right, and the robot selects the first command which can lead it to the goal state.

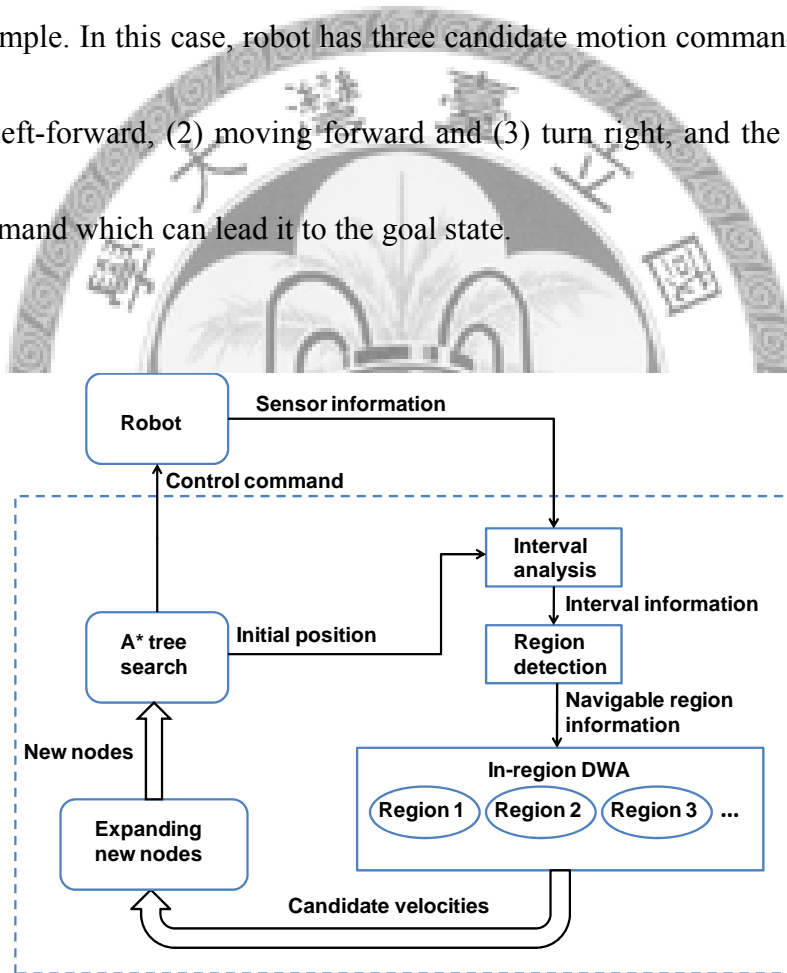


Figure 4.1: The process of DWA*

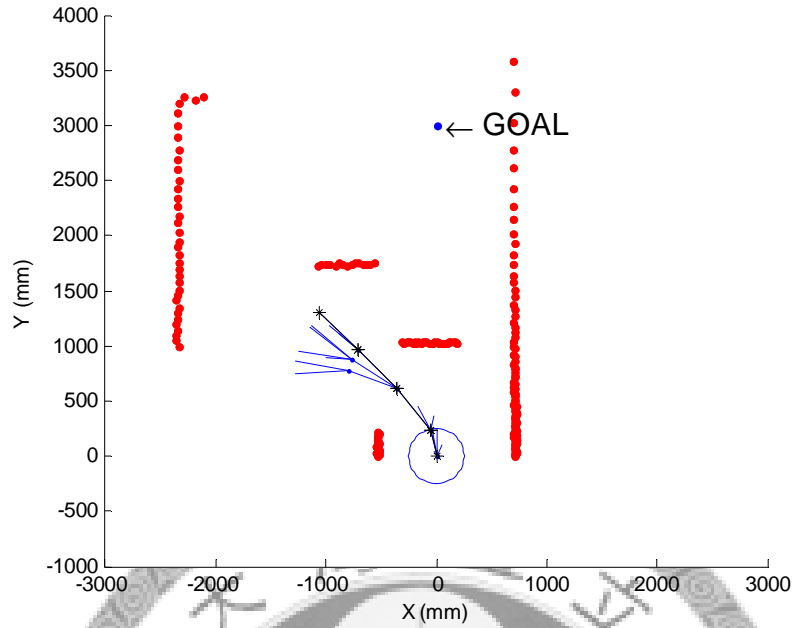


Figure 4.2: An example of DWA*. The blue lines spread from the robot position represent expanded paths. The selected path is indicated by '*'.

4.2 Interval Analysis

Velocity space can be realized as a set of intervals. An interval is formed by two lines in the velocity space. Look at Figure 3.1(b), a straight line in velocity space depicts a circular trajectory in physical space. Hence, an interval in velocity space means a region between two circular robot trajectories in the physical space.

In DWA*, the velocity space is separated as 181 intervals, and the *clearance* of each interval is computed. The clearance is the distance to the closest obstacle if the robot is asked to move in this interval. Its meaning is the same as the clearance mentioned in Section III. The intervals blocked by an obstacle are computed as

following:

$$\begin{aligned} R1 &= \frac{xo^2 + yo^2 - ro^2}{2(yo - ro)} \\ R2 &= \frac{xo^2 + yo^2 - ro^2}{2(yo + ro)} \end{aligned} \quad (4.1)$$

Two circular trajectories with radius R1 and R2 are found by the above equation, and the clearance values of intervals with a radius between R1 and R2 are updated.

There are two merits of using interval analysis in DWA*. First, the computation time is reduced. In original DWA, to compute the clearance values, each obstacle should be checked for each velocity vector in dynamic window. If the size of dynamic window is n and the number of obstacles is o , the computation complexity is $O(o \times n)$. On the other hand, if the interval analysis technique is applied, each obstacle is processed only once to update the interval clearance values and each velocity vector can be directly determined which interval it belongs to. Therefore, the computational complexity becomes $O(o) + O(n)$. In DWA*, because of look-ahead verification, the interval configuration may be repeatedly computed for tens or hundreds of times, and the time cost elimination becomes significant.

The second merit is that the vector field histogram in velocity space can be constructed so that the concepts of directional approaches can be utilized. For example, navigable region analysis is applied in DWA* for better performance, it will be discussed in next section.

4.3 Region Detection

Navigable region analysis is an important part of directional approaches. In directional approaches, a navigable region represents a *free walking area* [9: Minguez & Montano 2004] which contains no obstacles, and a candidate direction for robot motion is derived for each navigable region. By realizing the environment of the robot's neighborhood as a union of navigable region, the robot motion space can be reduced reasonably.

Different from VFH and ND, in DWA*, a navigation region is defined as a set of motions which can lead the robot into the same free walking area. Therefore, similar methods in directional approaches can be used to filter improper motions while the merits of velocity space approaches are still preserved.

In DWA*, a navigable region is composed of two contiguous *interval discontinuities*, the interval discontinuity and navigable region are defined as follows:

- **Interval discontinuity:** An interval discontinuity exists between two adjacent intervals i and j if they satisfy one of the following conditions: $|dist(i) - dist(j)| > 2R$ or $(dist(i) - d_t)(dist(j) - d_t) > 0$. Here $dist(i)$ means the clearance value of interval i , R represents robot radius and d_t is

a threshold value. Both of the two conditions imply that there is a passable gateway for the robot.

- **Navigable region:** A navigable region exists between two adjacent interval discontinuities if at least one of them is *rising*. A rising discontinuity is defined as follows: Given a region V and a discontinuity formed by two adjacent intervals i and j , i is on the edge of V and j is out of V , the discontinuity is rising for V if $dist(i) > dist(j)$.

Figure 4.3 illustrates an example where six discontinuities (d1~d6) and three regions (R1~R3) are shown in physical space and velocity space. Here d1 and d2 are both rising for R1, and, similarly, d2 and d4 are rising for R2, d5 and d6 are rising for R3. All the intervals in R1, R2 and R3 depict robot trajectories for avoiding obstacles. Note that d1 and d6 represent the discontinuities next to the robot trajectories of $v=0$, that is, rotating at the same place.

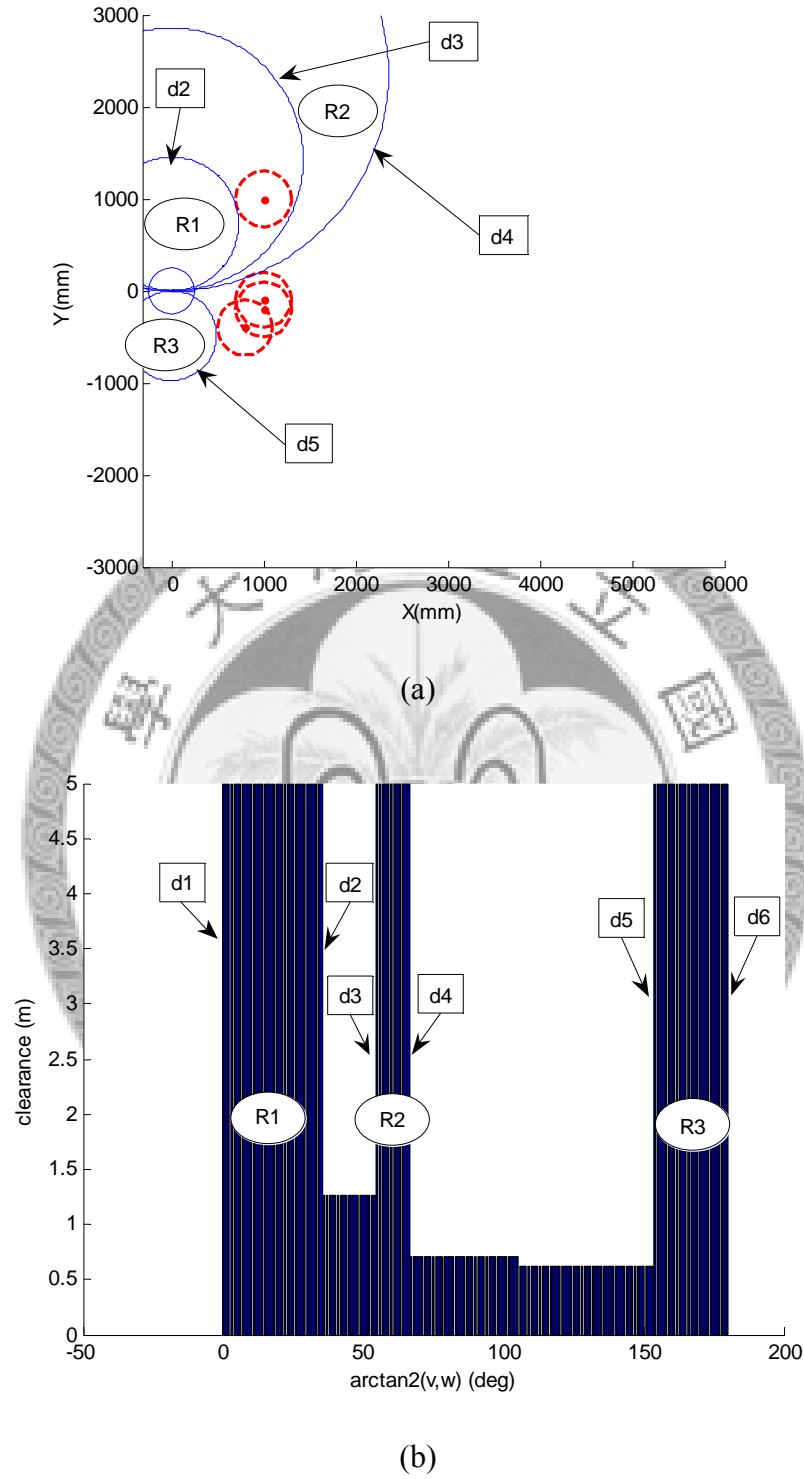


Figure 4.3: An example of region detection.

(a) Robot in the physical space.

(b) Vector field histogram in velocity space, the x-axis indicates different trajectories and the y-axis indicates interval clearances.

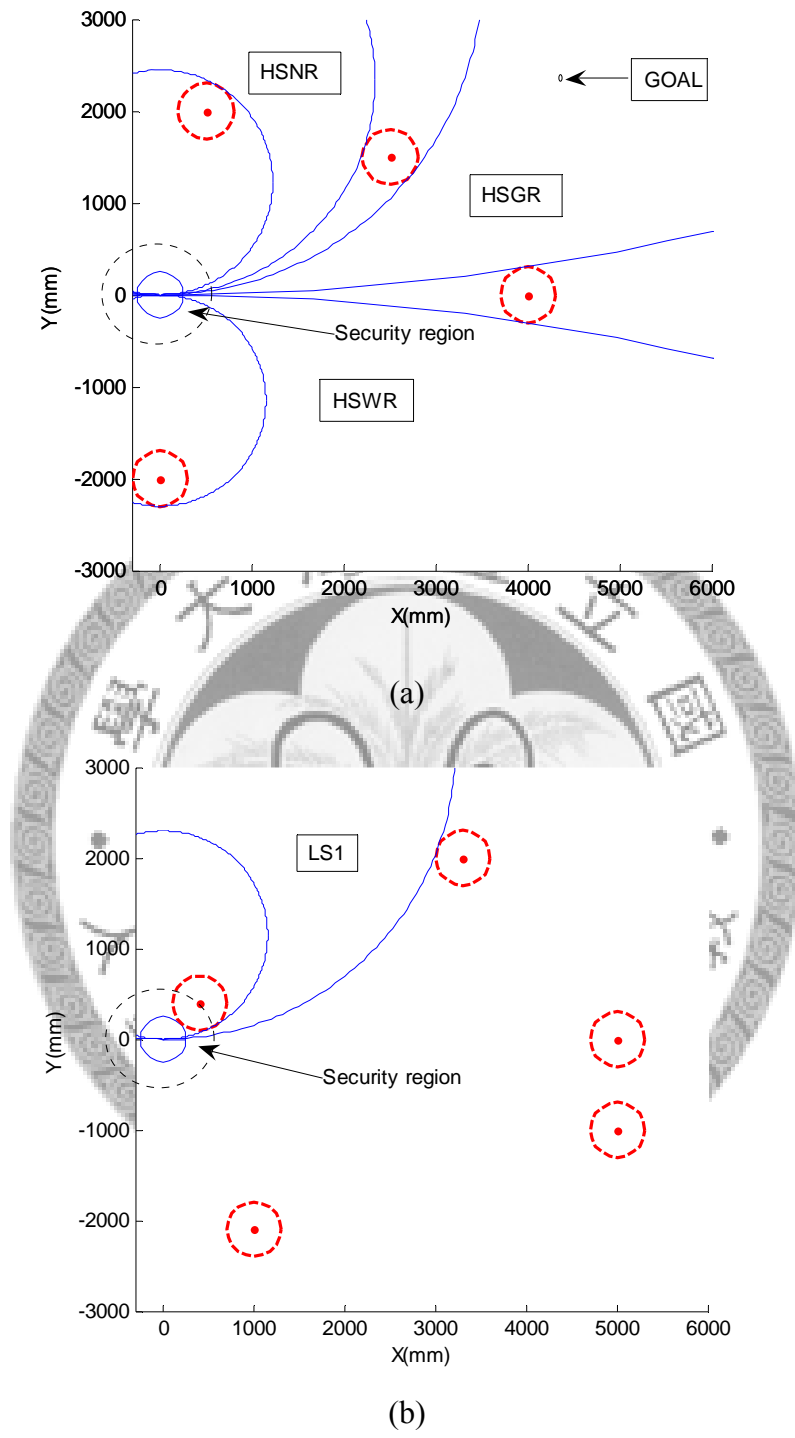
4.4 Set of Situations

For each navigable region, a candidate velocity is derived by in-region DWA.

