N° d'ordre: | 2 | 3 | 9 |

ÉCOLE CENTRALE DE LILLE

THÈSE

présentée en vue d'obtenir le grade de

DOCTEUR

Spécialité : Automatique, Génie Informatique, Traitement du Signal et Images

par

Yingchong MA

Doctorat délivré par l'École Centrale de Lille

Titre de la thèse :

Planification de trajectoire et commande pour les robots mobiles non-holonomes

Path planning and control of non-holonomic mobile robots

Soutenue le 19 décembre 2013 devant le jury d'examen :

Président	M. Philippe Fraisse	Université Montpellier 2
Rapporteur	M. Pascal Morin	Université Pierre et Marie CURIE
Rapporteur	M. Cédric Join	Université de Lorraine
Examinateur	Mme. Eva Crück	Direction Générale de l'Armement
Examinateur	M. Jean-Pierre Richard	École Centrale de Lille
Directeur de thèse	M. Wilfrid Perruquetti	École Centrale de Lille
Co-encadrant de thèse	M. Gang Zheng	INRIA LILLE

Thèse préparée dans le Laboratoire d'Automatique, Génie Informatique et Signal
L.A.G.I.S., CNRS UMR 8219 - École Centrale de Lille
École Doctorale SPI 072
PRES Université Lille Nord-de France

À mes parents,

à toute ma famille,

à mes professeurs,

et à mes chèr(e)s ami(e)s.

Acknowledgements

The PhD work presented in this thesis has been done at "Laboratoire d'Automatique, Génie Informatique et Signal (LAGIS)" in Ecole Centrale de Lille, from September 2010 to December 2013. This work is supported by the China Scholarship Council (CSC), EU INTERREG IVA 2 Mers Seas Zeeen Cross-border Cooperation Programme under SYSIASS project 06-020 and Ministry of Higher Education and Research Nord-Pas de Calais Regional Council and FEDER through the "Contrat de Projets Etat Region (CPER) CIA 2007-2013".

I would like to express my most sincere gratitude and appreciation to my supervisors Mr. Wilfrid Perruquetti and Mr. Gang Zheng, for their continuing support, endless patience and valuable guidance and encouragement throughout my years at Ecole Centrale de Lille. Their remarkable mind and admirable qualities have always inspired me, and always remind me of how a great professor should be.

I would like to express my sincere gratitude to the members of my PhD Committee, M. Pascal Morin, M. Cédric Join, M. Philippe Fraisse, Mme. Eva Crück and M. Jean-Pierre Richard.

I also would like to express my sincere gratitude to all the members in the teams of "Equipe des Systèmes Non-linéaires et à Retard" (SyNeR) and "Non-Asymptotic Estimation for Online Systems" (NON-A), it has been a privilege to work together with these intelligent and friendly colleagues. I would especially like to mention the colleagues and friends in my office, Christophe Fiter, Romain Delpoux, Emmanuel Bernuau, Hassan Omran and Qi Guo. Thanks to them, I have passed three agreeable years.

Further thanks give to my parents and family for their incessant love and care, finally, I would like to thank my dearest Hui Wang, who always be so patient and supportive through all the tough times.

ACKNOWLEDGEMENTS

Contents

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

The study of mobile robots started from 1960s, Nilsson et al. developed an autonomous robot named "Shakey" Nilsson (1969) in order to study the artificial intelligence, the autonomous planning and control of robot systems in complex environment. A new research peak occurred world widely in 1980s, a large number of world famous companies, such as *General Electric*, *Honda* and *Sony*, began to develop robot platforms, and these platforms were mainly used in university laboratories and research institutes, which improved the development of various researches of mobile robots. Since 1990s, with the development of computer science, sensor technology and artificial intelligence, there has been a rapid development of robot technology, and the applications of mobile robots were further developed.

In the past a few decades wheeled mobile robots have been more widely studied and attracted more and more interests of many researches because of their wide applications in industries and theoretical challenges Kolmanovsky & McClamroch (1995); Laumond (1998). For example, the *SR4* robot platform on Linux developed by *Smart Robots* company, the wheeled mobile robot *Pioneer P3-DX* developed by *ActivMedia Robotics* for research and teaching, the famous Mars Exploration Rover *Spirit* and etc. More recently, the applications and developments of mobile robots have become more and more popular, and mobile robots have been proposed for using in rescue missions Dissanayake *et al.* (2006); Murphy *et al.* (2009); Nagatani *et al.* (2011), explorations Burgard *et al.* (2005); Rooker &

(a) Mars Exploration Rover (b) Robot soccer game

Figure 1.1: Applications of mobile robots

Birk (2007); Weisbin & Rodriguez (2000), tour guide Han *et al.* (2010); Tomatis *et al.* (2002), and even entertainment such as robot soccer games Camacho *et al.* (2006); Cardoso *et al.* (2012); Kim (2004).

While however with the rapid development of automation and robot technology, there are higher challenges for mobile robots, and the requests for autonomous navigation in complex environment and control accuracy become crucial. This motivates us to focus on the autonomous navigation and trajectory tracking of mobile robots. The general objective is to design new path planning algorithms to navigate robots in complex environment and propose robust controller to track the desired trajectory.

1.2 State-of-the-art

Autonomous navigation is an important issue in robotics research. This problem is of theoretically interesting properties and of practical importance. Navigation is a task that an autonomous robot must do correctly in order to move safely from one location to another without getting lost or colliding with other objects Pearsall (2001). Three general problems are involved in navigation: localization, path planning and motion control. Since we consider the non-holonomic mobile robots, let us firstly take a look at non-holonomic systems.

1.2.1 Non-holonomic systems

Many mechanical systems are subject to constraints of position and velocity, that is to say several relations between the positions and velocities of the different points of the system must be satisfied throughout the movement. The constraints are called holonomic if it is possible to integrate and they lead to the algebraic relations linking to the configuration settings. These relationships can be removed by a suitable change of variables and the system is called holonomic system. In the case of non-integrable constraints, the elimination is not possible and the system is called non-holonomic system.

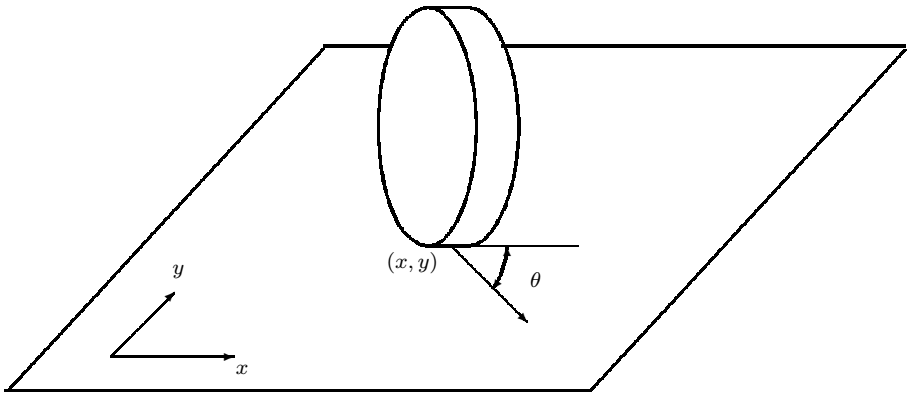


Figure 1.2: Description of a robot wheel

The kinematic expression of a unicycle wheel under the pure rolling and non-slipping assumptions (see Fig. 1.2) can be expressed as follows:

$$\begin{cases} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \omega \end{cases} \tag{1.1}$$

where v is the linear velocity, ω is the angular velocity and θ is the angle of the wheel with respect to x-axis. One can see that the equation (1.1) is subject to the non-holonomic constraint:

$$\dot{y}\cos\theta - \dot{x}\sin\theta = 0 \tag{1.2}$$

Equation (1.1) and (1.2) implies that there are only two possible movements for each configuration, i.e. rolling forward or backward and turning in space, and

this constraint has to be considered when designing control strategy.

The wheeled mobile robots are subject to non-holonomic constraints because of the constraints of rolling wheels, which means that the motion perpendicular to the wheels is impossible for wheeled robots based on the pure rolling hypothesis. Therefore the implementation of non-holonomic mobile robots is a challenge for nonlinear control theory.

The open-loop method and closed-loop method are described to control the non-holonomic mobile robot. The open-loop methods search for feasible trajectories with the initial system state connecting with the final one under the consideration of collision avoidance, minimum path cost, etc. In Dubins (1957) the shortest trajectory between two oriented points was studied, and in Reeds & Shepp (1990) the shortest trajectory is considered to be composed of at most three straight segments and arcs. However as we all know that the open-loop methods are not robust to the disturbance, errors or noises in the system.

Closed-loop methods of non-holonomic systems have been studied, in which the input is constructed as a function of the system state to compensate for noises and errors in the system. However according to Brockett's theorem Brockett *et al.* (1983), there is no smooth feedback control that stabilizes the given configuration, which is also discussed in Bloch (2003) and Bloch *et al.* (1992). This means that the class of the controllers should be extended to take into account the time-varying or the non-smooth controllers Samson & Ait-Abderrahim (1991). The smooth time-invariant control can only be used to achieve non-vanishing Cartesian trajectories tracking (the linear velocity of the robot is assumed to be always nonzero) Campion *et al.* (1991), and non-smooth controls have been used to stabilize non-holonomic mobile robots such as Park *et al.* (2000). For non-vanishing Cartesian trajectories tracking problems, PID controllers Normey-Rico *et al.* (2001), sliding mode controls Defoort *et al.* (2006) and some other methods are used Morin & Samson (2004). Moreover, using the flatness property of the robot system Fliess *et al.* (1995), the dimension of the system can be reduced, which will result in more computation efficiency.

1.2.2 Localization

The localization problem is a key problem in mobile robotics, and it is also the foundation of the path planning and control problem, because first of all the

robot need to know "Where am I?", the title of a publication Borenstein *et al.* (1996). In a little more detail, the goal of a localization task is to estimate the position of the robot in a given repository. In the applications of mobile robots, the accurate localization of the mobile robot is one of the key tasks to ensure the precise navigation, thus the localization should be as accurate as possible based on available measurements.

Generally, the methods of robot localization can also be divided into two categories: absolute localization and relative localization.

In absolute localization problems, robots need to detect different features in the environment to implement desired tasks. One of the absolute methods is landmark-based localization. The robot uses points of known position in the environment which can be "seen" by the robot, named as landmarks, to determine its position. In Conticelli *et al.* (2000) and Martinelli & Siegwart (2005), the localization problem is formulated as an observability problem based on more than three landmarks. In order to solve the problems that it may be difficult to find so many landmarks in some situations, single landmark based methods are developed, in Jang *et al.* (2005) and Lemaire *et al.* (2005), the proposed methods use a single landmark and the shape of the landmark. In Sert *et al.* (2011), the proposed method uses a single landmark and the relative angle between the landmark and the robot. GPS (Global Position System) was developed by the United States Department, and GPS-based navigation systems are also used in absolute localization for a variety of land-based vehicles Abbott & Powell (1999). Recently many different methods have been proposed to reduce errors of GPS in localization problem Qi & Moore (2002); Redmill *et al.* (2001).

In relative localization problems, odometry is widely used because of its short-term accuracy, high sampling rate and easy implementing O'Kane (2006). Odometry is based on some simple equations which can be easily implemented and that utilize data from inexpensive incremental wheel encoders. However the disadvantage of odometry is also well known, it is inaccurate with an unbounded accumulation of errors Gourley & Trivedi (1994). Another approach is based on inertial navigation with gyros and/or accelerometers Barshan & Durrant-Whyte (1995); Brooks (1986). The integration of inertial and visual information is investigated in Viéville & Faugeras (1990). However this approach is not advantageous because the data of accelerometers must be integrated twice to yield position, which makes

these sensors exceedingly sensitive to the drift. In addition, simultaneous localization and mapping (SLAM) problem also attracts interest of many researchers. SLAM is a process by which a mobile robot can build a map of an environment and at the same time use this map to deduce it's location Durrant-Whyte & Bailey (2006). Several techniques have been proposed to tackle the SLAM problem, in Di Marco *et al.* (2004); Dissanayake *et al.* (2001) they represent the environment by a set of characteristic elements detectable by the robot sensory system, in Wulf *et al.* (2004) lines and segments are used to represent the environment, and Williams (2001) generates a short-term submap with the robot's own local coordinate frame using the constrained local submap filter.

Each method described above has its own advantages and disadvantages, thus in real applications two or more methods are often used synthetically to achieve better performance.

1.2.3 Path planning

We call path planning as calculating a feasible path without collision for a robot between a start configuration to a given configuration in a particular environment. Path planning is quite important in robot navigation problems, since it enables the selection and the identification of a suitable path for robots to traverse in the environment. The path planning algorithms can be divided into two broad categories: global path planning and local path planning.