However, it is reasonable to apply different policies in different type of regions. [Figure](#)

[4.4](#) shows all region types defined in DWA*. In DWA*, navigable regions are classified according to width and safety defined as follows:

- **High safety wide region (HSWR)**: The robot is in high safety state, that is, there are no obstacles in the robot's security region, and the region is wide. The navigable region is defined as a wide region if the number of intervals in this region exceeds a constant s_w .
- **High safety narrow region (HSNR)**: The robot is in high safety state and the region is narrow.
- **High safety goal in region (HSGR)**: The robot is in high safety state and the goal can be reached in little time if the robot moves on a trajectory in this region.
- **Low safety 1 (LS1)**: Robot is in low safety state, that is, there are obstacles in the robot's security region, and the obstacles are only on one side of this navigable region.
- **Low safety 2 (LS2)**: LS2 means that the robot is in low safety state and the obstacles are on both sides of this navigable region.



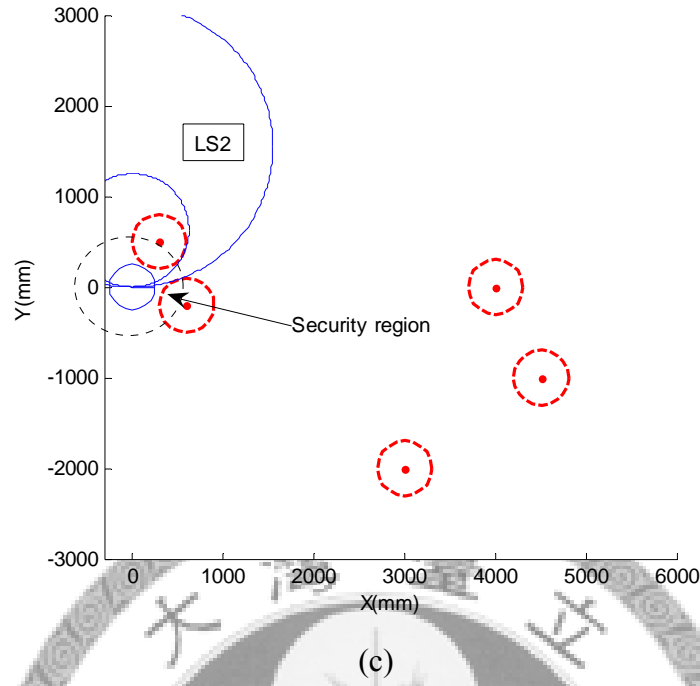


Figure 4.4: Different types of regions.

(a) High safety regions, the dot circle is security region..

(b) Low safety 1 region.

(c) Low safety 2 region.

4.5 In-Region DWA

For each region, a candidate velocity vectors are selected according to the

following objective function:

$$\begin{aligned} &\alpha(1) \cdot \text{heading}(v, w) + \alpha(2) \cdot \text{dist}(v, w) + \alpha(3) \cdot \text{vel}(v, w) \\ &+ \alpha(4) \cdot \text{margin}(v, w) \end{aligned} \quad (4.2)$$

It is almost the same as Eq. (3.1), except the modification of $\text{heading}(v, w)$ and the addition of $\text{margin}(v, w)$.

Before describing these objective functions, the definitions of several significant terms are discussed as follows.

- $ITVi(v, w)$: It depicts the index of the interval which contains the velocity vector (v, w) . In DWA*, the velocity space is divided as 181 intervals and

$ITVi(v, w)$ is computed as follows:

$$ITVi(v, w) = \left\lfloor \text{atan2}(v, w) \times \frac{\pi}{180} + 0.5 \right\rfloor \quad (4.3)$$

- $GoalITVi$: It depicts the index of the interval which contains (v_g, w_g) , and the trajectory of (v_g, w_g) passes the goal location. If the goal location related the robot coordinate is (x_g, y_g) , $GoalITVi$ is computed as follows:

$$GoalITVi = \left\lfloor \text{atan2}(x_g^2 + y_g^2, 2y_g) \times \frac{\pi}{180} + 0.5 \right\rfloor \quad (4.4)$$

- $ITVr(V), ITVI(V)$: Index of the intervals on the border of region V , here $ITVr(V) \leq ITVI(V)$. The width of V can be computed as $|ITVr(V) - ITVI(V)|$.

- $ITVg(V)$: It represents the **target interval** of region V . This term is used to compute $\text{heading}(v, w)$. For different types of regions, $ITVg(V)$ is designed as follows:

- $HSWR: \text{if } (|ITVg - ITVr| < |ITVg - ITVI|)$

$$ITVg = ITVr;$$

else

$$ITVg = ITVI;$$

- $HSNR: ITVg = \frac{ITVr + ITVI}{2}$

- $HSGR: ITVg = Goal_ITV$

- *LS1: If (the obstacles are on the right side)*

$$ITVg = ITVl;$$

else

$$ITVg = ITVr;$$

- *LS2: $ITVg = \frac{ITVr + ITVl}{2}$*

In DWA*, *heading*(*v*, *w*) is modified as follows:

heading(*v*, *w*)

{

V = The region contains ITVi(*v*, *w*);

(4.5)

return $1 - \frac{|ITVi(v, w) - ITVg(V)|}{|ITVr(V) - ITVl(V)|};$

}

Furthermore, a new objective function *margin*(*v*, *w*) is added to prevent the robot to graze obstacles. Hence, the collisions due to sensor noise and control error can be reduced.

margin(*v*, *w*)

{

V = The region contains ITVi(*v*, *w*);

(4.6)

return $\frac{|ITVi(v, w) - \frac{ITVr(V) + ITVl(V)}{2}|}{|ITVr(V) - ITVl(V)|};$

}

4.6 Look-Ahead Verification

After the candidate velocities are derived, a projected robot position is predicted as a new node for each candidate velocity. Given the present position(*x*, *y*, *θ*) and a candidate velocity(*v*, *w*), the new position is computed as follows:

$$\begin{aligned}
x' &= x + v \cdot \Delta t \cdot \cos(\theta + \frac{w \cdot \Delta t}{2}) \\
y' &= y + v \cdot \Delta t \cdot \sin(\theta + \frac{w \cdot \Delta t}{2}) \\
\theta' &= \theta + w \cdot \Delta t
\end{aligned} \tag{4.7}$$

where Δt depicts the predicted time length. The larger Δt is selected, the wider area can be explored by the look-ahead verification with the same search depth. However, if Δt is too large, the robot may be not able to find relatively narrow doors.

As mentioned in [Section 2.1](#), A* designs a cost function $f(n)$ for each node. The function represents the estimated total cost of getting to goal through node n . At the beginning of each iteration, the node with the minimum $f(n)$ is selected as the robot's position, and steps in [Section 4.2-4.5](#) are repeated. In this thesis, the robot is wished to reach the goal in minimum time. Hence, the earliest version of the cost function is designed as follows:

$$\begin{aligned}
f(n) &= g(start, n) + h(n, goal) \\
g(n_1, n_2) &= \Delta t \cdot (depth(n_2) - depth(n_1)) \\
h(n_1, n_2) &= \frac{dist(n_1, n_2)}{v_{max}}
\end{aligned} \tag{4.8}$$

Where $g(n)$ equals the predicted time length multiplying the depth of node n , that is, the actual time cost of moving to node n , and $h(n_1, n_2)$ is the straight distance between node n and the goal location divided by the maximum velocity of the robot.

$h(n_1, n_2)$ is *admissible*, that is, never over-estimated, as discussed in *appendix*.

Though the first version of cost function helps solving local minima problem, sometimes it causes the robot to accelerate or to decelerate rapidly. Hence, for

smoother navigation, the $g(n)$ term is modified as follows:

$$g(n_1, n_2) = \sum_{n=n_2}^{n_1} c(n_p, n)$$

$$c(n_1, n_2) = \Delta t \cdot (\text{depth}(n_2) - \text{depth}(n_1))$$

$$+ \rho_1 \cdot |v(n_2) - v(n_1)| + \rho_2 \cdot |w(n_2) - w(n_1)| \quad (4.9)$$

where $v(n)$ and $w(n)$ represent the translational velocity and rotational velocity of robot in node n , and n_p represents the parent node of n . In the modified version of cost function, the accumulation of velocity variation is also considered as cost. Thus, the robot can generate smoother trajectories.

4.7 Implementation on Laser-Based Mobile Robots

For robots equipped with laser range finders, a better heuristic function $h(n)$ can be applied to improve the performance of DWA*. Here the technique in ND is applied. In ND, the raw laser data is utilized to find **gaps** in the environment. A gap is a discontinuity in laser reading histogram and depicts a potentially passable gate for the robot. When the goal location is not in the region covered by laser beam, it is evident that the robot must pass one of the gates to reach goal. For this reason, a new version of heuristic function is designed as follows:

If (n is indoor for g)

$$h_g(n, g) = \min_{g \in G} \left(\frac{\text{dist}(n, g) + \text{dist}(g, \text{goal})}{v_{\max}} \right);$$

else

$$h_g(n, g) = \frac{\text{dist}(n, \text{goal})}{v_{\max}};$$

$$h(n, \text{goal}) = \min_{g \in G} (h_g(n, g))$$

(4.10)

where G is the union of all passable gates in the environment found by ND,

$h(n, \text{goal})$ is the minimum cost of reaching the goal through one of these gates,

and $h_g(n, g)$ means the estimated time cost of reaching the goal through gate g . Note

that if the expanded node n locates out of the gate g (the verification method is

showed in **appendix**), it is assumed that the robot can reach the goal from

node n without visiting gate g . In this case, $h_g(n, g)$ is still defined

as $\text{dist}(n, \text{goal})$ divided by v_{\max} .

For cost function 1 and cost function 2, the information utilizing rate is

proportional to the maximum search depth. On the other hand, with the help of door

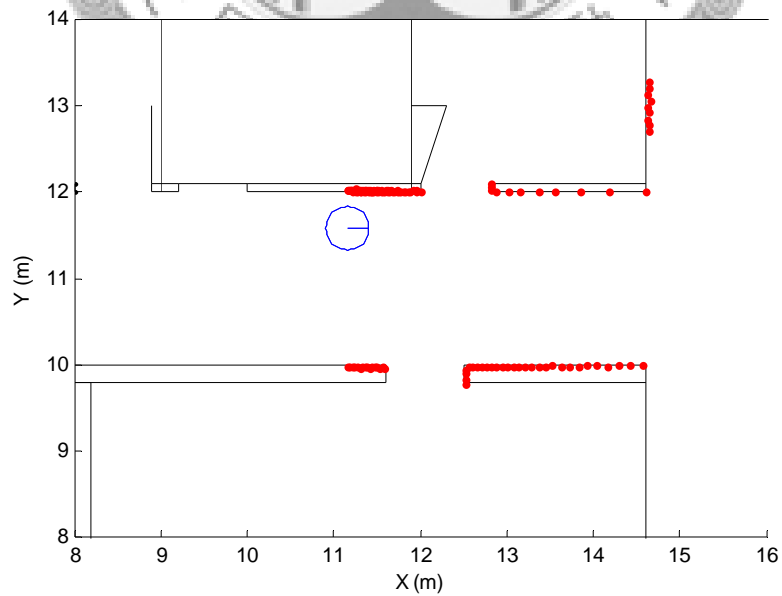
detection, the information of environment far from the robot can also be utilized by

cost function 3 to make a better estimation. Therefore, DWA* can overcome local

minima problem with less search depth.

4.8 Implementation on Sonar-Based Mobile Robots

On sonar-based mobile robots, collisions may happen because of the limited sensor system. Figure 4.5 compares the scenarios by using sonar reading and laser reading. Figure 4.5(b) shows obstacles detected by laser range finder and Figure 4.5(c) shows obstacles detected by sonar. For compensating the blind corners of sonar sensors, a First-In-First-Out obstacle queue is built to record the last 5 sonar measurements (In the simulation cases, it equals $16 \times 5 = 80$ points). However, compared with laser data, the accuracy of sonar data is still very poor. It can seriously affect the performance of DWA*. For example, in Figure 4.5(b), the door on upper right seems impassable for the robot. If the goal location is in the upper right room, robot will not turn left to pass the door but turn right to take a long route.