When the environment is completely known before the robot moves, a collision free trajectory with lowest cost from the starting point to the target can be obtained by global path planning algorithms, the cost can be defined to be the travelled distance, energy expended, time exposed to danger, etc. In such cases, the complete information can only be available in static environment, and collision free paths are selected and planned off-line. Different kinds of approaches have been proposed, such as cell decomposition Glavaški *et al.* (2009), visibility graph Bicchi *et al.* (1996); Dudek & Jenkin (2010); Huang & Chung (2004), retraction Ó'Dúnlaing & Yap (1985), Heuristic-based algorithms Dijkstra (1959); Hart *et al.* (1968), genetic algorithms Ismail *et al.* (2008); Nearchou (1998), and projection Schwartz & Sharir (1983) etc.

Cell decomposition is a basic path planning algorithm which divides the space into connected regions called cells. Then find out adjacent cells that have a com-

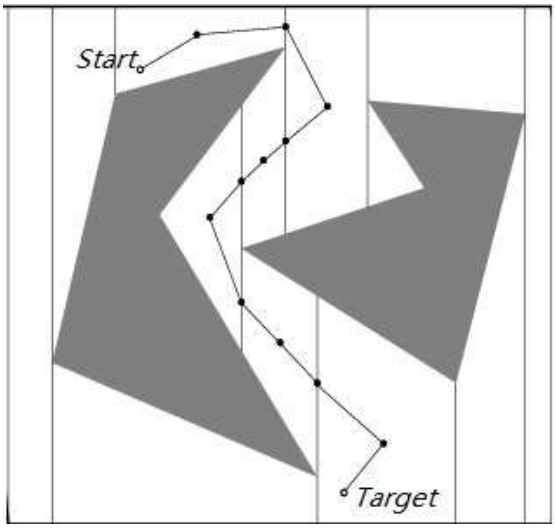


Figure 1.3: Example of cell decomposition

mon boundary, therefore the path can be obtained by connecting the midpoints of the adjacent cells and their common boundaries in order, an example is shown in Fig. 1.3.

The visibility graph is also an efficient approach, and it has the advantage of calculating the shortest collision-free optimal trajectory quickly and easy implementing. Let us firstly give out the definition.

Definition 1.2.1. *Alt & Welzl (1988) Let* $\mathcal{S} = \{S_1, S_2, ..., S_n\}$ *be a set of non-intersecting polygonal obstacles. We denote by* $\mathcal{V} = \{v_1, v_2, ..., v_m\}$ *the set of vertices and by* $\mathcal{E} = \{e_1, e_2, ..., e_m\}$ *the set of edges of polygons bounding the obstacle in* $\mathcal{S}$*. A pair* $\{v_i, v_j\}$ *of vertices in* $\mathcal{V}$ *is visible if the open line segment joining* v_i *and* v_j *is either an edge in* $\mathcal{E}$ *or does not intersect the interior of any obstacle in* $\mathcal{S}$*, and the segment connecting them is called a visibility. To calculate the path from the start point to the goal point, we need to add the start point* $O(x(t), y(t))$ *and the goal point* $G(x_f, y_f)$ *as vertices to* $\mathcal{V}$*, that is, we consider the visibility graph of the set* $\mathcal{V}^* := \mathcal{V} \cup \{O(x(t), y(t)), G(x_f, y_f)\}$*. The visibility graph* G_s *of* $\mathcal{S}$ *is the undirected graph with* $\mathcal{V}^*$ *as set of vertices and the visibility as edges (see Fig. 1.4).*

After the construction of visibility graph, assign each edge $\overline{v_i v_j}$ in the G_s a weight, such as the length of the path, then the shortest path can be easily

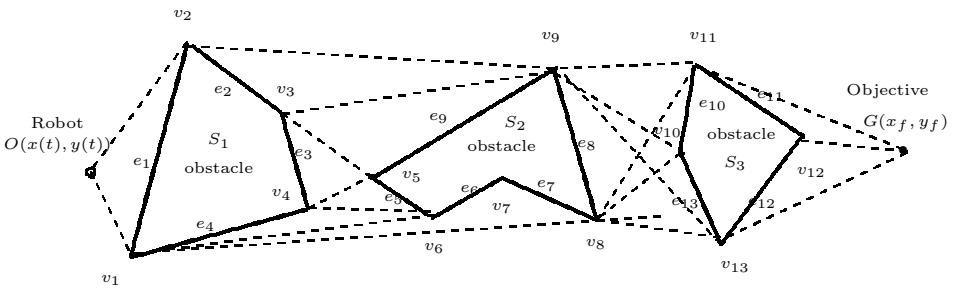


Figure 1.4: Three obstacles and the visibility graph

computed by using search methods such as Dijkstra's algorithm Aho & Hopcroft (1974).

Another well-known algorithm of global heuristics search is A^* Nilsson (1980), which can find the shortest collision free path through a fully mapped environment by using a priority queue. D^* search Stentz (1994) is an extension of the A^* algorithm, and has been used in many applications Choset & Nagatani (2001). It can modify the planned path dynamically if unknown obstacles are encountered.

When the robot has partial knowledge about the environment before it starts, the robot has to plan the path locally with the information captured by the sensor equipped on the robot Goto *et al.* (1987). The Bug1 and Bug2 algorithms Lumelsky & Stepanov (1987) are the earliest and simplest sensor-based path planning algorithms, based on the boundary following method. Another famous algorithm is the artificial potential field approach (APF) proposed in Khatib (1985). The basic idea of this approach is to fill the robot workplace with potential fields, the attractive potential field is caused by the target to attract the robot moving towards the target, and the repulsive potential field is caused by obstacles to repulse the robot away from obstacles. One of the main drawbacks of APF is the local minima when the composition of all forces on the robot is equal to zero. Some extension algorithms based on APF have been proposed Koren & Borenstein (1991); Latombe (1991).

When considering the path planning problem for unicycle-like mobile robots, the physical limitations and the kinematic constraints have to be taken into account. Some algorithms have been proposed for this kind of robots Guo & Tang (2008); Kolmanovsky & McClamroch (1995); Laumond (1998), and an algorithm proposed in Defoort *et al.* (2009) describes the path planning problem as a non-

linear optimal problem with constraints, which guarantees the navigation of the robot in unknown environments. An extended algorithm is proposed in Kökösy *et al.* (2008) based on the Tangent Bug algorithm Kamon *et al.* (1996) to treat the problem of irregular obstacles by following the obstacle boundary.

1.2.4 Motion control

After generating the desired trajectory for the mobile robot, the question that arises is how the physical system realizes the desired movement. Generally the robot control problem can be divided into two main problems: the trajectory tracking problem and stabilization problem. The control problem of trajectory tracking can also be categorized into two types: linear control and nonlinear control. Oelen & van (1994) proposed a linear controller which is robust to the perturbation in robot velocity control. Separated feedback loops control for robot position and velocity was used in Chung & Harashima (2001). The kinematic model of the robot was linearized in Pears (2001), and in which a proportional linear control was applied. The famous PID controller was applied in Normey-Rico *et al.* (2001), in which a simple linearized mobile robot model is used.

The linear control indeed has great advantages because of its simplicity in linear control theory, while however when comparing with nonlinear control its robustness is very limited. In linear control the initial states are often required to stay close to the reference to ensure the stability, instead nonlinear control is able to guarantee the stability without this kind of problems. Moveover, it is known that the feedback stabilization at a given posture cannot be obtained by smooth time-invariant control Campion *et al.* (1991), this implies that the problem is truly nonlinear, and linear control is ineffective here. For nonlinear non-holonomic robot systems, there are usually open loop controls where the inputs are calculated from the reference trajectory De Wit & Sordalen (1992), flatness based control Fliess *et al.* (1995) is a kind of open-loop control, whose robustness can be strengthened Ryu & Agrawal (2008), and it is widely applied in optimal control problems. However, it is well known that the open-loop control is not robust to disturbance and modeling errors so that it cannot guarantee the mobile robot to move along the desired trajectory. Nonlinear feedback control for mobile robots is used in Samson & Ait-Abderrahim (1991) to solve the trajectory tracking problem, and the dynamic feedback linearization is also used

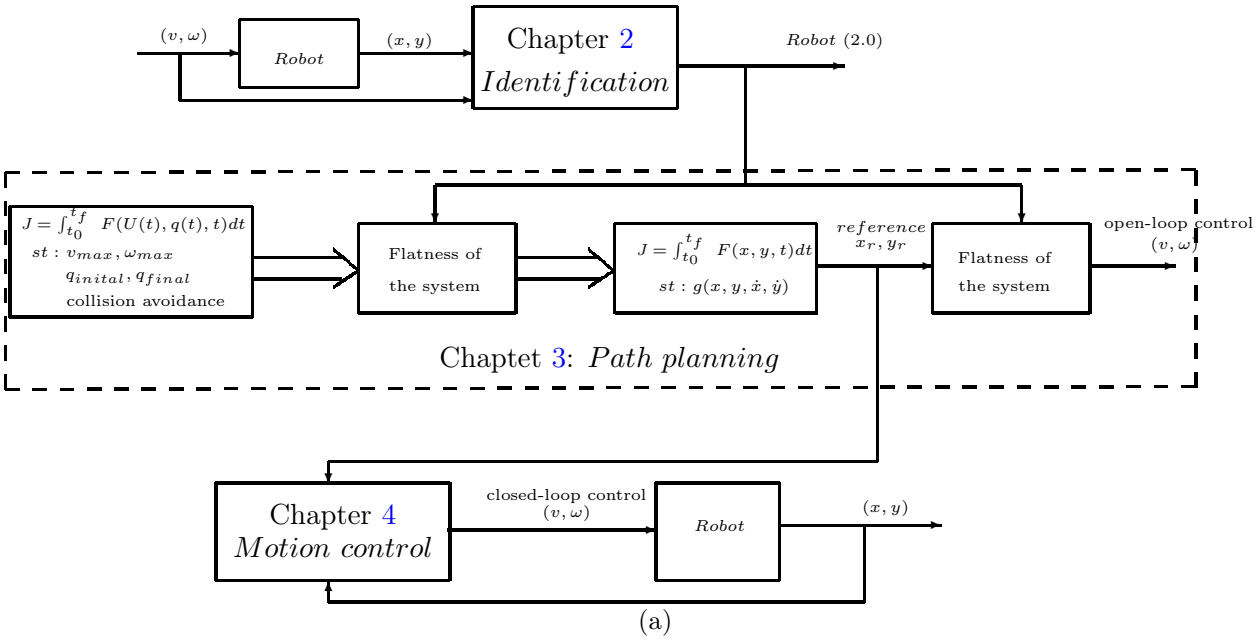
in D'Andréa-Novel *et al.* (1995). Tayebi & Rachid (1996) proposed a nonlinear control law based on partial state feedback linearization and Lyapunov's direct method, but the disturbance and uncertainty were not considered in the control design. There are also many other nonlinear control methods. Sliding mode control (Aguilar *et al.* (1997); Yang & Kim (1999)) is widely accepted because it is capable of coping with uncertainties Perruquetti & Barbot (2002). Neural networks approaches (Liu *et al.* (2007); Yang *et al.* (1998)) is able to predict future robot posture according to the current posture and the controls.

A model-free control approach is introduced in Fliess & Join (2008, 2009), which approximates the system model by a simple local model with unknown term. It exhibits robustness to the unmodeled dynamics and disturbance in the system Fliess *et al.* (2011), and it has been widely studied and applied to many electrical and mechanical processes Gédouin *et al.* (2008); Join *et al.* (2010); Riachy *et al.* (2011); Villagra & Balaguer (2010).

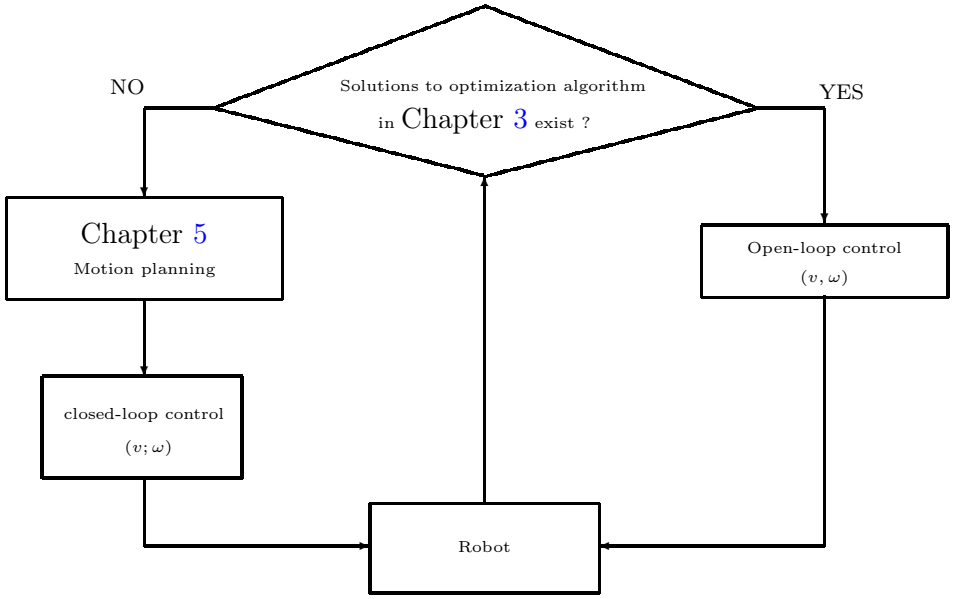
1.3 Outline of the thesis

The outline of the thesis is shown in Fig. 1.5. Chapter 2 presents the identification of different types of non-holonomic mobile robots. The robot models can not be avoided in the navigation problem, because in path planning algorithms, the physical constraints for example the velocity constraint, are expressed as constrains on robot configuration variables Bloch (2003). As for robot control, most of the controllers designed are based on robot models, thus the controllers are different with different robot models. Therefore, given an unknown model of non-holonomic mobile robot, the first task is to identify the robot kinematic model, and then we can design the model-based controller for it. In this chapter the robot identification problem is formulated as the identification of the switching signal of a switched singular nonlinear system, and the distinguishability of the deduced switched singular system is discussed.

With the identified robot kinematic model in Chapter 2, the path planning algorithm can be designed based on the known model. In Chapter 3, a new local path planning algorithm is proposed. Obstacles are assumed as circles in most of the local path planning algorithms, however in real situations, the environment may be complex and normally obstacles cannot be described as circles. Moreover



(a)



(b)

Figure 1.5: Frame of the thesis

in local planning problems, the robot can only "see" a part of an obstacle due to the limitation of the robot sensor. Therefore the obstacles can neither be represented as circles nor as complete obstacles. This motivates us to propose a new algorithm to represent irregular obstacles and generate optimal trajectories for robot without following the obstacle boundaries. The obstacles are represented as segments, and an algorithm of choosing intermediate objectives is proposed. The robot can reach the target and avoid obstacles by reaching the intermediate objectives generated by the new algorithm.

In Chapter 4, an intelligent PID controller (i-PID controller) is proposed to control the non-holonomic robot. The controller used in Chapter 3 is an open-loop control, which is not robust to errors and noises in the system, thus the robust i-PID controller is used to control the robot and the parameter in the controller is discussed to achieve better performance.

Chapter 5 proposes a new potential field method for robot motion planning, and uses the i-PID controller to track the desired velocity. The path planning algorithm proposed in Chapter 3 is considered as an optimal control problem, thus its efficiency largely depends on the optimization algorithm, an inefficient optimization algorithm will result in very long computation time, and normally the programme of optimization problem is rather complex. Moreover in some situations, such as when the robot gets very close to obstacles, there is no solution to the optimization problem. With these disadvantages, we propose a new motion planning method which takes into account the robot orientation and angular velocity, and is able to solve local minima problems and produce smooth repulsive force in complex environment. However as we know that the potential field approach may not give optimal paths, thus as shown in Fig. 1.5(b) we combine the two path planning methods, the robot will switch to the potential field strategy when there is no solution to the optimization algorithm, and then switch back to continue.

In Chapter 6 we consider the cooperation path planning between robots. When there is more than one robot in the environment, robots can share their detected information and the path can be planned more optimally, since a single robot will ultimately be spatially limited. The planning algorithm also generates intermediate objectives based on visibility graph, and in order to cope with

the disadvantages of visibility graph, an algorithm is proposed to generate polygons from a series of jointed segments and merge polygons when two polygons intercross. The reaching of the intermediate objectives is ensured by either optimization algorithm proposed in Chapter 3 or the path planning method proposed in Chapter 5.

1.4 Contribution

This thesis considers the autonomous navigation of non-holonomic mobile robots in complex environment.

First, the identification of different types non-holonomic mobile robot systems is discussed, the proposed technique for identification can be implemented in real-time and is quite robust to the noises in the measurement.

With the identified robot kinematic model, we propose a local path planning algorithm for non-holonomic mobile robots in unknown complex environment. In the proposed path planning algorithm, the irregular obstacles are represented as a series of segments, and local minima problems can be solved by choosing intermediate objectives. The robot can reach the target by reaching the selected intermediate objectives in order.

After having obtained the desired trajectory, an i-PID controller, which is robust to measurement disturbance and is able to stabilize the robot at a static point, is applied for the control of non-holonomic mobile robots. Then the parameter selection of the controller is discussed, and a selection criterion is given.

To cope with the disadvantages of optimal path planning algorithm and local minima problems in classic potential field approach, we propose a new potential field approach for non-holonomic mobile robot path planning, which is able to produce smooth repulsive force in complex environment to avoid oscillations. Then in order to improve the motion control performance, the i-PID controller is used, and the force generated by potential field function is used as reference.

At last, we take into consideration of cooperative path planning of multi-robots. A cooperative path planning algorithm is proposed for the navigation of non-holonomic mobile robots based on visibility graphs. In order to use visibility graph in local planning and to cope with the disadvantages of visibility graph,

an algorithm for expanding obstacles is proposed to provide safe path and merge polygons when two polygons intercross.

Chapter 2

Real-time identification of different types of non-holonomic mobile robots

2.1 Introduction

As stated in Chapter 1, the mobile robot navigation problem is of great importance, and there are considerable research efforts into solving the robot navigation problems in different applications Latombe (1991); Salichs & Moreno (2000). The path planning and motion control of the mobile robots are two main aspects in the navigation problem. Many path planning algorithms have been proposed for wheeled mobile robots Defoort *et al.* (2009); Guo & Tang (2008); Kökösy *et al.* (2008). As for the motion control, no matter what control approach is applied, like PID controllers Ardiyanto (2010), nonlinear feedback control approach Wan & Chen (2008), and sliding mode control Defoort *et al.* (2008); Mehrjerdi & Saad (2010), those controllers are designed based on the robot model. As a result, the robot models can not be avoided in the robot navigation problems, and the designed controllers are different according to different robot kinematic models (an example of different types of robots is shown in Fig. 2.1). Thus, given an unknown model of wheeled mobile robots, the first task is to identify the robot kinematic model, and then we can design the model-based controller for it. Since the kinematic model of mobile robot depends on the construction manners and wheel configurations, which seems difficult to be identified, fortunately, by intro-

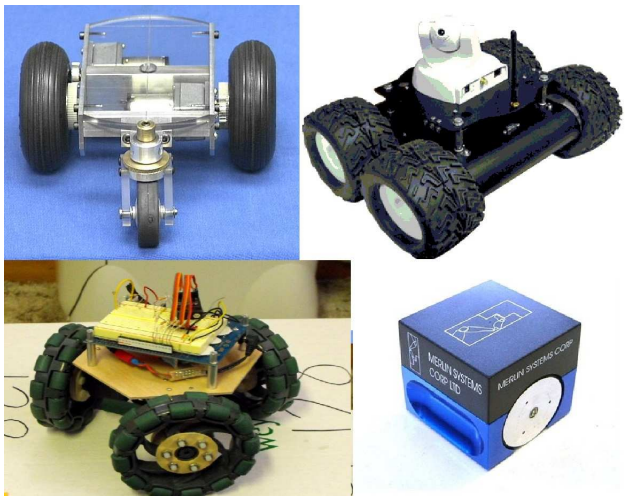


Figure 2.1: Different types of robots

ducing the concepts of *degree of mobility* and *degree of steerability*, the set of kinematic models of wheeled mobile robots can always be partitioned in five classes (see de Wit *et al.* (1996) and Campion *et al.* (1996) for precise definition and classification of mobile robot types).

For those different types of non-holonomic mobile robot, in this chapter we treat the identification problem of the kinematic models as a detection of active mode of a special switched system. A switched system is a dynamical system that consists of a family of subsystems (linear or nonlinear) and a logical rule, called the switching law, that orchestrates switching between these subsystems, and here only one value is possible. In recent years, there has been increasing interest in switched systems due to their significance from both theoretical and practical points of view, and several important results for such systems have been achieved, for example, stability Agrachev & Liberzon (2001); Vu & Liberzon (2005), stabilization De Persis *et al.* (2002); Moulay *et al.* (2007), controllability results Sun *et al.* (2002); Xie *et al.* (2002), and tracking Bourdais *et al.* (2007). Since switched system consists of different subsystems, if we model the subsystem as one of possible kinematic model of non-holonomic robots, then the robot model identification problem becomes the identification of the subsystems of this switched system.

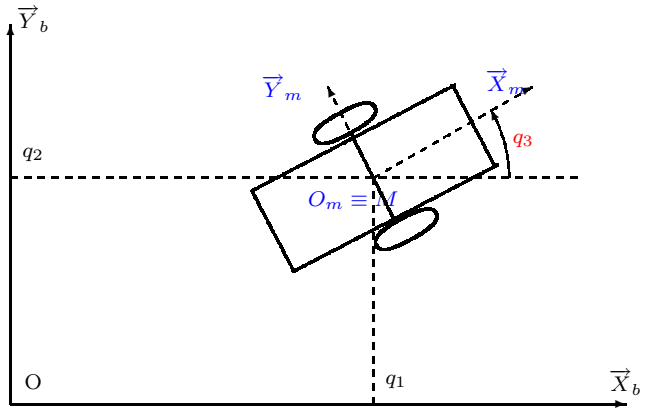


Figure 2.2: Unicycle-type mobile robot

2.2 Robot description

As stated in the introduction, by introducing the concepts of *degree of mobility* and *degree of steerability*, the set of kinematic models of wheeled mobile robots can be partitioned in five classes. Our research considers the first four classes. Let us take the simple unicycle model as an example, which is depicted in Fig. 2.2, with an arbitrary inertial base frame b being fixed in the plane of motion and a frame m being attached to the robot. For a general kinematic model, the state is given by $q = [q_1, q_2, q_3, q_4]^T$, where (q_1, q_2) is the coordinate of its origin O_m, q_3 is the orientation angle with respect to x-axis $\overrightarrow{X}_b$, q_4 is the angle of the plane of steering wheel with respect to the robot frame Y_m when it exists. In the following, the robot types are distinguished as $(a.b)$, where a represents the *degree of mobility* and b represents the *degree of steerability* (see de Wit *et al.* (1996) and Campion *et al.* (1996) for precise definition). Without loss of generality, the kinematic models under the non-holonomic constraints of pure rolling and no slipping exists, thus they can be described as follows:

Type (2.0)

$$\Sigma_1 \begin{cases} \dot{q}_1 &= \nu_1 \cos q_3 \\ \dot{q}_2 &= \nu_1 \sin q_3 \\ \dot{q}_3 &= \nu_2 \end{cases} \tag{2.1}$$

where the control input is $\nu = [\nu_1, \nu_2]^T$ with ν_1 and ν_2 being linear and angular velocity respectively.

Type (3.0)

$$\Sigma_2 \begin{cases} \dot{q}_1 &= \nu_1 \cos q_3 - \nu_2 \sin q_3 \\ \dot{q}_2 &= \nu_1 \sin q_3 + \nu_2 \cos q_3 \\ \dot{q}_3 &= \nu_3 \end{cases} \tag{2.2}$$

where the control is $\nu = [\nu_1, \nu_2, \nu_3]^T$ with ν_1 and ν_2 being the robot velocity components along X_m and Y_m respectively, and ν_3 is the angular velocity.

Type (2.1)

$$\Sigma_3 \begin{cases} \dot{q}_1 &= -\nu_1 \sin(q_3 + q_4) \\ \dot{q}_2 &= \nu_1 \cos(q_3 + q_4) \\ \dot{q}_3 &= \nu_2 \\ \dot{q}_4 &= \nu_3 \end{cases} \tag{2.3}$$

where ν_3 is the angular velocity of the steering wheel, ν_1, ν_2 are defined as those of type (2.0) robot. The control input of this system is defined as that of type (3.0) robot.

Type (1.1)

$$\Sigma_4 \begin{cases} \dot{q}_1 &= -L\nu_1 \sin q_3 \sin q_4 \\ \dot{q}_2 &= L\nu_1 \cos q_3 \sin q_4 \\ \dot{q}_3 &= \nu_1 \cos q_4 \\ \dot{q}_4 &= \nu_2 \end{cases} \tag{2.4}$$

where L is half of the distance between the two fixed wheels, and the input is $\nu = [\nu_1, \nu_2]^T$ with ν_1 being the linear velocity and ν_2 being the angular velocity of the steering wheel.