(a)

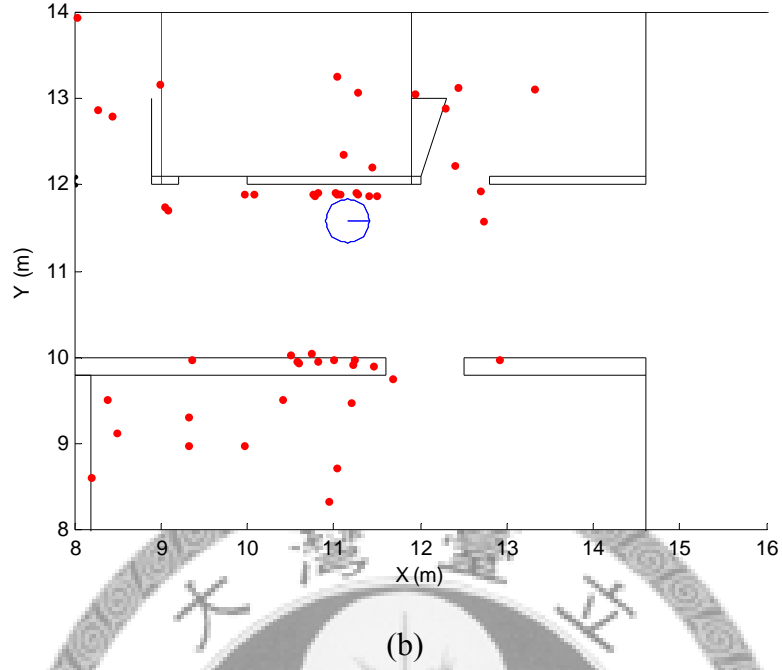


Figure 4.5: Comparison of laser and sonar data.

The solid line circle means the robot's position. Red points depict sensor readings.

(a) Obstacles detected by laser.

(b) Obstacles detected by sonar..

For increasing the safety of sonar-based robots, the *bug algorithm* [15: Choset et al. 2005] is added. That is, when there are any obstacles in the robot's security region, it will perform wall-following actions. It is designed as follows:

$$v = v_{\max} \cdot \frac{d}{d_{\text{save}}} \cdot \frac{|\theta|}{\frac{\pi}{2}} \quad (4.11)$$

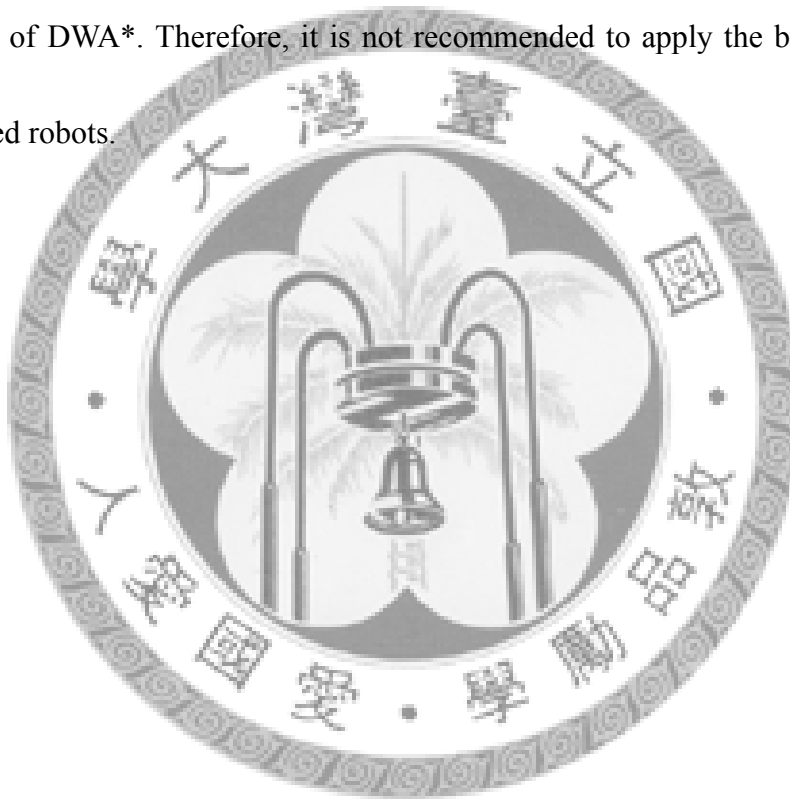
$$w = w_{\max} \cdot \left(\frac{\theta - \text{sign}(\theta) \cdot \frac{\pi}{2}}{\frac{\pi}{2}} \right) \quad (4.12)$$

where v_{\max} and w_{\max} are the maximum translational and rotational velocities of the robot. d_{save} is a constant depicts the size of security region. θ and d represent the direction and distance of the nearest obstacle in the robot's coordinate. The robot

switches to bug algorithm if $d < d_{save}$. In this mode, the robot always moves toward the direction parallel to the nearest obstacle and decelerates when it is close to obstacles.

With the help of the bug algorithm, the safety of sonar-based robots can be raised substantially. Nevertheless, it is not an ideal algorithm for high-speed navigation.

When sensory information is good enough, adding the bug algorithm may lower the performance of DWA*. Therefore, it is not recommended to apply the bug algorithm on laser-based robots.



Chapter 5

Simulation Results

In this chapter, simulation results of DWA* on different types of robots are demonstrated. In [Section 5.1](#), the simulation platform and the format of simulation data are introduced. The results of DWA* on laser-based mobile robots are shown in [Section 5.2](#) and the results of DWA* on sonar-based mobile robots are shown in [Section 5.3](#).

5.1 Simulation Platform

The simulation platform is the simulator developed by ActivMedia Robotics, LLC, for Pioneer 3™ robot. The Pioneer 3 robot is a circular differential robot, its maximum velocity is 0.5m/s. The maximum computational time of DWA* is limited in 0.25 seconds. If look-ahead verification is not able to find goal state in 0.25 second, the node with maximum depth in the present tree is selected as the goal state. So there is

no need to worry that the addition of look-ahead verification may lower the robot's reactive speed.

For each case, the following four terms are computed to compare different algorithms.

- **Run time (RT) (sec)**: Total run time of the simulation.
- **Average velocity (AV) (m/s)**: The average robot's translational velocity. It is hoped that the robot can always navigate in maximum velocity.
- **Average translational acceleration (ATA) (m/s^2)**: Average absolute value of translational velocity variation.
- **Average rotational acceleration (ARA) (rad/s^2)**: Average absolute value of rotational velocity variation. **ATA** and **ARA** reflect the trajectory smoothness.

In the following figures, the four terms are represented in this form: (**RT**, **AV**, **ATA**, **ARA**). Note that DWA* with search depth n is named as DWA*- n .

5.2 Simulation Results for Laser-Based Robot

In following simulations, the Pioneer 3 robot is equipped with a laser range finder which can scan the front 180 degrees. Since the laser range finder provides very dense and accurate data about the environment, the characteristic of each algorithm can be demonstrated clearly.

Three simulation results are shown in this section. The first result shows the effects of region analysis and look-ahead verification, the second and the third results are presented to compare the differences of the three cost functions shown as Eq. (4.8), Eq. (4.9), and Eq. (4.10). Where Eq. (4.8) is the simplest cost function defined by intuition. Eq. (4.9) is a modified cost function which takes care of trajectory smoothness. Eq. (4.10) is a modified version Eq. (4.9), ND technique is applied to design a more reasonable heuristic function.

Figure 5.1 shows the first simulation result. The effects of region analysis and look-ahead verification are shown in this case. In this situation, the robot has to pass through two doors to reach the goal, and the original DWA gets stuck in local minima and cannot pass through the first door. Figure 5.1(c)-(f) are results of DWA* with cost function 1 (see Eq. (4.8)). The DWA*-1 algorithm can pass through the first door because improper motions commands are filtered by the region analysis. However, when the robot approaches the second door, the cost values of turning left to pass through the second door and turning right are almost the same. Hence, in this scenario, the DWA*-1 algorithm chooses to turn right and gets stuck. In order to overcome this problem, the DWA*-5 algorithm can be used to move back and pass through the second door. However, it may spend more time on generating this trajectory. Figure 5.1(g)-(i) shows the results of using DWA* with cost function 2, namely (see Eq. (4.9)). Figure

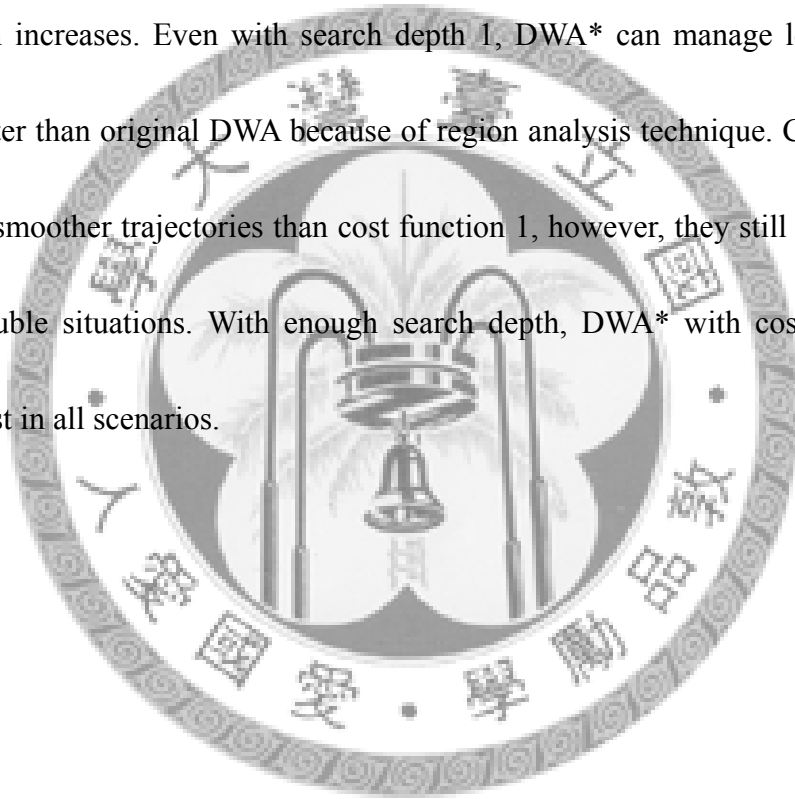
10(g) shows that cost function 2 affects the exploring ability of DWA*. Nevertheless, with enough search depth, DWA* with cost function 2 can perform much better than that with cost function 1. As shown in Figure 5.1(i), with the consideration of trajectory smoothness, the robot chooses to pass the second door and avoids vibrating in front of the door, as shown in Figure 5.1(c) and (e).

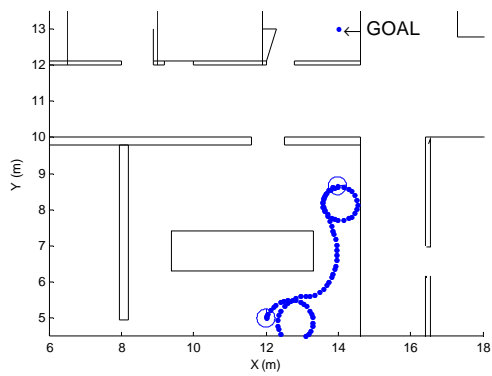
Figure 5.2 presents the effect of cost function 3. In this case, the robot needs to get rid of a typical local trap to reach goal. It can be seen that the original DWA gets stuck in the local minima easily. And DWA* needs a high search depth to overcome this local trap using cost function 1 or 2. On the other hand, with cost function 3, the robot can detect narrow doors and move toward them earlier. As shown in Figure 5.2(g), with cost function 3, the robot can be aware of the narrow door at the beginning and achieve a smoother navigation than that shown in Figure 5.2(e).

Figure 5.3 shows the simulation results the third scenario. In this case, the robot needs to go through a narrow and winding passage. The scenario is difficult for many local methods. In Figure 5.3(a), it can be seen that the original DWA falls soon from the starting location. On the other hand, DWA* can successfully complete the task because of region analysis and look-ahead verification. Comparing Figure 5.3(c) and Figure 5.3(e), although DWA*-5 with cost function 1 is able to overcome local minima problem, DWA*-5 with cost function 2 achieves faster and smoother navigation

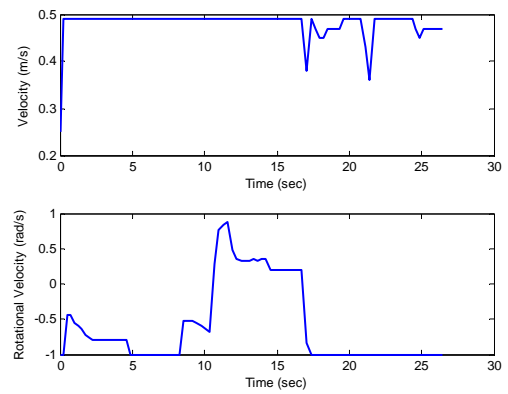
because of the consideration of trajectory smoothness. Figure 5.3(g) shows the result of DWA*-5 with cost function 3. With the modified heuristic function, DWA* can correctly predict the direction of exit from the beginning and perform even better than that shown in Figure 5.3(e).

In the above simulation results, we can find that DWA* perform better as the search depth increases. Even with search depth 1, DWA* can manage local minima problem better than original DWA because of region analysis technique. Cost function 2 generates smoother trajectories than cost function 1, however, they still can be stuck in some trouble situations. With enough search depth, DWA* with cost function 3 performs best in all scenarios.