Note 2.2.1. *Let us note that a last class exists (Type* (1.2)*), if the first four robot kinematic models can be identified, then one can know that the unidentified model is the fifth model (Type* (1.2)*).*

2.3 Determination of input-output equations

2.3.1 Coordinate transformation

As stated above, the robot kinematic models are sets of ordinary differential equations (ODE), our objective is to identify which ODE is active. For each pair of the ODEs, they are distinguishable if for any non trivial input these two systems produce different outputs. In this chapter, it is assumed that one can only measure the position of the robot, i.e. the outputs of the studied system are q_1 and q_2. Then it is necessary to study input-output equations of the systems to

study the distinguishability of the subsystems. In order to facilitate the analysis, let us consider the following change of coordinates

$$\begin{cases} Z &= q_1 + jq_2 \\ \Theta &= e^{jq_3} \end{cases} \tag{2.5}$$

where j represents the imaginary unit ($j^2 = -1$). Applying the change of coordinates to systems (2.1) - (2.4), one can obtain a switched singular system of the following general form:

$$\begin{cases} E_{\sigma(t)}\dot{x} &= G_{\sigma(t)}(x)u \\ Y &= Cx \end{cases} \tag{2.6}$$

where $x = [Z, \Theta, q_4]^T$ is the system state, $u = [\nu_1, \nu_2, \nu_3]^T$ is the input, Y is the output with $C = [1, 0, 0]$. The switching function is defined as

$$\sigma(t) : \mathbb{R}^+ \rightarrow \mathfrak{I},\ \mathfrak{I} \triangleq \{1, 2, 3, 4\}$$

and for different subsystems, one has

$$E_1 = E_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, E_3 = E_4 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$G_1(x) = \begin{pmatrix} \Theta & 0 & 0 \\ 0 & j\Theta & 0 \\ 0 & 0 & 0 \end{pmatrix}, G_3(x) = \begin{pmatrix} j\Theta e^{jq_4} & 0 & 0 \\ 0 & j\Theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$G_2(x) = \begin{pmatrix} \Theta & j\Theta & 0 \\ 0 & 0 & j\Theta \\ 0 & 0 & 0 \end{pmatrix}, G_4(x) = \begin{pmatrix} jL\Theta \sin q_4 & 0 & 0 \\ j\Theta \cos q_4 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

It is well known that for a singular system (switched or not), the output may be not differentiable due to the existence of the singular matrix ($E_{\sigma(t)}$ in system (2.6)). However, since system (2.6) possesses special structure, i.e. $C = CE_{\sigma(t)}$ and $G_{\sigma(t)}(x) = G_{\sigma(t)}(E_{\sigma(t)}x)$, thus the output of this system is successively differentiable, i.e. $Y \in C^\infty$.

2.3.2 Input-output equations

Now the problem formulated here becomes the real time computation of the switching signal $\sigma(t)$ to identify the subsystems of (2.6). Since one can identify the

switching signal $\sigma(t)$ by using the input and the output of the system, it is clear that one needs to deduce some input-output representations of each subsystem. One can notice that system (2.6) is defined in complex domain, and we need to take complex transforms, thus let us firstly give some necessary definitions.

For a given scalar complex function of $x \in \mathbb{C}^n$, one can note it as $z(x) = a(x) + jb(x)$, where $z : \mathbb{C}^n \to \mathbb{C}$, $a : \mathbb{C}^n \to \mathbb{R}$, $b : \mathbb{C}^n \to \mathbb{R}$, the partial derivative of z with respect to x is defined as $\frac{\partial z}{\partial x} = \frac{\partial a}{\partial x} + j\frac{\partial b}{\partial x}$. If the matrix $\frac{\partial(z,\dot{z},...,z^{(n)})}{\partial x}$ has row rank r in complex domain, then we note as $rank_{\mathbb{C}}\frac{\partial(z,\dot{z},...,z^{(n)})}{\partial x} = r$. Then we have the following theorem on input-output equations.

Theorem 2.3.1. *Given switched singular system of the form (2.6), where $x \in \mathbb{C}^n$, $u \in \mathbb{R}^m$, $E_{\sigma(t)} \in \mathbb{R}^{n\times n}$, $G_{\sigma(t)} \in \mathbb{C}^{n\times m}$, $C \in \mathbb{C}^{1\times n}$ with $C = CE_{\sigma(t)}$ and $G_{\sigma(t)}(x) = G_{\sigma(t)}(E_{\sigma(t)}x)$, if*

$$rank_{\mathbb{C}}\frac{\partial(Y,\dot{Y},...,Y^{(l_{\sigma(t)}-1)})}{\partial x} = rank_{\mathbb{C}}\frac{\partial(Y,\dot{Y},...,Y^{(l_{\sigma(t)})})}{\partial x}$$

then there exists an input-output representation of each subsystem of (2.6), and this input-output function can be obtained by taking $l^{th}_{\sigma(t)}$ derivative of the output Y.

Proof. The proof of this theorem is based on Conte *et al.* (1999), which proves the existence of the input-output functions of the regular nonlinear systems.

For system (2.6), if

$$rank_{\mathbb{C}}\frac{\partial(Y,\dot{Y},...,Y^{(l_{\sigma(t)}-1)})}{\partial x} = rank_{\mathbb{C}}\frac{\partial(Y,\dot{Y},...,Y^{(l_{\sigma(t)})})}{\partial x}$$

one can conclude that

$$\frac{\partial Y^{(l_{\sigma(t)})}}{\partial x} \in span_{\mathbb{C}}\left\{\frac{\partial Y}{\partial x}, \frac{\partial \dot{Y}}{\partial x}, ..., \frac{\partial Y^{(l_{\sigma(t)}-1)}}{\partial x}\right\}$$

thus the vector $S = (Y, \dot{Y}, ..., Y^{l_{\sigma(t)}-1})$ satisfies the relation

$$rank_{\mathbb{C}}\frac{\partial S}{\partial x} = l_{\sigma(t)}$$

In this case, there exist analytic functions $p_1(x), ..., p_{n-l_{\sigma(t)}}(x)$ such that the

matrix

$$J = \frac{\partial(S, p_1, p_2, ..., p_{n-l_{\sigma(t)}})}{\partial x}$$

has full rank. Then one has the system of equations

$$\begin{cases} \tilde{x}_1 & = & CE_{\sigma(t)}x \\ \tilde{x}_2 & = & CE_{\sigma(t)}\dot{x} = CG_{\sigma(t)}(E_{\sigma(t)}x)u \\ & . & \\ & . & \\ & . & \\ \tilde{x}_{l_{\sigma(t)}} & = & (CE_{\sigma(t)}x)^{(l_{\sigma(t)}-1)} = h(E_{\sigma(t)}x, u, ..., u^{(l_{\sigma(t)}-1)}) \\ \tilde{x}_{\sigma(t)+k} & = & p_k(E_{\sigma(t)}x, u, ..., u^{(\gamma)}), \quad k = 1, 2, ..., n - l_{\sigma(t)} \end{cases} \tag{2.7}$$

It can be concluded that (2.7) is of the form $F_k(E_{\sigma(t)}x, \tilde{x}, u, ..., u^{(\gamma)}) = 0, k = 1, 2, ..., n$ with

$$\frac{\partial(F_1, F_2, ..., F_n)}{\partial x} = J$$

Therefore there exist n functions

$$\phi_k(\tilde{x}, u, ..., u^{(\gamma)}) = x_k \quad for \quad 1 \leq k \leq n$$

which defines a local diffeomorphism ϕ parameterized by $u, \dot{u}, ..., u^{(\gamma)}$:

$$x = \phi(\tilde{x}) \tag{2.8}$$

Applying the change of coordinates induced by (2.8), the system (2.6) becomes:

$$\begin{cases} \dot{\tilde{x}}_1 & = & \tilde{x}_2 \\ \dot{\tilde{x}}_2 & = & \tilde{x}_3 \\ & . & \\ & . & \\ & . & \\ \dot{\tilde{x}}_{l_{\sigma(t)}} & = & h^{(l_{\sigma(t)})}(\phi(\tilde{x}), u, ..., u^{(\gamma)}) \\ \dot{\tilde{x}}_{l_{\sigma(t)}+k} & = & p_k(\phi(\tilde{x}), u, ..., u^{(\gamma)})) \quad k = 1, 2, ..., n - l_{\sigma(t)} \\ Y & = & \tilde{x}_1 \end{cases} \tag{2.9}$$

Since we notice that

$$\begin{aligned} Y &= \tilde{x}_1 \\ \dot{Y} &= \tilde{x}_2 \\ &\vdots \\ Y^{(l_{\sigma(t)}-1)} &= \tilde{x}_{l_{\sigma(t)}} \end{aligned}$$

thus the input-output functions can be obtained from (2.9) as follows:

$$Y^{l_{\sigma(t)}} = h^{(l_{\sigma(t)})}(\phi(Y, \dot{Y}, ..., Y^{(l_{\sigma(t)}-1)}), u, ..., u^{(\gamma)}) \tag{2.10}$$

□

From Theorem 2.3.1 one can conclude that there always exist input-output equations for each subsystem of (2.6), and the input-output equations can be obtained by taking 2^{nd} order derivative of the output Y. Taking the subsystem $\sigma(t) = 1$ as an example, since $Y = Z$, then one has

$$\dot{Y} = \dot{Z} = \nu_1 \Theta \tag{2.11}$$

and

$$\ddot{Y} = \dot{\nu}_1 \Theta + \nu_1 \dot{\Theta} = (\dot{\nu}_1 + j\nu_1\nu_2)\Theta \tag{2.12}$$

which leads to

$$\frac{\partial(Y, \dot{Y})}{\partial x} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \nu_1 & 0 \end{pmatrix}$$

and

$$\frac{\partial(Y, \dot{Y}, \ddot{Y})}{\partial x} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \nu_1 & 0 \\ 0 & \dot{\nu}_1 + j\nu_1\nu_2 & 0 \end{pmatrix}$$

One can see that, in the real field, $rank_{\mathbb{R}} \frac{\partial(Y,\dot{Y})}{\partial x} = 2$ and $rank_{\mathbb{R}} \frac{\partial(Y,\dot{Y},\ddot{Y})}{\partial x} = 3$. However, in the complex field, one has $rank_{\mathbb{C}} \frac{\partial(Y,\dot{Y})}{\partial x} = rank_{\mathbb{C}} \frac{\partial(Y,\dot{Y},\ddot{Y})}{\partial x} = 2$, thus the condition of Theorem 2.3.1 is satisfied, and the input-output equation can be calculated with the 2^{nd} order derivative of Y. An easy calculation via equation (2.11) and (2.12) yields the following input-output equation:

$$\ddot{Y} = \frac{\dot{\nu}_1}{\nu_1}\dot{Y} + j\nu_2\dot{Y} \tag{2.13}$$

Analogously, one can obtain input-output equations for other subsystems. When $\sigma(t) = 2$, one obtains

$$\ddot{Y} = \frac{\dot{\nu}_1 + j\dot{\nu}_2}{\nu_1 + j\nu_2}\dot{Y} + j\nu_3\dot{Y} \tag{2.14}$$

For $\sigma(t) = 3$, one has

$$\ddot{Y} = \frac{\dot{\nu}_1}{\nu_1}\dot{Y} + j(\nu_2 + \nu_3)\dot{Y} \tag{2.15}$$

and if $\sigma(t) = 4$, the input-output equation is of the following form

$$\ddot{Y} = \frac{\dot{\nu}_1}{\nu_1}\dot{Y} - \frac{\nu_2 L\cos(\arg\dot{Y})\frac{d(\arg\dot{Y})}{dt}}{Re(\dot{Y})}\dot{Y} + j\frac{d(\arg\dot{Y})}{dt}\dot{Y} \tag{2.16}$$

where $Re(\dot{Y})$ and $\arg\dot{Y}$ are the real part and the argument of the complex number $\dot{Y}$ respectively, and $\dot{F} = \frac{d(F)}{dt}$ represents the differentiation of function F with respect to t.

2.4 Distinguishability

2.4.1 Distinguishability of input-output equations

Once the input-output equations are obtained, one can use them to analyze the distinguishability of the subsystems. Let us firstly recall the definition of the distinguishability.

Definition 2.4.1. *Fliess* et al. *(2008a) The two subsystems are said to be strongly distinguishable if, and only if, the subsystems have the same input-output behavior only when* $U = [\ 0 \quad 0 \quad 0\ ]$ *and* $Y = 0$. *If not, the two subsystems are said to be weekly distinguishable.*

It is clear that the subsystems of (2.6) are distinguishable with non trivial inputs, thus the problem consists in seeking the peculiar inputs that produce the same output for the subsystems, in which cases the subsystems are not distinguishable.

Theorem 2.4.1. *The subsystems of* (2.6) *can be distinguished, if and only if the input* $\nu_1 \neq 0$, $\nu_2 \neq 0$ *and* $\nu_3 \neq 0$.