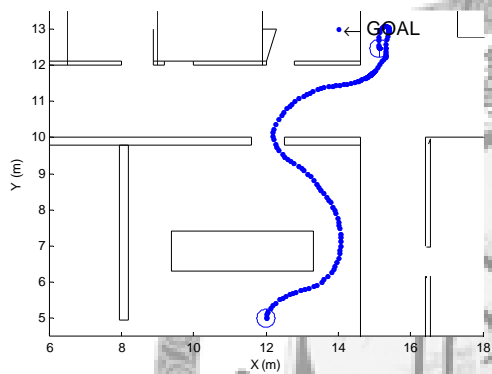




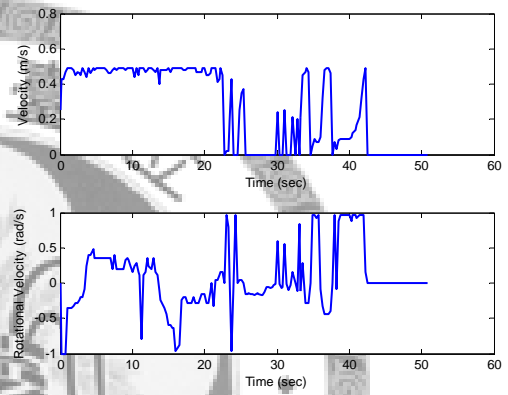
(a)



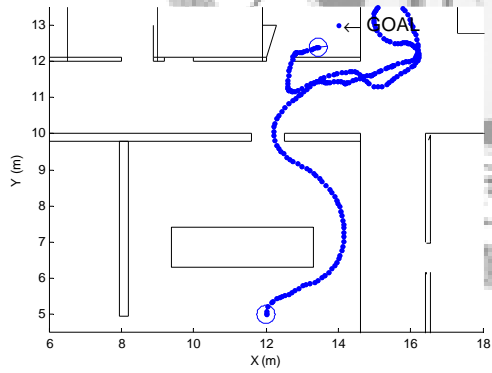
(b)



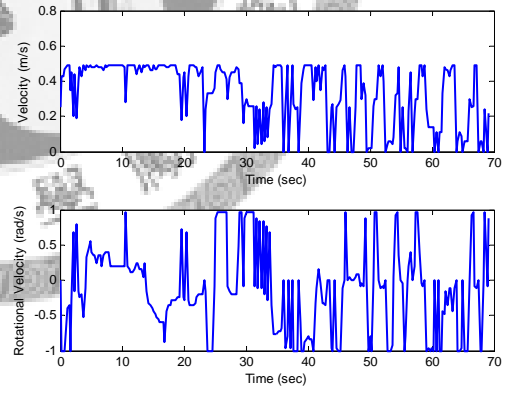
(c)



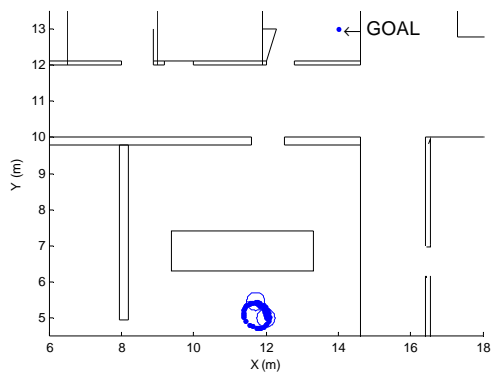
(d)



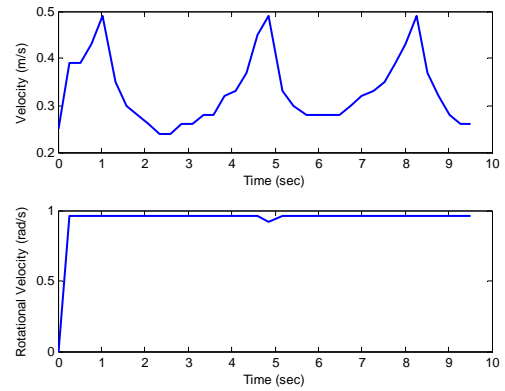
(e)



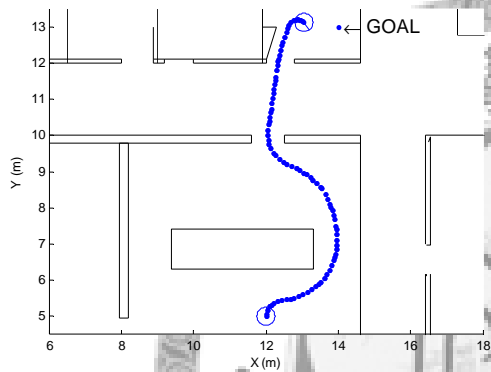
(f)



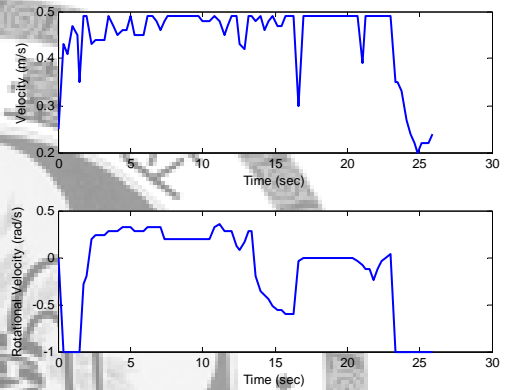
(g)



(h)



(i)



(j)

Figure 5.1: Simulation result 1.

(a) DWA result: (26.42, 0.48, 0.03, 0.19)

(b) Velocity-time plot of DWA result.

(c) DWA*-1, cost1 result: (50.77, 0.27, 0.16, 0.59)

(d) Velocity-time plot of DWA*-1, cost1 result.

(e) DWA*-5, cost1 result: (69.14, 0.33, 0.34, 1.27)

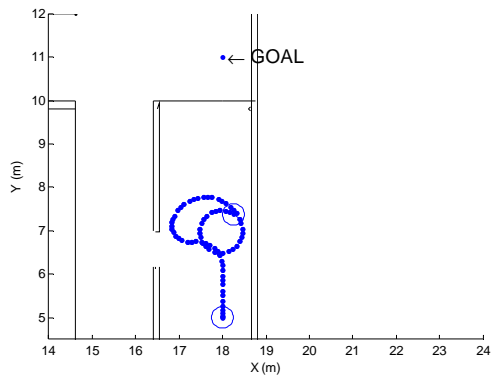
(f) Velocity-time plot of DWA*-5, cost1 result.

(g) DWA*-1, cost2 result: (9.52, 0.33, 0.14, 0.10)

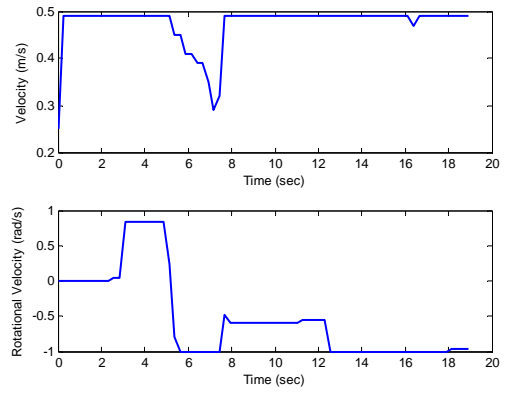
(h) Velocity-time plot of DWA*-1, cost2 result.

(i) DWA*-5, cost2 result: (25.91, 0.44, 0.08, 0.22)

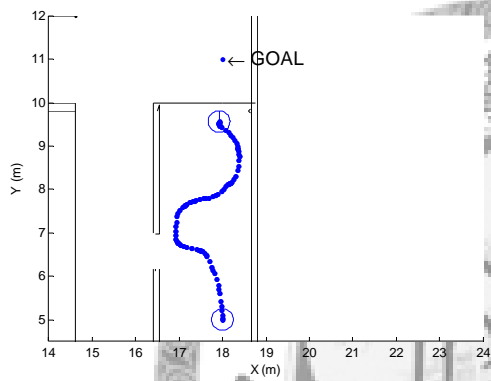
(j) Velocity-time plot of DWA*-5, cost2 result.



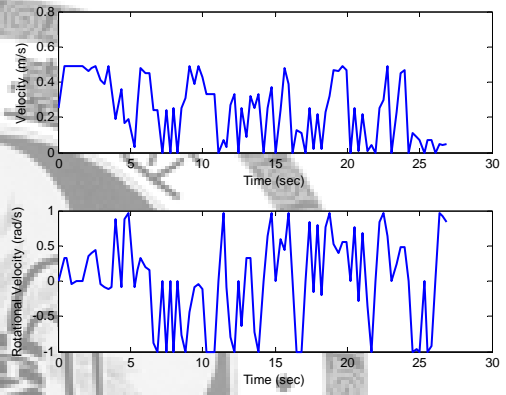
(a)



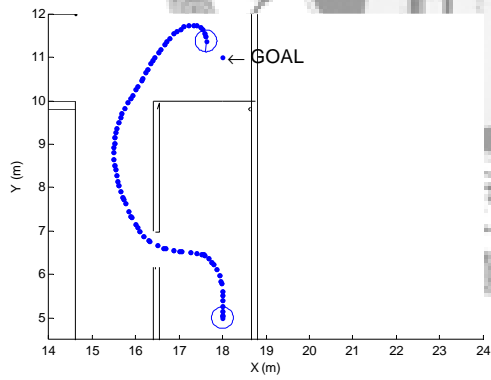
(b)



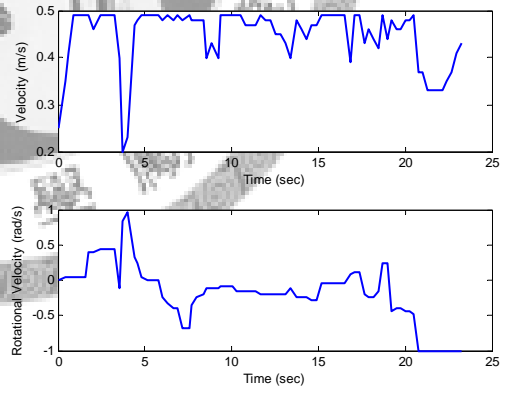
(c)



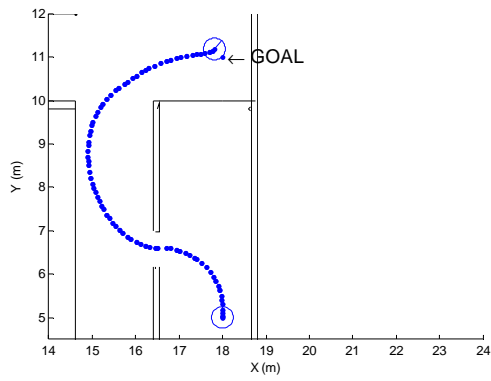
(d)



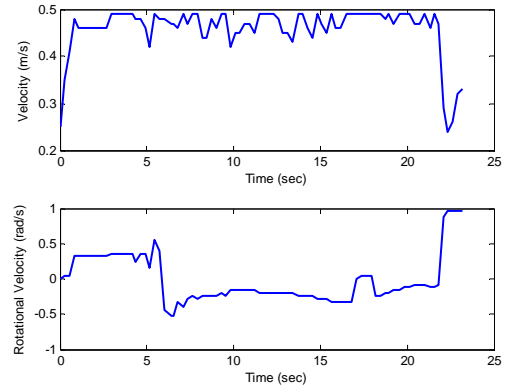
(e)



(f)



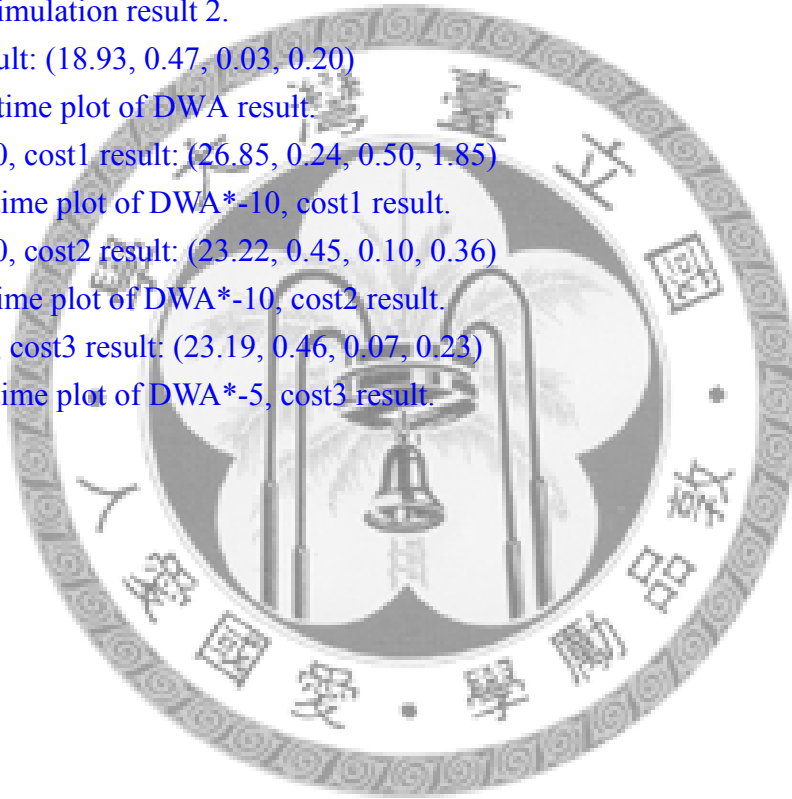
(g)

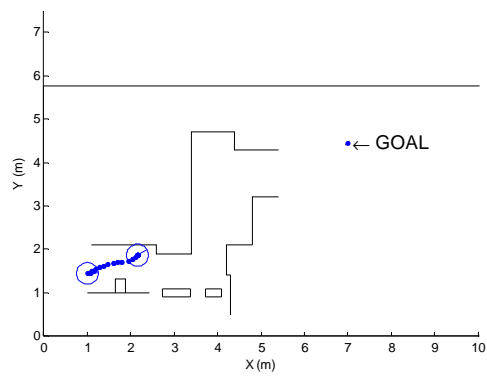


(h)

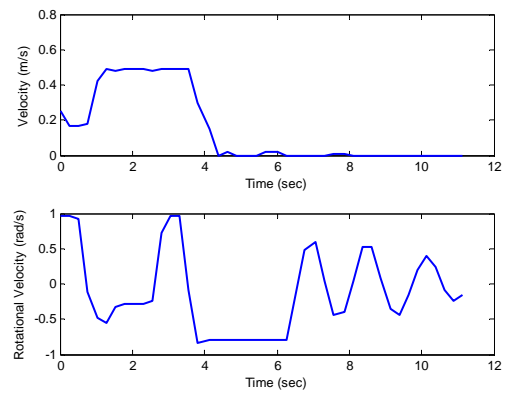
Figure 5.2: Simulation result 2.

- (a) DWA result: (18.93, 0.47, 0.03, 0.20)
- (b) Velocity-time plot of DWA result.
- (c) DWA*-10, cost1 result: (26.85, 0.24, 0.50, 1.85)
- (d) Velocity-time plot of DWA*-10, cost1 result.
- (e) DWA*-10, cost2 result: (23.22, 0.45, 0.10, 0.36)
- (f) Velocity-time plot of DWA*-10, cost2 result.
- (g) DWA*-5, cost3 result: (23.19, 0.46, 0.07, 0.23)
- (h) Velocity-time plot of DWA*-5, cost3 result.

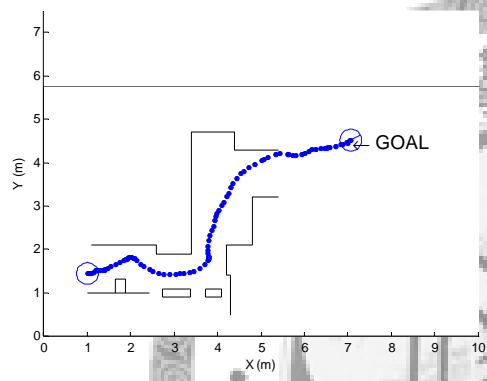




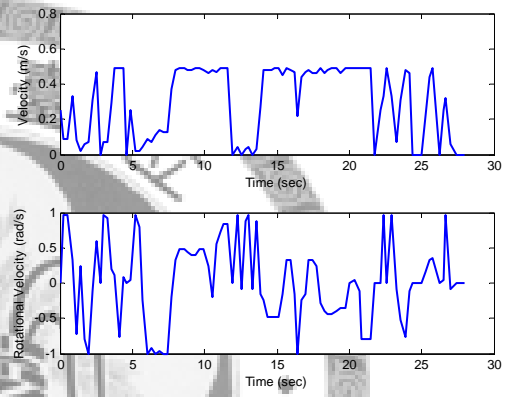
(a)



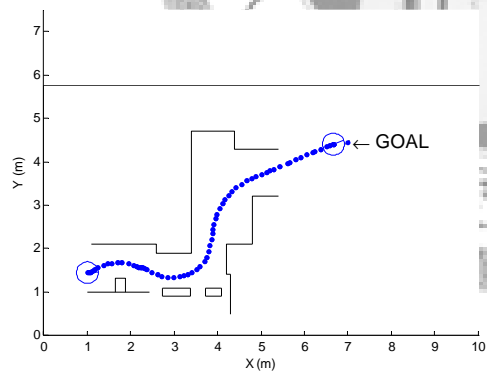
(b)



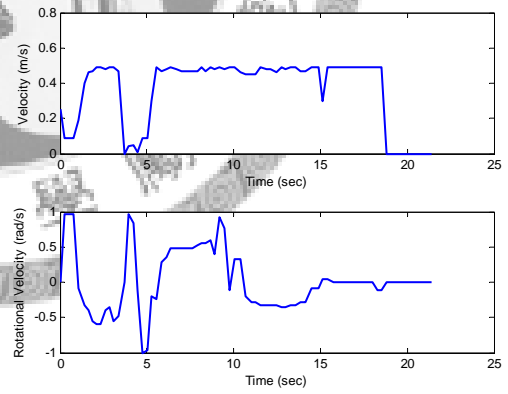
(c)



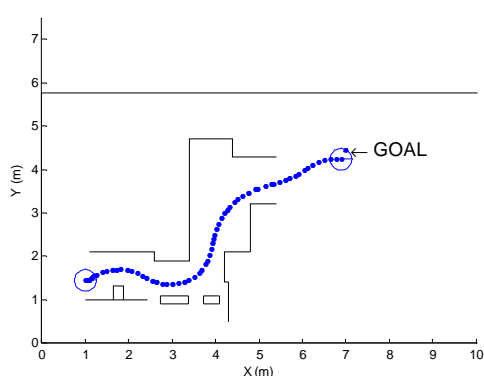
(d)



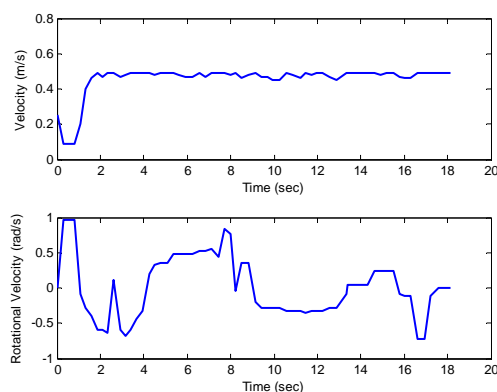
(e)



(f)



(g)



(h)

Figure 5.3: Simulation result 3.

(a) DWA result: (11.13, 0.15, 0.09, 0.98)

(b) Velocity-time plot of DWA result.

(c) DWA*-5, cost1 result: (27.93, 0.30, 0.34, 1.46)

(d) Velocity-time plot of DWA*-5, cost1 result.

(e) DWA*-5, cost2 result: (21.42, 0.36, 0.13, 0.54)

(f) Velocity-time plot of DWA*-5, cost2 result.

(g) DWA*-5, cost3 result: (18.13, 0.45, 0.06, 0.59)

(h) Velocity-time plot of DWA*-5, cost3 result.

5.3 Simulation Results for Sonar-Based Robot

In following cases, the Pioneer 3 robot is equipped with 16 sonar sensors. As mentioned in [Section 4.8](#), an obstacle queue with a maximum size of 80 is maintained.

Since the ND technique is specifically designed for laser data, the cost function 2 is used for the sonar-based robots. The velocity of the Pioneer 3 robot is limited to 0.4m/s for safety reason.

Two simulation cases are shown in this section. The first case is in a narrow place, this case presents the effect of bug algorithm. The second case is in a situation with two narrow doors, this case shows that DWA* is able to pass narrow doors even

with relatively poor sensory information.

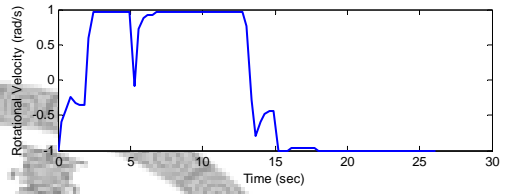
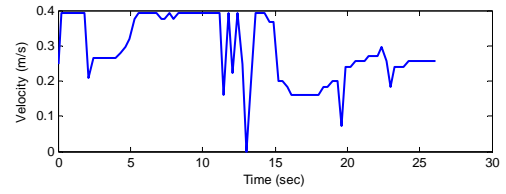
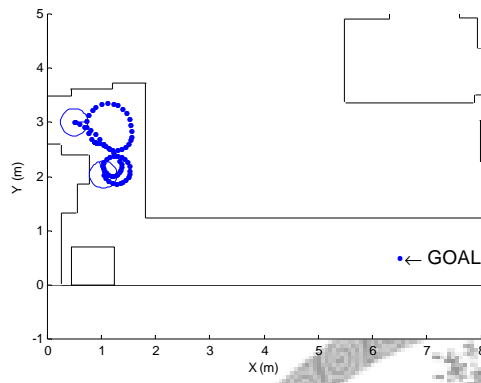
The fourth simulation result is shown in [Figure 5.4](#). The environment in this case is similar to the simulation case 3. [Figure 5.4\(a\)](#) and [Figure 5.4\(e\)](#) show that the original DWA has difficulty to avoid local minima whether laser or sonar data is utilized. The result of DWA*-8 on laser-based robot is shown in [Figure 5.4\(g\)](#).

Although the robot selects improper motion command for several times due to wrong sensory information, collisions are avoided by switching to the bug algorithm. As a result, DWA*-8 can overcome the complex situation using sonar data.

The fifth simulation result is presented in [Figure 5.5](#). It is like the situation in simulation result 1. As shown in case 4, with original DWA, either laser-based or sonar-based robots are not able to escape from the local minima. On the other hand, laser-based robot with DWA*-8 can reach the goal with high speed and smoothness. Sonar-based robot with DWA*-8 meets difficulty to pass through the second door because of poor sensory information (as depicted in [Figure 4.5](#)), but it can still go back and arrive goal.

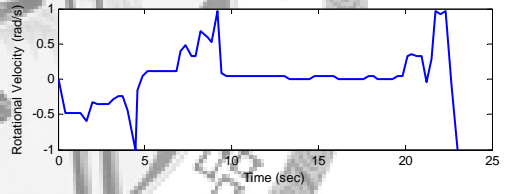
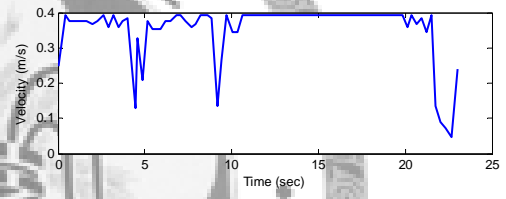
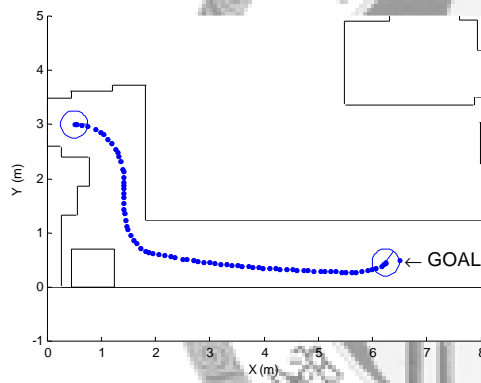
As a result, compared to laser-based robot, sonar-based robot has difficulty to navigate in narrow places or pass narrow doors using pure DWA* because of the poor sensory information. However, with the addition of bug algorithm, the robot can apply DWA* to move on proper directions in broad areas and switch to bug algorithm to pass

narrow places safely.



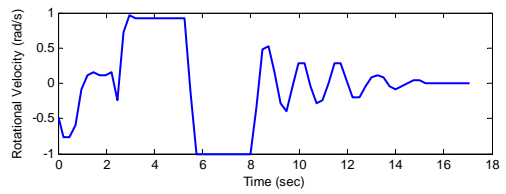
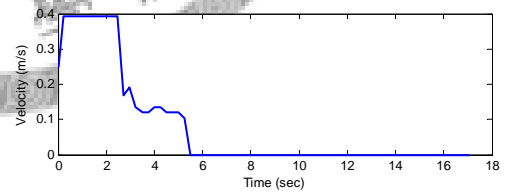
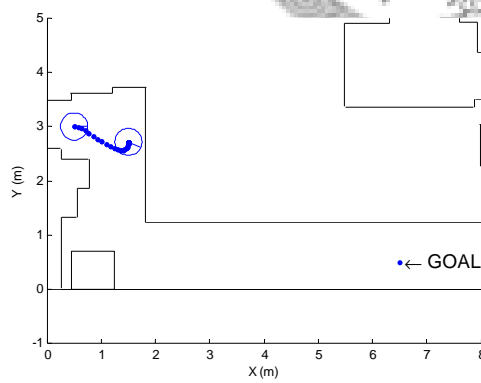
(a)

(b)



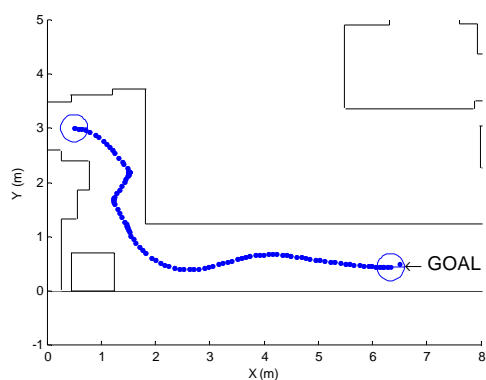
(c)

(d)

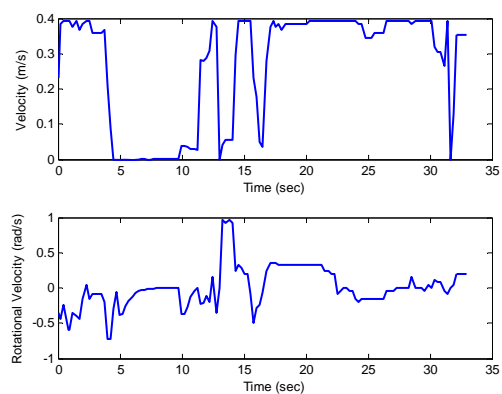


(e)

(f)



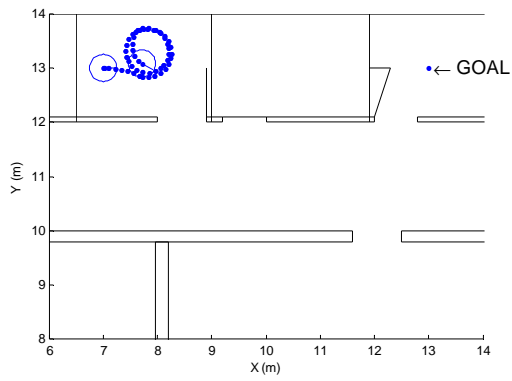
(g)



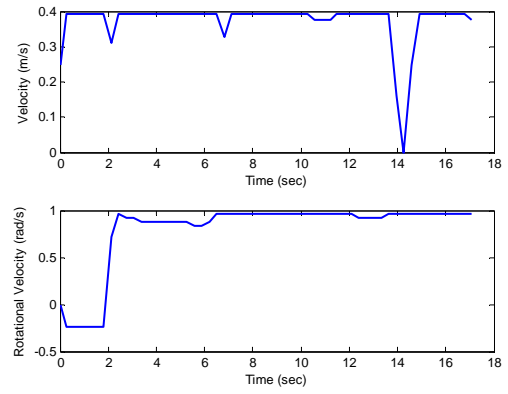
(h)

Figure 5.4: Simulation result 4.