Proof. Firstly let us prove the sufficiency. For the input-output equations (equation (2.13) - (2.16)) of subsystems of system (2.6), let us denote I_i as the i^{th} input-output equation, where $i \in [1, 2, 3, 4]$. If the four equations have the same form, i.e. $I_i = I_j$, $i, j \in [1, 4]$, and $i \neq j$, then we can obtain that $\nu_1 = \nu_2 = \nu_3 = 0$. Thus one can see that if $\nu_1 \neq 0$, $\nu_2 \neq 0$ and $\nu_3 \neq 0$, there is no input-output equation that has both the same real part and imaginary part as another input-output equation, therefore the input-output equations are of the different form, and we can conclude that these subsystems are distinguishable if $\nu_1 \neq 0$, $\nu_2 \neq 0$ and $\nu_3 \neq 0$.

Then let us prove the necessity. Firstly one can notice that if the subsystems are distinguishable, we must have $\nu_1 \neq 0$, since the input-output equations ((2.13),(2.15) and (2.16)) can not be calculated when $\nu_1 \neq 0$. Now let us consider each pair of the subsystems.

If subsystems $\sigma(t) = 1$ and $\sigma(t) = 2$ are distinguishable, the equations (2.13) and (2.14) must be different, the two equations are of the same form if and only if $\nu_2 = 0$ and $\nu_3 = 0$. Thus if the two subsystems are distinguishable, we have $\nu_2 \neq 0$ and $\nu_3 \neq 0$.

For subsystems $\sigma(t) = 1$ and $\sigma(t) = 3$, (2.13) and (2.15) are of the same form if and only if $\nu_3 \neq 0$. Thus we have $\nu_3 \neq 0$, if subsystems $\sigma(t) = 1$ and $\sigma(t) = 3$ are distinguishable.

Analogously for subsystems $\sigma(t) = 2$ and $\sigma(t) = 3$, we have $\nu_2 \neq 0$, if equation (2.14) and (2.15) are dinstinguishable.

For subsystems $\sigma(t) = 1$ and $\sigma(t) = 4$, the equations are of the same form if and only if when $\nu_2 = \frac{d(\arg \dot{Y})}{dt}$ and $\nu_2 = 0$ or $\cos(\arg \dot{Y}) = 0$ or $\frac{d(\arg \dot{Y})}{dt} = 0$. However $\cos(\arg \dot{Y})$ is determined by the inputs and varies during the control process, thus we can conclude that the two equations are of the same form if and only if $\nu_2 = \frac{d(\arg \dot{Y})}{dt} = 0$. Consequently, we have $\nu_2 \neq 0$, if subsystems $\sigma(t) = 1$ and $\sigma(t) = 4$ are distinguishable.

As for subsystems $\sigma(t) = 2$ and $\sigma(t) = 4$, equations (2.14) and (2.16) are of the same form if and only if $\nu_2 = \nu_3 = \frac{d(\arg \dot{Y})}{dt} = 0$, thus we have $\nu_2 \neq 0$ or $\nu_3 \neq 0$ if subsystems $\sigma(t) = 2$ and $\sigma(t) = 4$ are distinguishable.

Analogously for subsystems $\sigma(t) = 3$ and $\sigma(t) = 4$, equations (2.15) and (2.16) are of the same form if and only if when $\nu_2 = \nu_3 = \frac{d(\arg \dot{Y})}{dt} = 0$, thus we have $\nu_2 \neq 0$ or $\nu_3 \neq 0$ if subsystems $\sigma(t) = 3$ and $\sigma(t) = 4$ are distinguishable.

In summary, we have $\nu_1 \neq 0$, $\nu_2 \neq 0$ and $\nu_3 \neq 0$, if the subsystems are distinguishable. □

Once the input-output equations are distinguishable, then one can use those equations to identify the switching signal, which will be detailed in the following.

2.4.2 Calculation of residuals

After having obtained the distinguishable input-output equation for each subsystem of (2.6), let us define the residual associated to the subsystem as follows:

$$R_i(t) = \begin{cases} \ddot{Y} - \frac{\dot{\nu}_1}{\nu_1}\dot{Y} - j\nu_2\dot{Y}, & i = 1 \\ \ddot{Y} - \frac{\dot{\nu}_1 + j\dot{\nu}_2}{\nu_1 + j\nu_2}\dot{Y} - j\nu_3\dot{Y}, & i = 2 \\ \ddot{Y} - \frac{\dot{\nu}_1}{\nu_1}\dot{Y} - j(\nu_2 + \nu_3)\dot{Y}, & i = 3 \\ \ddot{Y} - \frac{\dot{\nu}_1}{\nu_1}\dot{Y} + \frac{\nu_2 L \cos(\arg \dot{Y}) \frac{d(\arg \dot{Y})}{dt}}{Re(\dot{Y})}\dot{Y} - j\frac{d(\arg \dot{Y})}{dt}\dot{Y}, & i = 4 \end{cases}$$

It is clear that the current i^{th} subsystem is active if $R_i(t) = 0$. Since the residuals are complex numbers, the corresponding $\sigma(t)$ can be identified, if both the real part and the imaginary part of $R_i(t)$ converge to zero within a short time period, thus

$$\sigma(t) = i, \quad if \int_{T_R} |R_i(t)|dt = 0, i \in \mathcal{I} \tag{2.17}$$

where T_R is a freely chosen but very short residual judging window.

It should be noted that the judging rule (2.17) is valid only for the case where one can precisely measure the input, the output and their derivatives. However, for the case where the input or the output are corrupted with noises, or the derivatives of the input and the output are not known (in this case one need to calculate them by some additional techniques), the calculated residuals do not exactly equal to 0 within the time window T_R, then the judging rule (2.17) can be replaced by the following one:

$$\sigma(t) = \arg\min_{i \in \mathcal{I}} \int_{T_R} |R_i(t)|dt \tag{2.18}$$

In this case, the problem is then reduced to a real-time computation of time derivative of the input and the output of the studied system despite of noises, which makes the calculation of the derivative become a crucial issue.

2.4.3 Numerical differentiation

The numerical differentiation technique presented here was proposed by Fliess et al. in Sira-Ramirez & Fliess (2006), and more details can be found in Fliess *et al.* (2008b); Liu *et al.* (2011); Mboup *et al.* (2009) and references therein.

Consider a signal $y(t) = \sum_{k=0}^{\infty} y^{(k)}(0)\frac{t^k}{k!}$ which is assumed to be analytic around $t = 0$ and its truncated Taylor expansion $y_N(t) = \sum_{k=0}^{N} y^{(k)}(0)\frac{t^k}{k!}$, where $t > 0$. Its Laplace transform is of the form:

$$Y_N(s) = \sum_{k=0}^{N} \frac{y^{(k)}(0)}{s^{k+1}} \tag{2.19}$$

Introducing the *algebraic derivation* $\frac{d}{ds}$, and multiply both sides of equation (2.19) by $\frac{d^\alpha}{ds^\alpha}s^N$, $\alpha = 0, 1, ..., N$, one has a triangular system of linear equations and from which the derivatives can be obtained:

$$\frac{d^\alpha s^N Y_N}{ds^\alpha} = \frac{d^\alpha}{ds^\alpha}(\sum_{k=0}^{N} y^{(k)}(0)s^{N-k-1}) \tag{2.20}$$

which is independent of all the unknown initial conditions, and the coefficients $y(0), ..., y^{(k)}(0)$ are *linearly identifiable* Fliess & Sira-Ramírez (2003), then $y^{(k)}(0)$ can be obtained by taking inverse laplace transform of (2.20) over a time window T.

It is worth noting that the algebraic technique stated here is robust with respect to noises involved into the control inputs and outputs Fliess *et al.* (2008b). Noises are viewed here as highly fluctuations around 0, therefore they can be attenuated by low-pass filters, as iterated integrals with respect to time Fliess *et al.* (2004). Moreover this algebraic technique has other advantages: it is of non-asymptotic nature, the desired estimation can be obtained instantaneously; it provides explicit formulae, which can be implemented directly; it does not require any assumption concerning the statistical distribution of the unstructured noise.

In practice, this algebraic technique is implemented with discrete measured data, thus it is necessary that the sampling time T_s should be small enough with respect to the duration time between two successive switchings* Liu *et al.*

*In practice it is at least 100 times smaller, thus the Zeno phenomenon are excluded

(2011); Mboup *et al.* (2009). Moreover in Liu *et al.* (2011) some other analysis are discussed for several classes of noises.

2.5 Simulation results

Usually the model of the robot will not change, thus $\sigma(t)$ is fixed for $t > 0$. However since we assume that the robot type is unknown, thus $\sigma(t)$ depends on different types of robots. In order to show the feasibility of the proposed method, it is assume that $\sigma(t)$ is a time-varying signal. In the previous section we have discussed the condition of the distinguishability, thus one can choose $v = [1.5, 1.3, 0.5]^T$ to avoid indistinguishable cases. For the simulation settings, the sampling time of numerical differentiator is $Ts = 0.005\ s$, the sliding time window is $T = 0.5\ s$ and the residual judging window is $T_R = T = 0.5\ s$. The switching signal $\sigma(t)$ is shown in Fig. 2.3, and the output of the switching system is shown in Fig. 2.4.

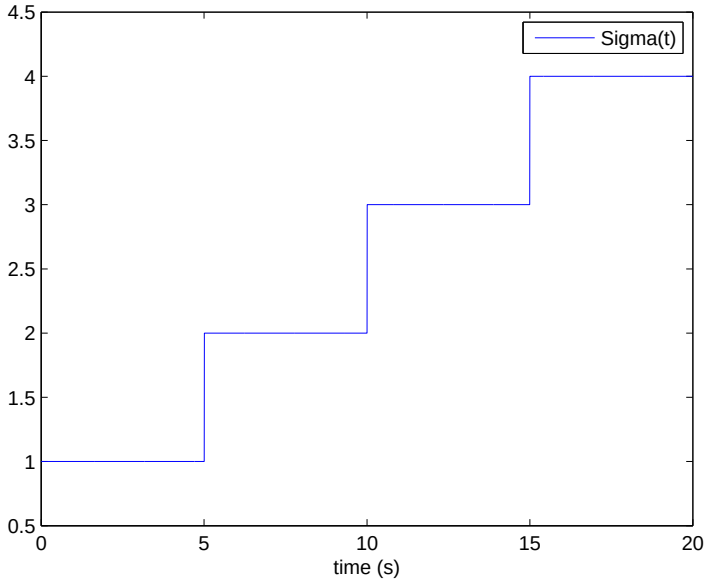


Figure 2.3: Switching signal $\sigma(t)$

The first scenario supposes that the output and the differentiation of the output can be directly obtained without noises, and the simulation results are

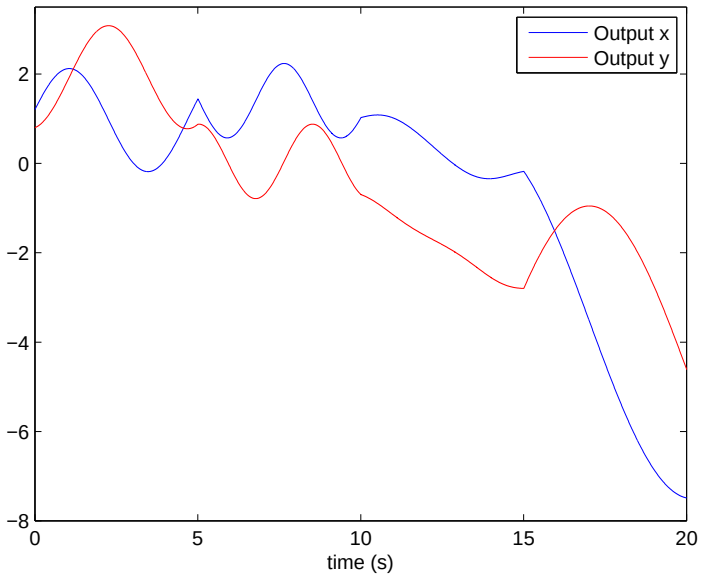


Figure 2.4: Output of the system

shown in Fig. 2.5 and Fig. 2.6. In this case, the switching signal is identified by $\int_{T_R} |R_i(t)|dt = 0$ and one can see that the identification of the active mode is perfect.

The second scenario assumes that the differentiation of the outputs is not directly known but without noises, and the 1^{st} and 2^{nd} order derivatives of the outputs are calculated by numerical differentiator presented in section 2.4.3. Simulation results are shown in Fig. 2.7 and Fig. 2.8 with the same input and output as the former simulation. One can see that the residuals do not equal to 0 because of the calculation errors, and the switching signal is identified by $\sigma(t) = \arg\min_{i \in \mathcal{I}} \int_{T_R} |R_i(t)|dt$. One can also notice that there exists a short interval where $\sigma(t)$ can not be identified, and this is due to the fact that we use the numerical differentiator and the integral of R_i over the residual judging window T_R, so this non identifiable interval is equal to T and can be reduced by reducing the time window T and residual judging window T_R.