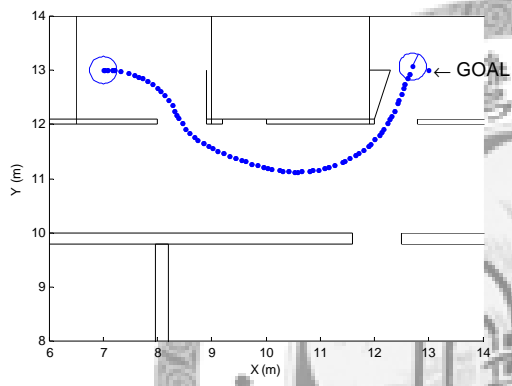
- (a) DWA, laser result: (26.08, 0.29, 0.11, 0.27)
- (b) Velocity-time plot of DWA laser result.
- (c) DWA*-8, cost2, laser result: (22.98, 0.35, 0.11, 0.43)
- (d) Velocity-time plot of DWA*-8, cost2, laser result.
- (e) DWA*, sonar result: (17.05, 0.08, 0.04, 0.59)
- (f) Velocity-time plot of DWA* sonar result.
- (g) DWA*-8, cost2, sonar result: (32.91, 0.26, 0.11, 0.36)
- (h) Velocity-time plot of DWA*-8, cost2, sonar result.



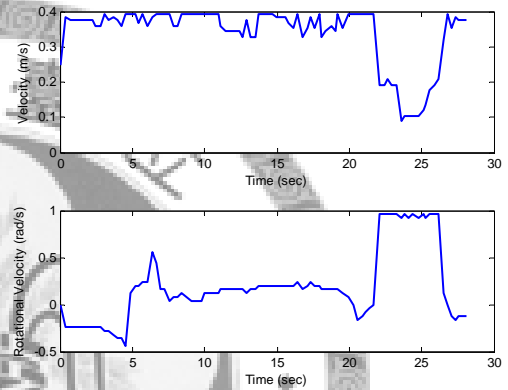
(a)



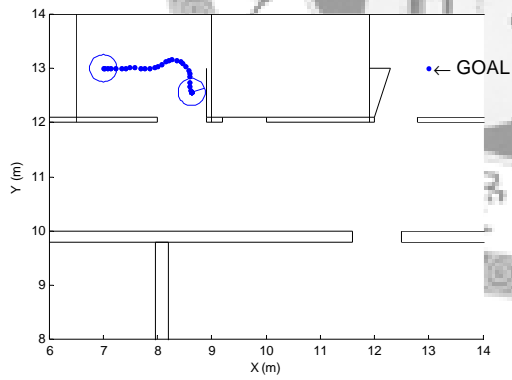
(b)



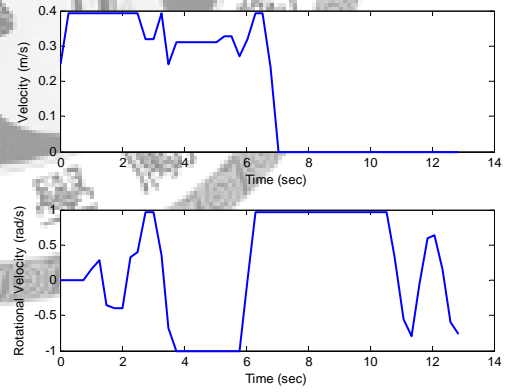
(c)



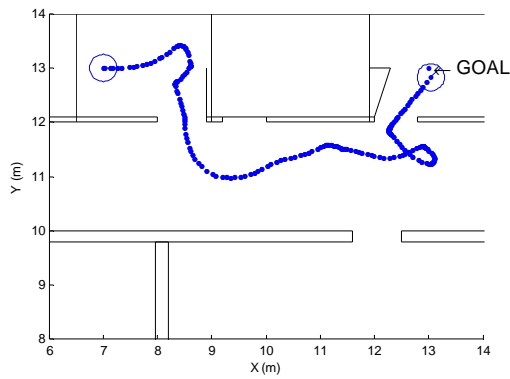
(d)



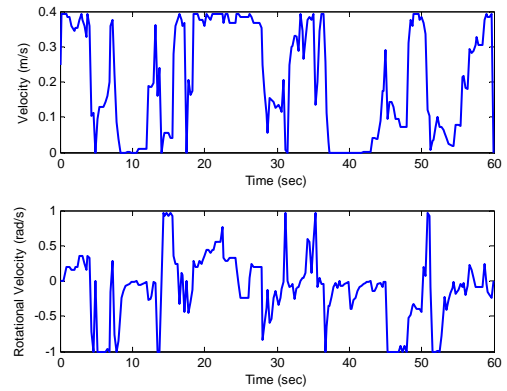
(e)



(f)



(g)



(h)

Figure 5.5: Simulation result 5.

- (a) DWA, laser result: (17.08, 0.37, 0.07, 0.10)
- (b) Velocity-time plot of DWA laser result.
- (c) DWA*-8, cost2, laser result: (22.08, 0.37, 0.07, 0.17)
- (d) Velocity-time plot of DWA*-8, cost1, laser result.
- (e) DWA, sonar result: (12.84, 0.18, 0.08, 0.82)
- (f) Velocity-time plot of DWA sonar result.
- (g) DWA*-8, cost2, sonar result: (59.98, 0.20, 0.15, 0.67)
- (h) Velocity-time plot of DWA*-8, cost2, sonar result.

Chapter 6

Experimental Results

In this chapter, experimental results of DWA* on real Pioneer 3™ robot are demonstrated. [Section 6.1](#) shows the results on laser-based robot and [Section 6.2](#) shows the results on sonar-based robot.

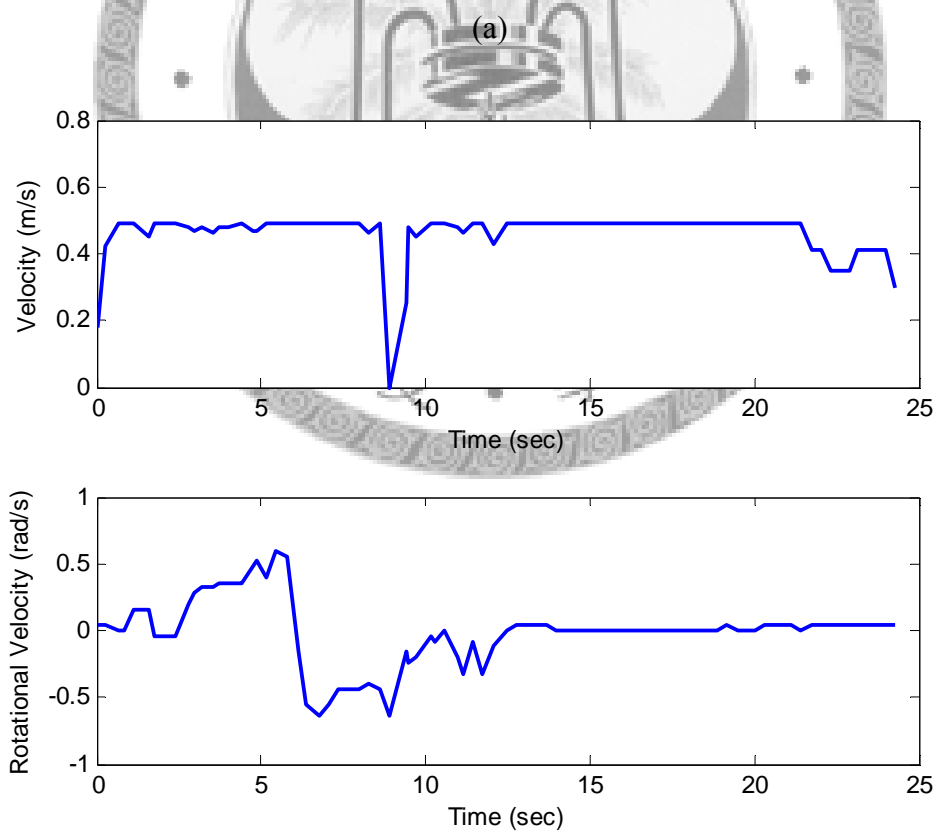
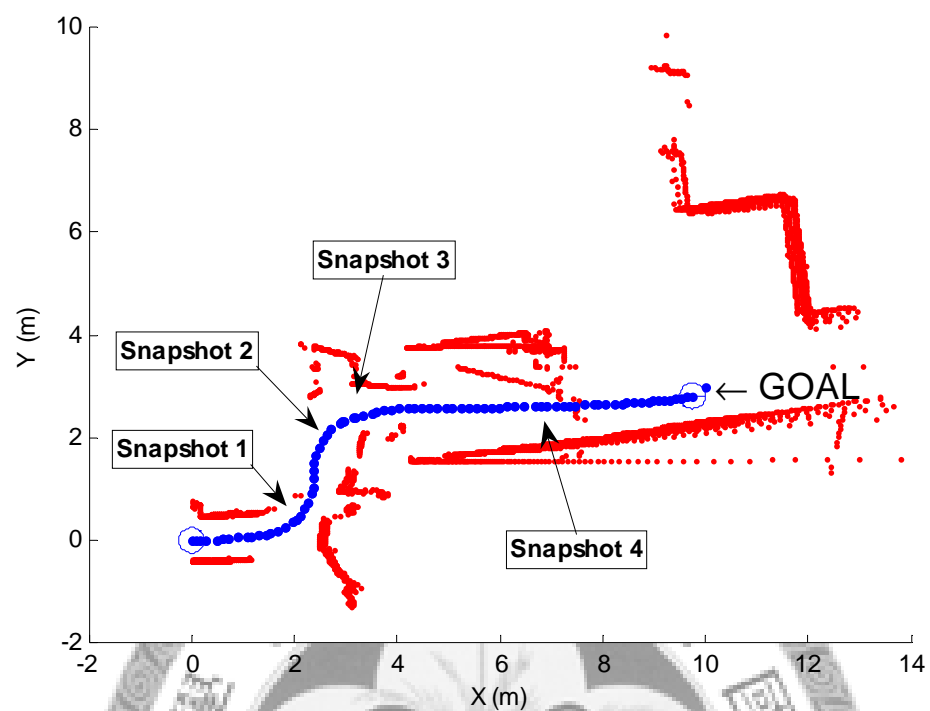
6.1 Experimental Results for Laser-Based Robot

In this section, the experimental results of DWA* on laser-based robot are presented. Experimental result 1 is shown in [Figure 6.1](#). Although this environment is similar as that in [Figure 5.3](#), the obstacle distribution in real environment is much messier than that in simulation. However, DWA* is still able to drive the robot to the goal position in high speed and smoothness.

Experimental result 2 is shown in [Figure 6.2](#); the merits of DWA* are strongly demonstrated in this experiment. In this scenario, the robot has to pass a broad area and

then go through a narrow passage. Because of region analysis, DWA* can detect the entrance of the narrow passage when the robot is in the position shown in [Figure 6.2\(c\)](#). Therefore, the robot can make a smooth turn to go into the passage. With the help of look-ahead verification, DWA* can select control commands by evaluating their consequences after several steps, so the robot can pass the narrow passage without deceleration, as shown in [Figure 6.2\(d\)](#), [\(e\)](#), and [\(f\)](#).





(b)



(c)



(d)



(e)



(f)

Figure 6.1: Experimental result 1.

(a) DWA*-8, cost3, laser result: (24.27, 0.46, 0.10, 0.22)

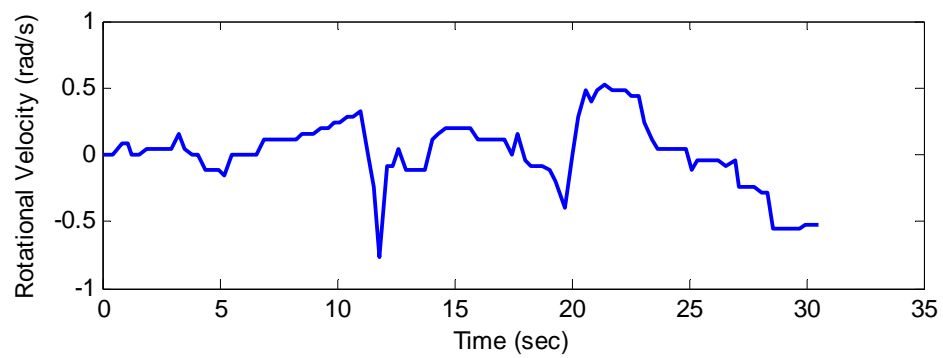
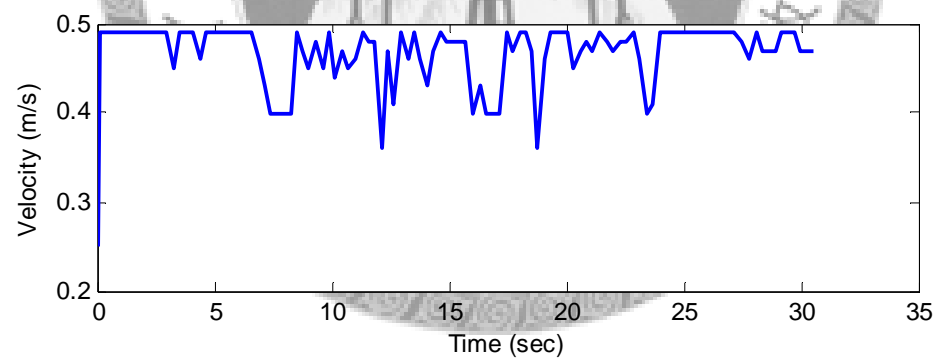
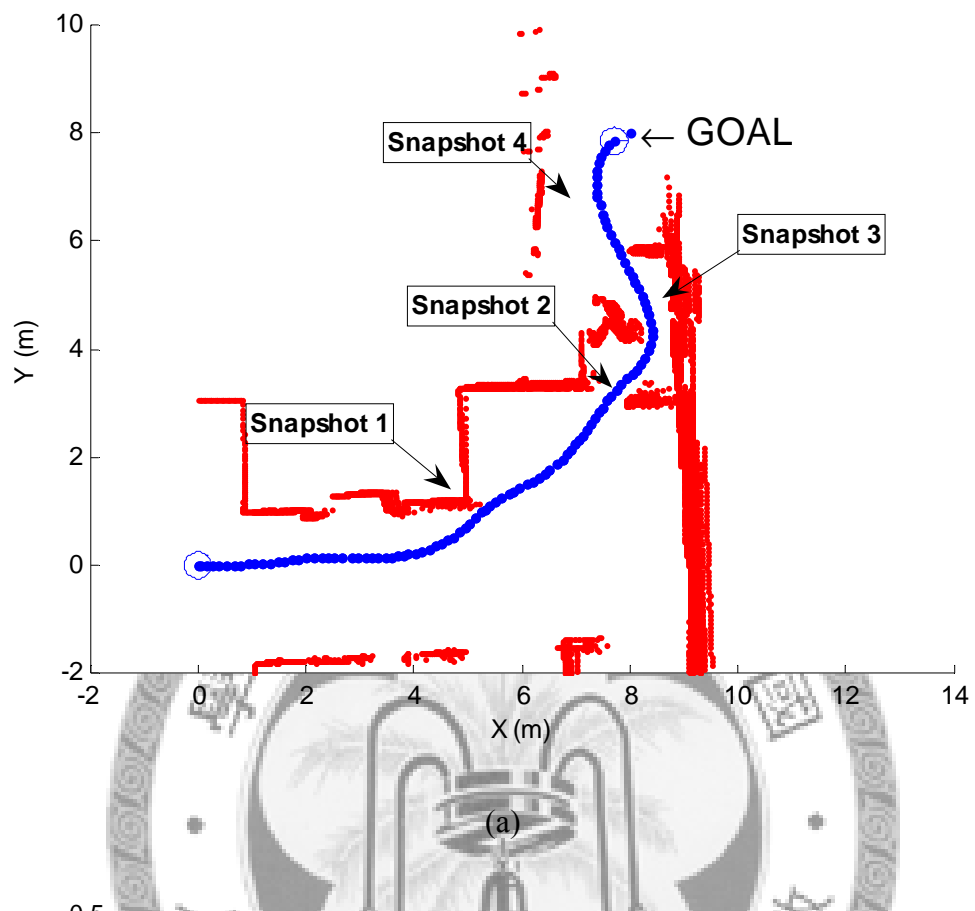
(b) Velocity-time plot of DWA*-8, cost3, laser result.

(c) Snapshot 1.

(d) Snapshot 2.

(e) Snapshot 3.

(f) Snapshot 4.



(b)



(c)



(d)



(e)



(f)

Figure 6.2: Experimental result 2.

(a) DWA*-8, cost3, laser result: (30.52, 0.47, 0.08, 0.22)

(b) Velocity-time plot of DWA*-8, cost3, laser result.

(c) Snapshot 1.

(d) Snapshot 2.

(e) Snapshot 3.

(f) Snapshot 4.

6.2 Experimental Results for Sonar-Based Robot

In this section, the experimental results of DWA* on sonar-based robot are presented. The real dead-reckoning system performs worse and the error rate of recorded sensory information is higher than that in simulations. Hence, the role of bug algorithm becomes more important.

Experimental result 3 is demonstrated in [Figure 6.3](#). In this case, the robot has to pass through three narrow doors, as shown in [Figure 6.3\(c\)](#), [\(e\)](#), and [\(f\)](#). Although DWA* can pass through the first and second doors smoothly, it meets problem in [Figure 6.3\(e\)](#) and [\(f\)](#). In [Figure 6.3\(e\)](#), because of the dead corners of sonar system, the robot considers that there are no obstacles and move toward the wall. In [Figure 6.3\(f\)](#), due to the error of sonar reading and dead reckoning system, DWA* computes the position of the narrow door mistakenly and move toward the wall again. However, with the help of the bug algorithm, the robot still can reach goal without collision.

Experimental result 4 is shown in [Figure 6.4](#); the scenario is the same as that in [Figure 6.1](#). Although the environment is a little broader than that in [Figure 6.3](#), it is more winding. In this case, it can be seen that DWA* is able to select correct direction for approaching the goal location. Collisions due to sensor error are avoided by using the bug algorithm, as shown in [Figure 6.4\(c\)](#), [\(e\)](#), and [\(f\)](#).



(c)



(d)



(e)



(f)

Figure 6.3: Experimental result 3.

(a) DWA*-8, cost2, sonar result: (57.71, 0.15, 0.20, 0.71)

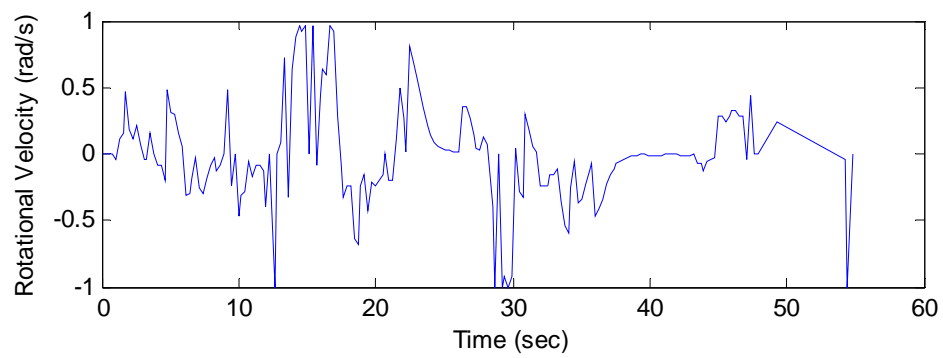
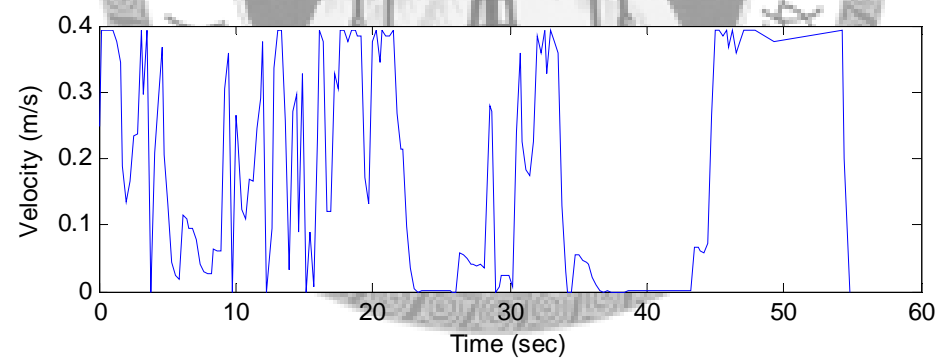
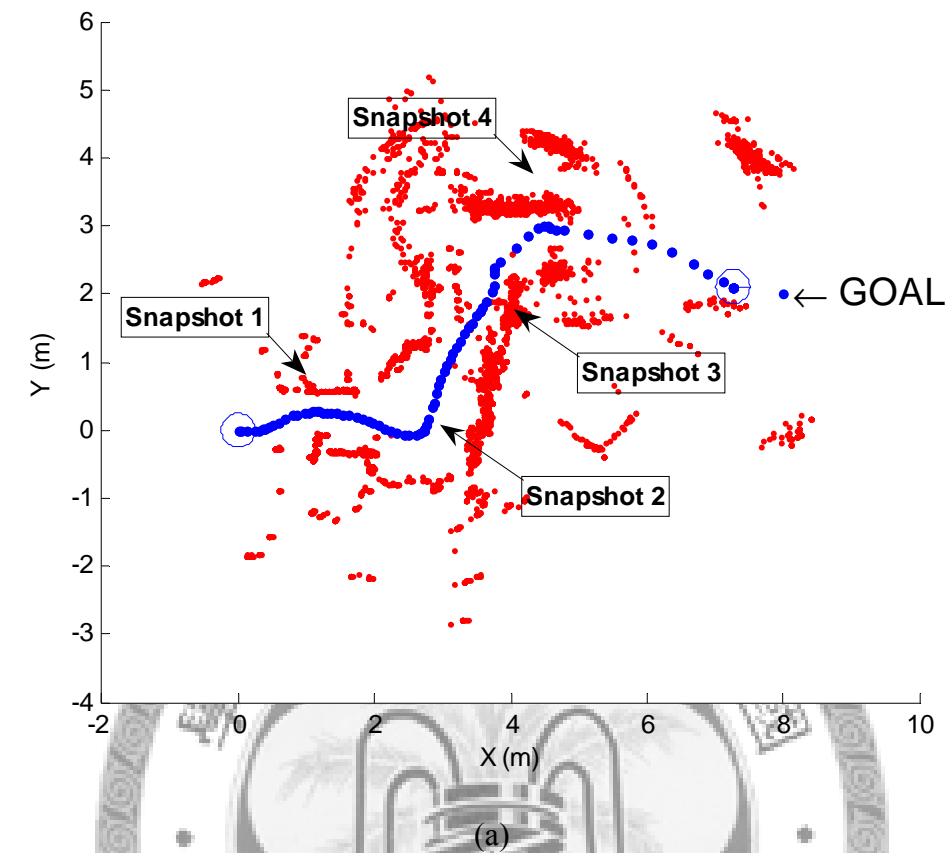
(b) Velocity-time plot of DWA*-8, cost2, sonar result.

(c) Snapshot 1.

(d) Snapshot 2.

(e) Snapshot 3.

(f) Snapshot 4.



(b)



(c)



(d)



(e)



(f)

Figure 6.4: Experimental result 4.

(a) DWA*-8, cost2, sonar result: (41.53, 0.24, 0.15, 0.58)

(b) Velocity-time plot of DWA*-8, cost2, sonar result.

(c) Snapshot 1.

(d) Snapshot 2.

(e) Snapshot 3.

(f) Snapshot 4.

Chapter 7

Discussion

In this thesis, a newly developed reactive navigation method, called DWA* is presented. With the addition of region analysis and the look-ahead verification, the DWA* algorithm can utilize environmental information effectively to achieve high-speed, smooth and local-minima-free navigation even in narrow and winding environments.

The DWA* algorithm can be divided into the following three steps: (1) constructing motion space, (2) filtering and classifying motion commands, and (3) look-ahead verification. Under this framework, velocity approaches, directional approaches and global methods are integrated systematically. Since these three steps can be modified independently, DWA* has high flexibility. For example, although DWA* in this thesis is designed for wheeled robots, after the modification of motion space construction step, it can also be applied on robot arms or legged robots. By modifying the cost function of look-ahead verification, DWA* can be designed to

satisfy various kinds of requests.

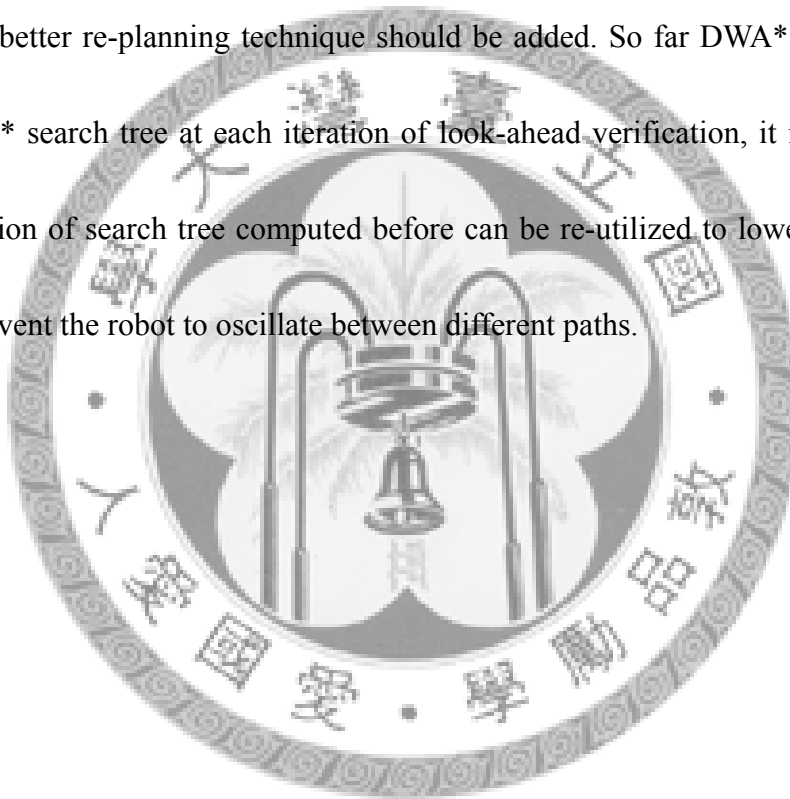
The most significant feature of DWA* is the employment of region analysis technique, that is, the part of filtering and classifying motion commands. The filtering and classifying step can be seen as branch factor reduction of the search tree in look-ahead verification; it is important for real-time implementation. Compared to the techniques in directional approaches, in this thesis, region analysis is applied on the robot motion space to find candidate velocities but not candidate directions. Therefore, DWA* can generate more reasonable velocity commands than original DWA and can navigate much smoother than directional approaches.