The final scenario is similar to the second scenario, but supposes that there are noises (as shown in Fig. 2.9 and Fig. 2.10) adding to the output measurement, shown in Fig. 2.11. The simulation results are depicted in Fig. 2.12 and Fig. 2.13 with the white Gaussian noise of $SNR = 50dB$ (signal-to-noise ratio), and

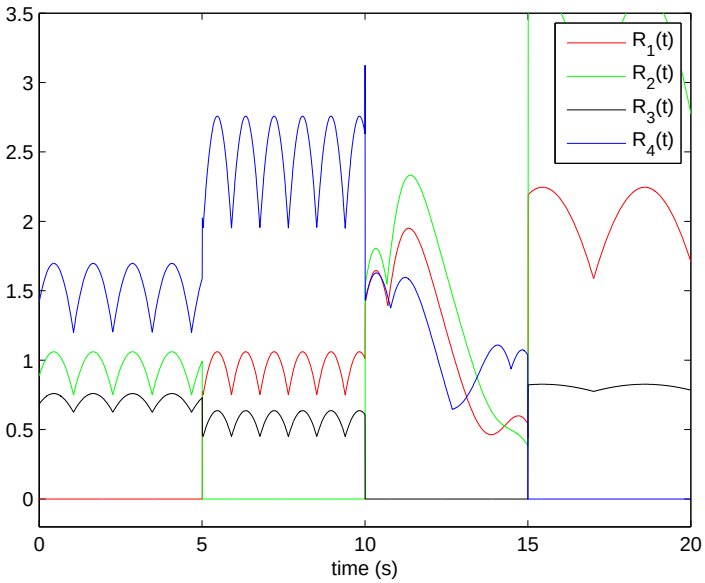


Figure 2.5: Residuals when the differentiations can be obtained directly

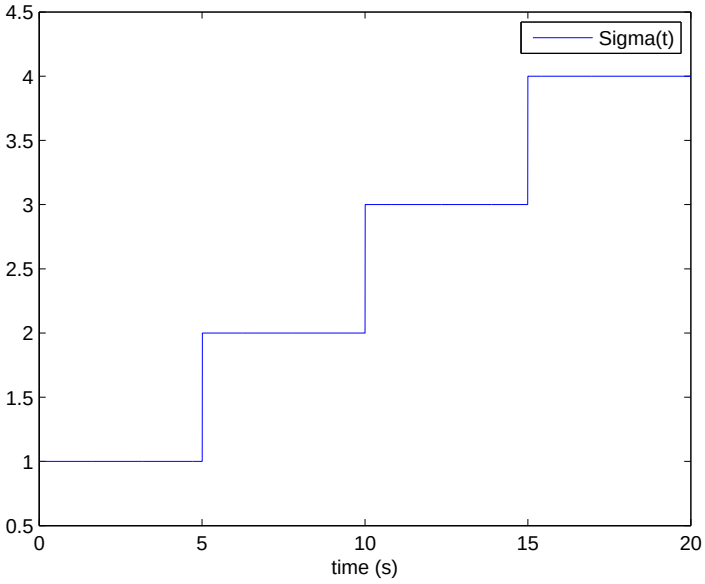


Figure 2.6: Switching signal $\sigma(t)$ identified when the differentiations can be obtained directly

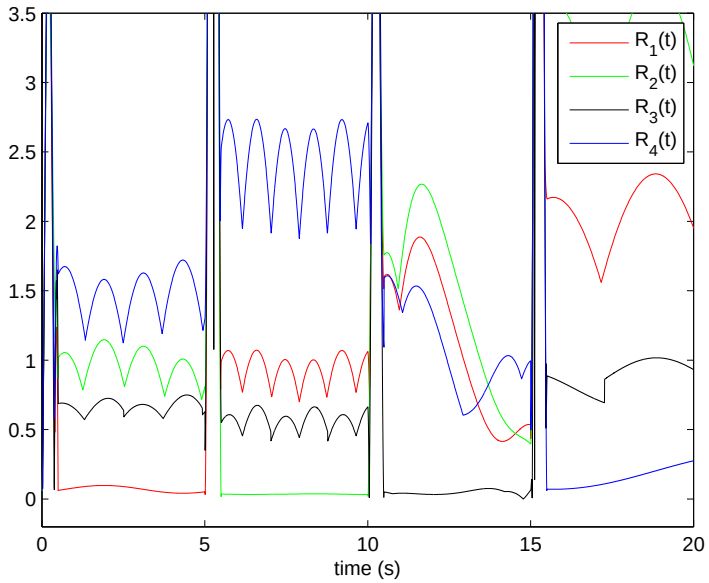


Figure 2.7: Residuals when the differentiations are not known and calculated by the numerical differentiator

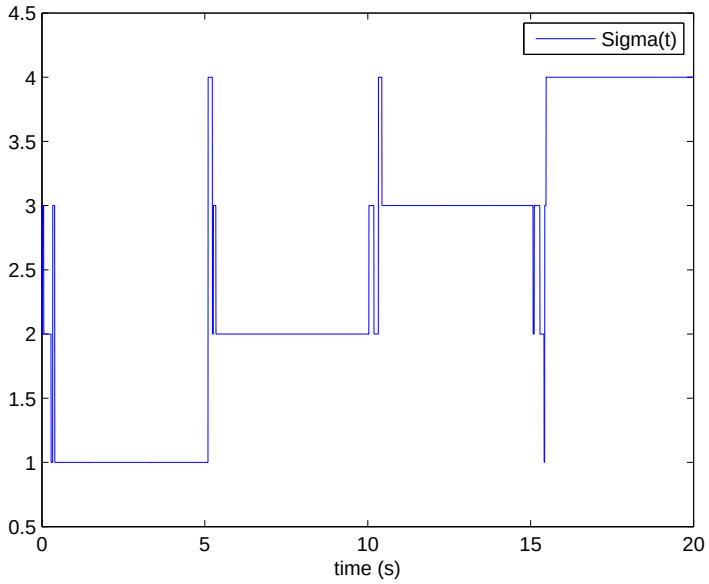


Figure 2.8: Switching signal $\sigma(t)$ identified when the differentiations are not known and calculated by the numerical differentiator

the identified switching signal $\sigma(t)$ is the same as previous simulations. One can conclude from the results that the proposed method is robust to the noises, and the subsystems can be identified quickly in real time by using the judging rule: $\sigma(t) = \arg\min_{i \in \mathcal{J}} \int_{T_R} |R_i(t)| dt$.

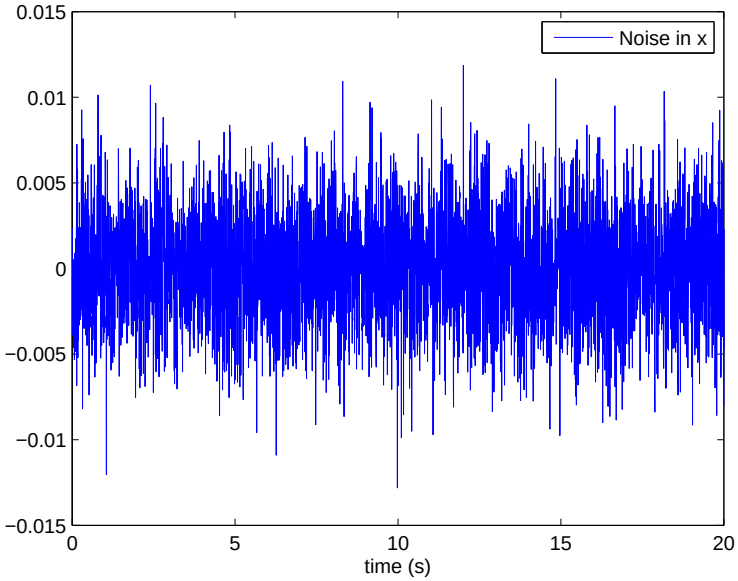


Figure 2.9: Noise imposed in x direction

2.6 Conclusion

The identification of non-holonomic mobile robot systems is discussed in this chapter, the problem is formulated as the identification of the switching signal of a switched singular nonlinear system, and the distinguishability of the deduced switched singular system is studied. The proposed technique can be implemented in real-time and it is quite robust to the noises in the measurement. The good performance of the technique was validated by several simulations.

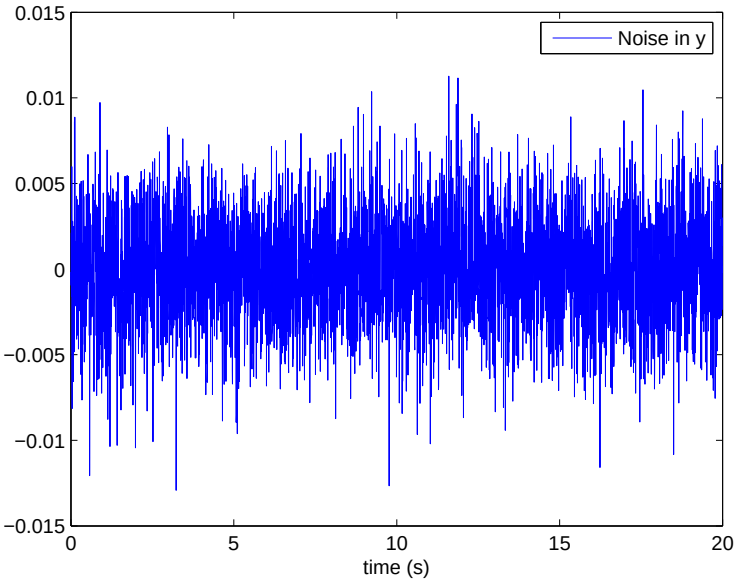


Figure 2.10: Noise imposed in y direction

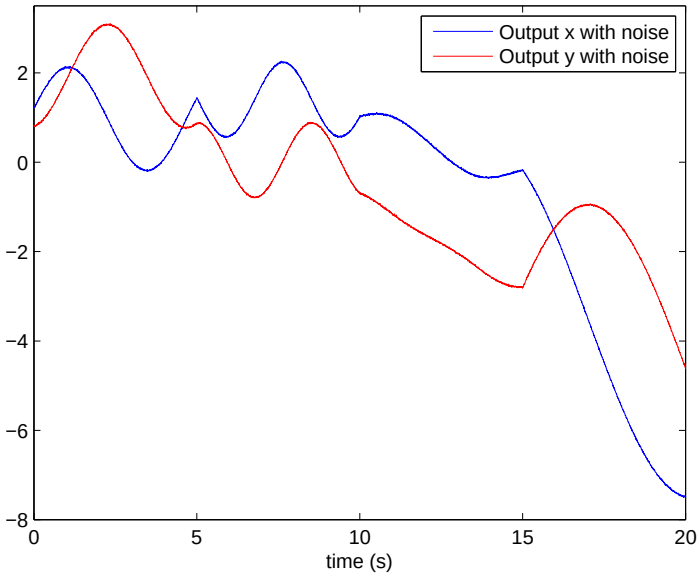


Figure 2.11: output with noise

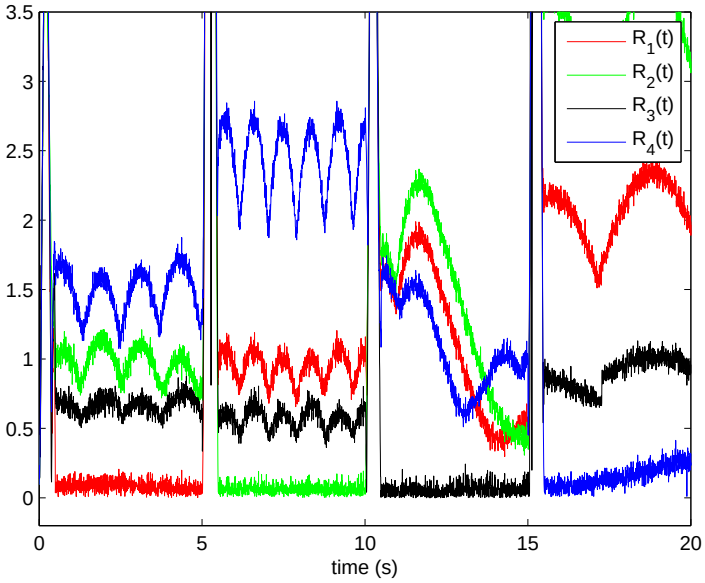


Figure 2.12: Residuals with white noise SNR=50dB

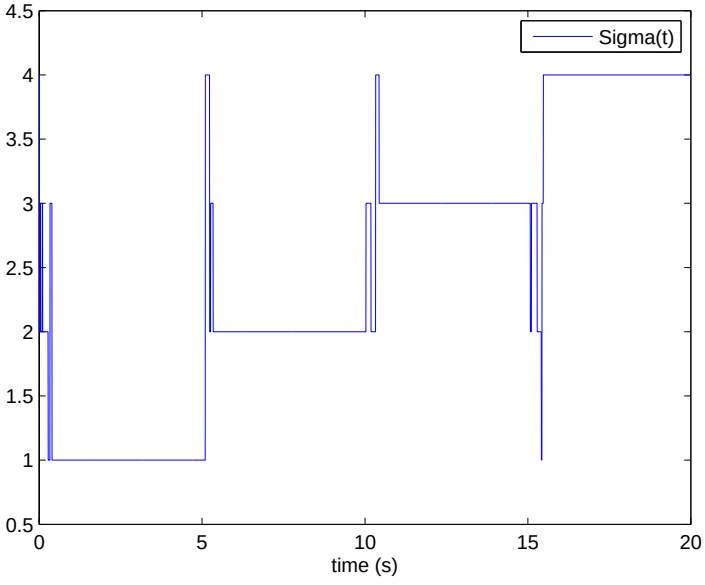


Figure 2.13: Switching signal $\sigma(t)$ identified with white noise SNR=50dB

Chapter 3

Real-time local path planning for non-holonomic mobile robots

3.1 Introduction

Path planning is quite important since it enables the selection and the identification of a suitable path for robots to traverse in the environments.