Though DWA* performs well in many scenarios, there are several points to be improved in the future. First, the filtering and classifying step should be designed more considerably. In this thesis, navigable regions are found by computing the discontinuities on the clearance histogram of robot trajectories, and all velocity commands out of navigable regions are filtered. However, sometimes acceptable velocity commands may also be filtered. For example, when the robot is in very narrow place, it may not be able to find navigable regions and get stuck.

Second, new cost functions should be designed for integrating with global methods and mapping techniques. Though DWA* with cost function 3 in this thesis can overcome many complex scenarios, since the map information is not recorded,

DWA* is possible to fall in scenarios which have many doors and only few of them lead to the goal. On the other hand, presently, the areas which have not been explored are considered as free space so that collisions may happen because of undetected obstacles. Mapping techniques and the concepts of uncertainty should be applied to improve this problem.

Third, better re-planning technique should be added. So far DWA* re-computes the whole A* search tree at each iteration of look-ahead verification, it is hoped that the information of search tree computed before can be re-utilized to lower computing time and prevent the robot to oscillate between different paths.



APPENDIX

A.1 A* Optimality

In A*, if $h(n_1, n_2)$ is *admissible*, it is guaranteed that the path to the goal expanded first is the optimal solution. The meaning of admissible is *never over-estimate*, that is, $h(n_1, n_2) \leq g(n_1, n_2)$ for any two nodes n_1 and n_2 .

An admissible $h(n_1, n_2)$ means that for any node n_1 and n_2 in the A* search tree, if n_1 is the parent node of n_2 , it satisfies that $f(n_2) \geq f(n_1)$. It can be proved as follows:

$$\begin{aligned} f(n_2) &= g(\text{start}, n_2) + h(n_2, \text{goal}) \\ &= g(\text{start}, n_1) + g(n_1, n_2) + h(n_2, \text{goal}) \\ &\geq g(\text{start}, n_1) + h(n_1, n_2) + h(n_2, \text{goal}) \\ &= f(n_1) \end{aligned} \tag{A.1}$$

Since the node with less f-value is expanded earlier, given two nodes g_1 and g_2 satisfy the goal condition, and the cost of reaching g_1 is smaller than reaching g_2 , g_1 will be selected earlier than g_2 . If the parent node of g_1 is p_1 , here we have $f(g_2) > f(g_1) \geq f(p_1)$. It shows that p_1 must be selected earlier than g_2 to expand g_1 , and g_1 will be selected as a solution earlier than g_2 . Therefore, the optimal path is guaranteed to be found first.

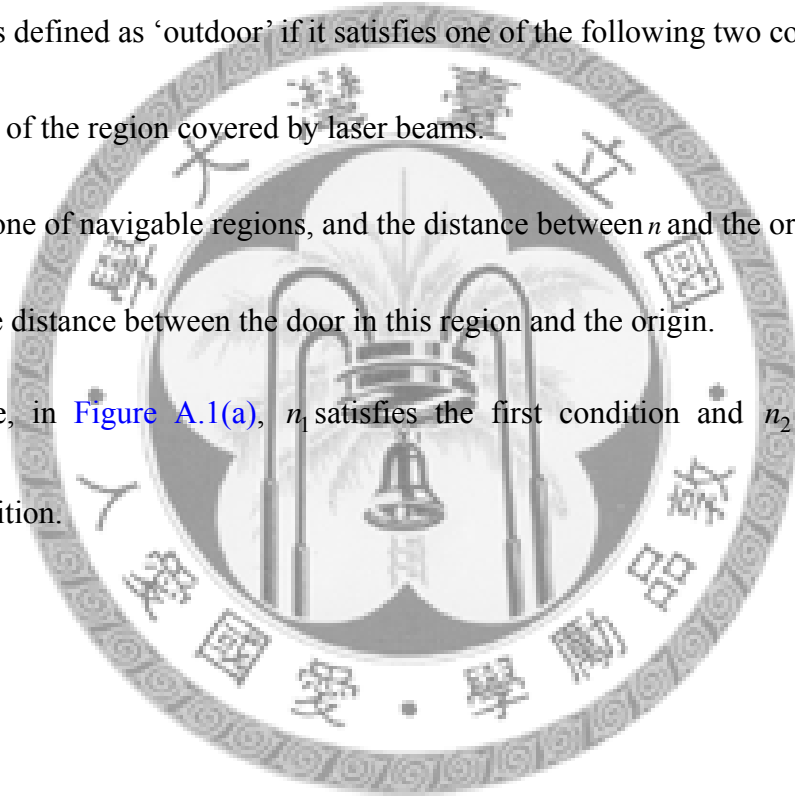
A.2 Door Detection

Figure A.1 shows an example of door detection. As discussed in Section 4.3, the discontinuities on the laser reading histogram are found to derive different navigable regions. For each navigable region, a door position is computed.

A node n is defined as ‘outdoor’ if it satisfies one of the following two conditions:

- n is out of the region covered by laser beams.
- n is in one of navigable regions, and the distance between n and the origin is larger than the distance between the door in this region and the origin.

For example, in Figure A.1(a), n_1 satisfies the first condition and n_2 satisfies the second condition.



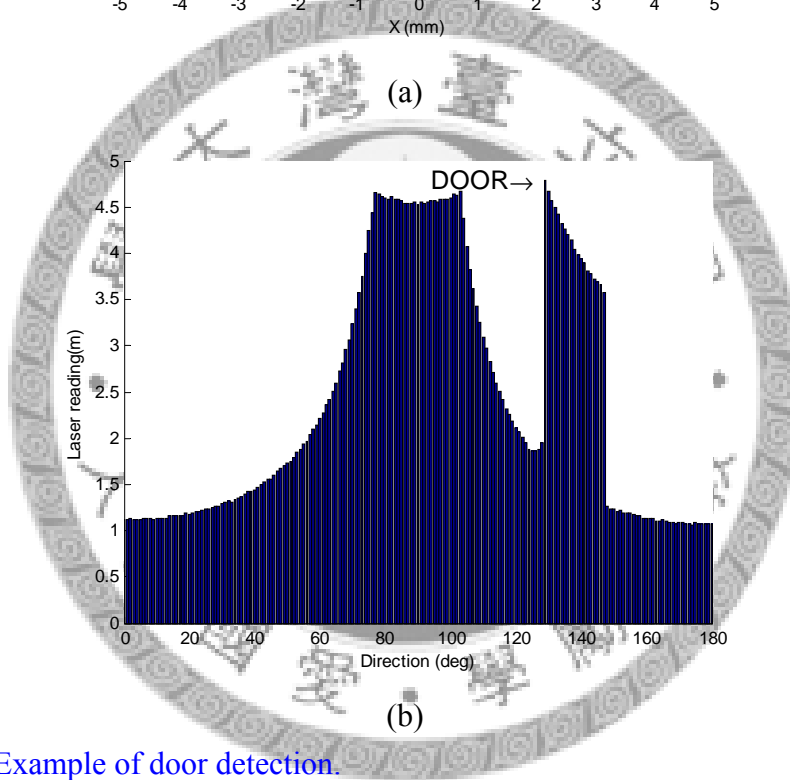
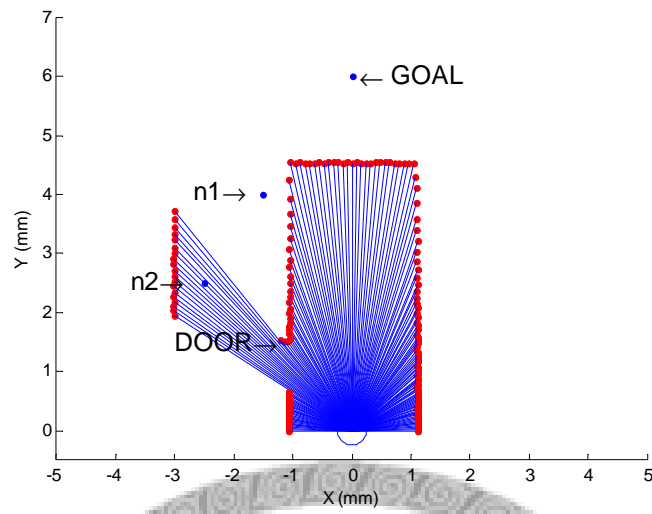


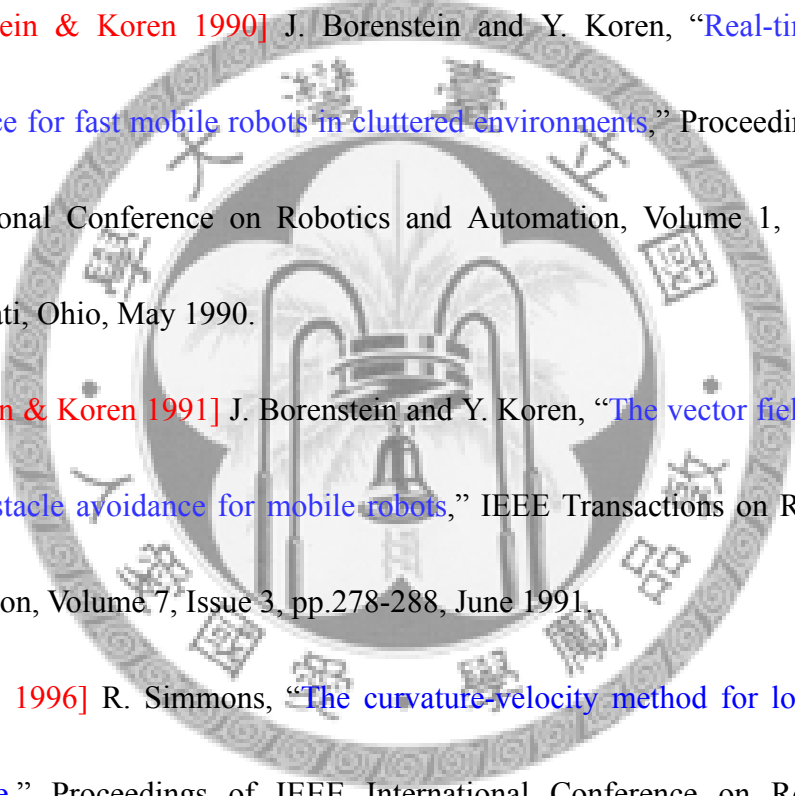
Figure A.1. Example of door detection.

(a) Robot in environment, the blue lines depicts laser beams.

(b) Laser reading histogram.

References

Papers:

- 
- [1: Borenstein & Koren 1990] J. Borenstein and Y. Koren, “Real-time obstacle avoidance for fast mobile robots in cluttered environments,” Proceedings of IEEE International Conference on Robotics and Automation, Volume 1, pp.572-577, Cincinnati, Ohio, May 1990.
- [2: Borenstein & Koren 1991] J. Borenstein and Y. Koren, “The vector field histogram - fast obstacle avoidance for mobile robots,” IEEE Transactions on Robotics and Automation, Volume 7, Issue 3, pp.278-288, June 1991.
- [3: Simmons 1996] R. Simmons, “The curvature-velocity method for local obstacle avoidance,” Proceedings of IEEE International Conference on Robotics and Automation, pp.3375-3382, Minneapolis, Minnesota, Apr. 1996.
- [4: Fox et al. 1997] D. Fox, W. Burgard and S. Thrun, “The dynamic window approach to collision avoidance,” IEEE Robotics & Automation Magazine, Volume 4, Issue 1, pp. 23-33, Mar. 1997.
- [5: Fox et al. 1998] D. Fox, W. Burgard and S. Thrun, “A hybrid collision avoidance

method for mobile robots,” Proceedings of IEEE International Conference on Robotics and Automation, pp. 1238-1243, Leuven, Belgium, May 1998.

[6: Brock & Khatib 1999] O. Brock and O. Khatib, “High-speed navigation using the global dynamic window approach,” Proceedings of IEEE International Conference on Robotics and Automation, Volume 1, pp. 341-346, Detroit, Michigan, May 1999.

[7: Ulrich & Borenstein 1998] I. Ulrich and J. Borenstein, “VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots.” Proceedings of IEEE International Conference on Robotics and Automation, pp. 1572-1577, Leuven, Belgium, May 1998.

[8: Ulrich & Borenstein 2000] I. Ulrich and J. Borenstein, “VFH*: Local Obstacle Avoidance with Look-Ahead Verification.” Proceedings of IEEE International Conference on Robotics and Automation, pp.2505-2511, San Francisco, California, Apr. 2000.

[9: Minguetz & Montano 2004] J. Minguetz and L. Montano, “Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios,” IEEE Transactions on Robotics & Automation, Volume 20, Issue 1, pp.45-59, Feb. 2004.

[10: Seder et al. 2005] M. Seder, K. Macek and I. Petrović, “An integrated approach to real-time mobile robot control in partially known indoor environment,” Proceedings of IEEE International Conference on Industrial Electronics, pp.

1785-1790, 2005.

[11: Seder & Petrović 2007] M. Seder and I. Petrović, “Dynamic window based approach to mobile robot motion control in the presence of moving obstacles,”

Proceedings of IEEE International Conference on Robotics and Automation, pp1986-1981, Roma, Italy, Apr. 2007

[12: Li et al. 2006] G. Li, G. Wu and W. Wei, “ND-DWA: A Reactive Method for Collision Avoidance in Troublesome Scenarios,” Proceedings of the Sixth World

Congress on Intelligent Control and Automation, Volume 2, pp.9307-9311, June. 2006

[13: Stentz 1994] A. Stentz, “Optimal and efficient path planning for partially-known environments,” Proceedings of IEEE International Conference on Robotics and

Automation, San Diego, California, May 1994

Books:

[14: Russell & Norvig 2003] Stuart Russell and Peter Norvig, “Artificial intelligence: A modern approach,” Prentice hall, 2003

[15: Choset et al. 2005] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun, “Principles of robot motion,” The MIT Press, 2005