When considering the path planning problem for unicycle-like mobile robots, the physical limitations and the kinematic constraints have to be taken into account. Previous work Defoort *et al.* (2009) describes the path planning problem as a nonlinear optimal problem with constraints, which guarantees the navigation of the robot in unknown environments. However, it simply represents the obstacles by circles, and there are at least two drawbacks for these algorithms: firstly only the circular obstacles are taken into account, and secondly local minima cannot be avoided when robots getting closed to the complex obstacles. Therefore, this algorithm is not suitable for a complex environment with different shapes of obstacles. An extended algorithm is proposed in Kökösy *et al.* (2008) based on the Tangent Bug algorithm Kamon *et al.* (1996) to treat this problem by following the boundaries of obstacles, which however involves unnecessary detours along the obstacle boundaries and leads to non optimal trajectories.

In this chapter, the irregular contours of obstacles are represented by segments. The path planning problem for unicycle-like mobile robots is described as an optimal control problem by involving all physical constraints. Local minima are avoided by choosing intermediate objectives based on the real-time environment.

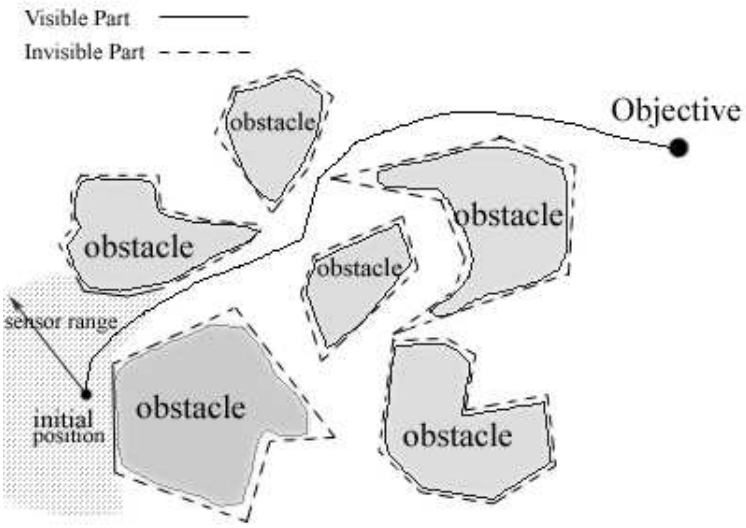


Figure 3.1: Description of the environment

3.2 Path planning: an optimal control point of view

3.2.1 Problem statement

In general cases, the environment may be complex and normally obstacles cannot be described as circles, as assumed in Defoort *et al.* (2009) (see Fig. 3.1 for example). Moreover, due to the distance limitation of sensors equipped on robots, only a portion of an obstacle can be captured, so that the robot may not know the exact shape of the obstacle. In this case, obstacles can neither be described as circles nor be described as complete polygons.

As a result, our goal is to represent obstacles in a more accurate way, and to propose an efficient path planning algorithm which guarantees the safe navigation of robot from a known initial position to a desired target in unknown environments while satisfying the physical constraints of the robot.

3.2.2 Mobile robot model

As stated in Chapter 2, the path planning algorithms for non-holonomic mobile robots are based on robot kinematic models. Since we consider the robot path

planning as an optimal control problem with constraints, we need to choose flat outputs such that all the system states can be expressed by the flat outputs and their successive time derivatives, thus the constraints of the robot can be considered as the constraints of the flat outputs, and the optimal control problem of the robot can be considered as an optimal problem of the flat outputs, which is described in the following sections. The flat outputs are different for different robot models, once the the robot model is identified, the flat outputs can be decided.

This chapter considers the type (2.0) robot, let us firstly recall the type (2.0) robot kinematic model:

$$\begin{cases} \dot{x} &= \upsilon \cos\theta \\ \dot{y} &= \upsilon \sin\theta \\ \dot{\theta} &= \omega \end{cases} \tag{3.1}$$

where υ and ω are the linear and angular velocity respectively, θ is the orientation of the robot body with respect to x-axis, $U = [\ \upsilon, \omega\ ]^T$ is the control inputs, and $q = [x, y, \theta]^T$ is the system state (see Fig. 2.2).

3.2.3 Optimal control problem

3.2.3.1 Nonlinear optimization problem formulation

As mentioned before, the path planning problem for mobile robots with physical constraints can be formulated as an optimal control problem. Generally speaking, it is to find the optimal control $U = [v, \omega]$ for system (3.1) and to minimize the following cost function:

$$J = \int_{t_0}^{t_f} F(U(t), q(t), t)\ dt \tag{3.2}$$

where t_0 and t_f are the initial time and the final time respectively, $U = [v, \omega]$ and $q = [x, y, \theta]$. F is a function of U and q which defines the cost function to be minimized. F can be chosen in advance, and can take several different forms. For example, when $F = 1$, i.e. to minimize the time $t_f - t_0$, it implies that the robot reaches the target as fast as possible. In this chapter the cost function is chosen as follows to guarantee the robot moving towards the objective:

$$F = ((x(t) - x_f)^2 + (y(t) - y_f)^2) \tag{3.3}$$

where (x_f, y_f) is the desired final position.

Moreover, the expected optimal control and the resulting states should satisfy the following physical constraints:

C_1 : the constraint on optimal control and state, i.e. the optimal control U and the states q should satisfy the kinematic model (3.1) for $t \in [\ t_0, t_f\ ]$.

C_2 : the constraint on initial and final conditions, i.e.

$$q(t_0) = q(0),\ q(t_f) = q_{final}$$

C_3 : the constraint on boundedness of control, i.e.

$$\mid \upsilon \mid \leq \upsilon_{max} \text{ and } \mid \omega \mid \leq \omega_{max}$$

C_4 : the constraint on collision avoidance, i.e.

$$d(O, R) \geq r$$

where $d(O, R)$ is the distance between the robot and any obstacle, and r is the given distance which guarantees the obstacle avoidance criterion.

3.2.3.2 Receding horizon planner

When the map is large, or is partially known, it is impossible to solve the above optimal control problem to obtain the whole optimal trajectory. In order to avoid this problem, the receding horizon planner Mayne & Michalska (1990) can be used to compute only a part of the trajectory from the current position to the final one over a time interval $[\tau_k, \tau_k + T_c]$, where T_c is the update period, and $0 < T_c < T_p$, where T_p is the trajectory planning horizon.

As shown in Fig. 3.2, the robot only computes a trajectory of horizon T_p and updates at each step $\tau_k = \tau_{initial} + kT_c$.

In order to simplify the optimal problem described above, we introduce as well the following two elements: the flatness properties of the systems and parameterized trajectory, proposed in Defoort *et al.* (2009).

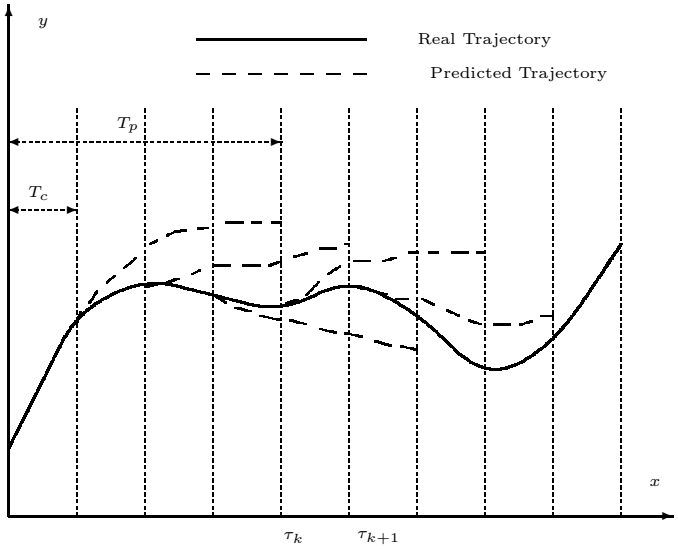


Figure 3.2: Planning and update horizons

3.2.3.3 Determination of the flat outputs

Definition 3.2.1. *Fliess* et al. *(1995) Let a system be described by the nonlinear equation $\dot{q} = f(q, U)$, where $q \in \mathbb{R}^n$ and $U \in \mathbb{R}^m$. This system is differentially flat if there exists a vector $z = [z_1, ..., z_m]^T \in \mathbb{R}^m$ called flat output, such that the state variable q and the input U can be expressed by the flat output and a finite number of its derivatives.*

For the robot describe by equation (3.1), by choosing vector $z = [x, y]^T$, it can be shown that x, y are the flat outputs for the studied system. Indeed, θ, υ and ω can all be expressed by x, y and their first and second-order derivatives:

$$\begin{cases} \theta &= \arctan \frac{\dot{y}}{\dot{x}} \\ \upsilon &= \sqrt{\dot{x}^2 + \dot{y}^2} \\ \omega &= \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2} \end{cases} \tag{3.4}$$

Then the optimal control problem (3.2) with constraints is mapped into the flat output space, and one only needs to optimize x and y to obtain the optimal values of θ, υ and ω.

Remark 3.2.1. *Let us mention that all non-degenerate mobile robots (see Campion* et al. *(1996)) are flat de Wit* et al. *(2001). Thus the proposed method works for all non-degenerate mobile robots. For example, mobile robot of type* (1.1) *described by equation* (2.4) *is flat, with the flat output being $z = [q_1, q_2]^T$.*

3.2.3.4 Parameterized trajectory

In order to transform the optimal trajectory generation problem into a nonlinear parameter optimization problem, the trajectory of the robot can be approximated by finite dimensional curves, which can be expressed as a function of t as follows:

$$\begin{cases} x(t) &= \sum_{i=0}^{N} c_i B_i(t) \\ y(t) &= \sum_{i=0}^{N} d_i B_i(t) \end{cases} \tag{3.5}$$

where $t \in [0, T_p]$, T_p is the planning period, $c_i, d_i \in \mathbb{R}$ are the coefficients, and $B_1, B_2, ...B_N$ are sequences of N functions of t used to approximate $x(t)$ and $y(t)$, which can take different forms.

A finite order polynomial $B_i(t) = t^i$ would be necessary to satisfy the constraints as in Guo & Tang (2008), and the simplicity of the polynomial is beneficial in many aspects.

A solution of B-spline functions can also be applied Defoort *et al.* (2009), when

$$B_{i,0}(t) = \begin{cases} 1 & if\ nod_i \leq t < nod_{i+1}, \\ 0 & otherwise. \end{cases}$$

$$B_{i,d}(t) = \frac{t - nod_i}{nod_{i+d+2} - nod_i} B_{i,d-1}(t) + \frac{nod_{i+d+1} - t}{nod_{i+d+1} - nod_{i+1}} B_{i+1,d-1}(t)$$

where d is the order of the B-spline function, and nod_i is the control knots divided over the time interval $[\tau_k, \tau_k + T_p]$. B-spline is very efficient for curve approximation in terms of both approximation quality and computational time, and it features interesting properties such as continuity, robustness and flexibility Delahaye *et al.* (2010). A comprehensive list of B-spline properties can be found in De Boor (2001).

There are also other methods for trajectory representation, a comparison can be found in Sillito & Fisher (2009).

The optimal coefficients c_i and d_i can be numerically found using the constrained feasible sequential quadratic optimization algorithm Lawrence *et al.* (94), then the open loop control $U = [\upsilon, \omega]^T$ is deduced by using equation (3.4).

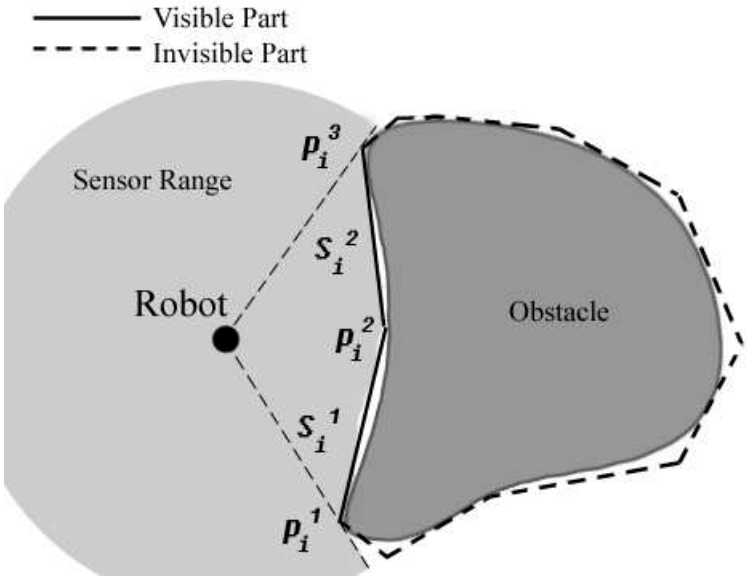


Figure 3.3: Approximation of obstacles with complex shape

3.3 Path planning algorithm with intermediate objectives

3.3.1 Representation of obstacles

In real situations, as stated in section 3.2.1, the obstacles can be described neither as circles nor as complete polygons.

Since a robot can only see a portion of an obstacle contour, as shown in Fig 3.3, the visible portion of the i^{th} obstacle contour can be approximated by a succession of segments S_i^j, where $j = 1, 2, ..., q$, and q is the number of segments on an obstacle. Each segment is represented by its two end points p_i^j and p_i^{j+1}, and each point has their coordinates $(x_{p_i^j}, y_{p_i^j})$ and $(x_{p_i^{j+1}}, y_{p_i^{j+1}})$ respectively. The functions of the segments of obstacle contours can be obtained by applying the image processing algorithms Graham (1972), which is beyond the scope of our discussion. Thus, we assume that the irregular obstacles are represented by a serial of segments.

Remark 3.3.1. *If the distance between two obstacles $d_{obs} < 2r$, where r is the given distance which guarantees the obstacle avoidance criterion, then we can*

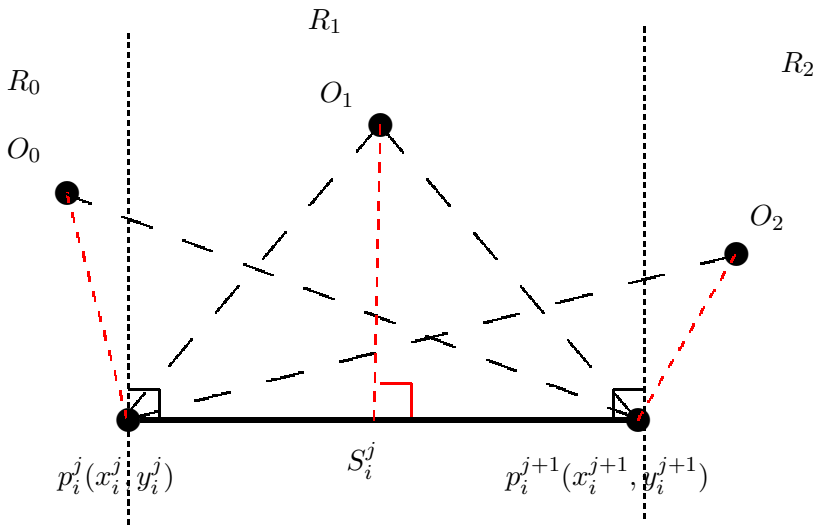


Figure 3.4: The three cases for distance calculation

consider the two obstacles as one obstacle, since the robot can not pass through the space between the two obstacles.

3.3.2 Distance between robot and segments

Since obstacles are represented by segments, the obstacle avoidance constraint C_4 in the optimal control problem (3.2) becomes the distance constraint between the robot and segments.

Denote $O(x_o, y_o)$ the robot position, $p_i^j(x_i^j, y_i^j)$, $p_i^{j+1}(x_i^{j+1}, y_i^{j+1})$ the two end points of segment S_i^j. Then one can define the distances between those points:

$$\begin{aligned} d(O, p_i^j) &= \sqrt{(x_o - x_i^j)^2 + (y_o - y_i^j)^2} \\ d(O, p_i^{j+1}) &= \sqrt{(x_o - x_i^{j+1})^2 + (y_o - y_i^{j+1})^2} \\ d(p_i^j, p_i^{j+1}) &= \sqrt{(x_i^j - x_i^{j+1})^2 + (y_i^j - y_i^{j+1})^2} \end{aligned}$$

Thus, the distance between the robot and the segment S_i^j, noted as $d(O, S_i^j)$, can be calculated according to the relative position of the robot and the segment. There are three possible cases (see O_0, O_1, O_2 in Fig. 3.4) :

Case 1. *$d(O, p_i^{j+1})^2 > d(O, p_i^j)^2 + d(p_i^j, p_i^{j+1})^2$. In this case the robot locates in the left region R_0 of S_i^j. It is easy to see that $d(O, S_i^j) = d(O, p_i^j)$, which is the red dotted line in R_0.*

Case 2. *$d(O,p_i^j)^2 > d(O,p_i^{j+1})^2 + d(p_i^j,p_i^{j+1})^2$. In this case the robot locates in the right region R_2 of S_i^j. It is obvious that $d(O,S_i^j) = d(O,p_i^{j+1})$, which is the red dotted line in R_2.*

Case 3. *If not in case 1 and case 2, the robot will be in region R_1. The distance between the robot and the segment $d(O,S_i^j)$ can be obtained by simply using Heron's formula. A straightforward computation yields:*

$$d(O,S_i^j) = 2\frac{\sqrt{M(M-d(O,p_i^j))(M-d(O,p_i^{j+1}))(M-d(p_i^j,p_i^{j+1}))}}{d(p_i^j,p_i^{j+1})}$$

where $M = \frac{d(O,p_i^j)+d(O,p_i^{j+1})+d(p_i^j,p_i^{j+1})}{2}$. See the red dotted line in R_1.

Summary, the distance between the robot and segment S_i^j is determined by the following equation:

$$d(O,S_i^j) = \begin{cases} d(O,p_i^j), \ \textit{case 1} \\ d(O,p_i^{j+1}), \ \textit{case 2} \\ 2\frac{\sqrt{M(M-d(O,p_i^j))(M-d(O,p_i^{j+1}))(M-d(p_i^j,p_i^{j+1}))}}{d(p_i^j,p_i^{j+1})}, \ \textit{case 3} \end{cases}$$

One can notice that $d(O,S_i^j)$ is not differentiable at the points between the pieces, but it is piecewise differentiable, therefore it will not introduce problems to the optimal control problem.

However, it will be explained in the next section that this algorithm of using segments to represent obstacles suffers from local minima problems.

3.3.3 Local minima

It is worth noting that using of segments to represent the obstacle contour inevitably involves the local minima problems. This phenomenon happens when the robot arrives a point where the distance between the robot and the objective is minimum under the constraint of obstacle avoidance.

As shown in Fig. 3.5, S_i^1, S_i^2 and S_i^3 are segment obstacles, the robot gets to the local minima point $O(x,y)$, r is the obstacle avoidance criterion. The robot needs to go left or right to avoid the segment obstacles, however, no matter the robot moves to left side or right side of point $O(x,y)$, the cost function

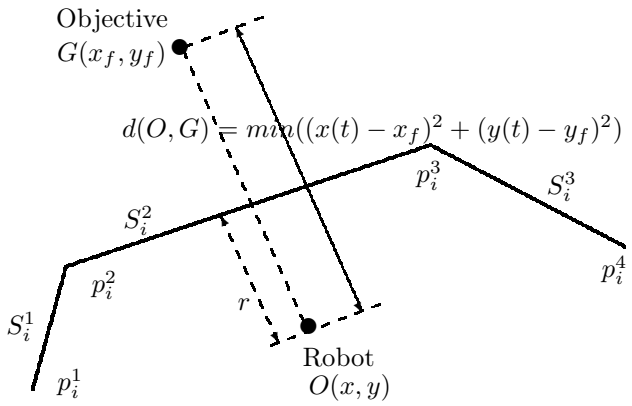


Figure 3.5: Local Minima

$((x(t) - x_f)^2 + (y(t) - y_f)^2)$ in optimization problem (3.2) will become larger, thus the robot will stop at this point.

3.3.4 Avoidance of local minima by choosing intermediate objectives

This local minima problem cannot be avoided by the optimal path planning algorithm stated above. However one can notice that the local minima might occur when the connection between the current robot position and the objective crosses with the segment (see in Fig. 3.5 the segment GO crosses S_i^2). As a result, one can introduce some intermediate objectives for the robot if these intermediate objectives can guide the robot to escape the local minima and to achieve the final objective.

Generally, the selection of the intermediate objectives is according to the information detected by the sensor equipped on the robot. Once the intermediate objectives are chosen, optimal path planning algorithm can be then used to calculate optimal trajectory between the current position of the robot and the intermediate objectives without the local minima phenomenon.

For example, in Fig. 3.5, one can choose the intermediate objectives $\{p_i^1, p_i^2, G\}$ instead of the final objective $\{G\}$, navigating the robot to reach p_i^1, then p_i^2 and finally G. Then the optimal sub trajectories: $O \to p_i^1$, $p_i^1 \to p_i^2$ and $p_i^2 \to G$ can be calculated by solving the optimal control problem stated in section 3.2. It can

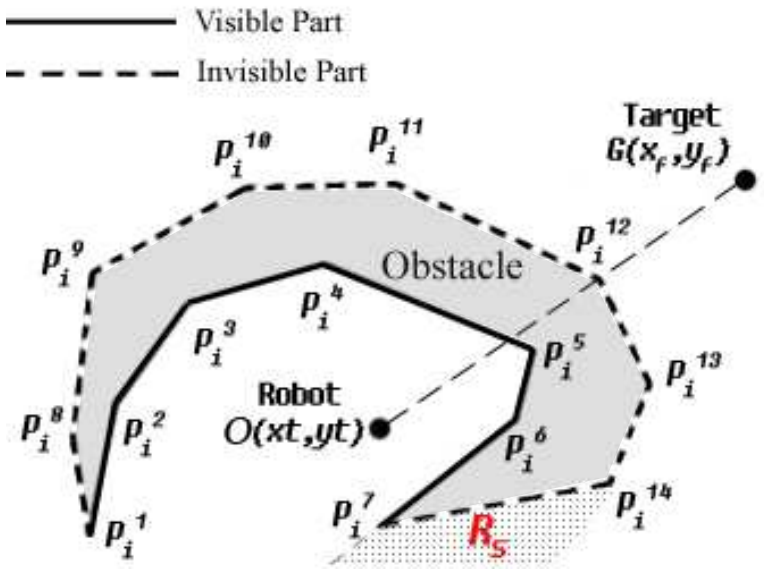


Figure 3.6: Complex Environment

be seen that if the robot follows this way, there is no local minima phenomenon.

3.3.5 Path planning algorithm with intermediate objectives

The "following the obstacle boundary mode" proposed in Kökösy *et al.* (2008) can avoid the obstacles without local minima, but the robot needs to unnecessarily detour along with the contour of polygons. For example in Fig. 3.6, robot need follow the contours $\{p_i^4, p_i^3, p_i^2, p_i^1\}$ or $\{p_i^5, p_i^6, p_i^7\}$ to get outside of the concave obstacle. The proposed algorithm in this chapter takes into account only the disjoint endpoints (the head p_i^1 and the tail p_i^7) of a serial of joint segments which is used to represent the detected partial obstacle.

The procedure of the proposed algorithm with intermediate objectives can be illustrated in Fig. 3.6. At the first time, the robot detects via sensors a serial of joint segments: $\{p_i^1, \cdots, p_i^7\}$ around its local environment. Since the dotted part $\{p_i^8, \cdots, p_i^{14}\}$ is invisible for this moment, the robot assumes that there are no obstacles in the invisible part, thus it thinks the obstacle is only $\{p_i^1, \cdots, p_i^7\}$. Then the robot chooses a temporary set of intermediate objectives in order to avoid the local minima, noted as $IO\_List = \{p_i^7, p_i^6, p_i^5, G\}$. The robot gets the

head of IO_List, i.e. p_i^7, then it generates an optimal sub trajectory $O \rightarrow p_i^7$ by solving optimal control problem defined in section 3.2. When the robot arrives in the region R_s, i.e. the region where robot can always see the second element in IO_List, p_i^6 in this scenario, we remove the reached point p_i^7 into a close list, noted as $CloseList \leftarrow p_i^7$. Robot scans again its surrounding and detects new obstacles represented by $\{p_i^7, p_i^{14}\}$. Since the head point p_i^7 belongs to $CloseList$ which means robot has already reached this point, thus the intermediate objectives should be deduced from the tail point p_i^{14}, which updates the temporary list of intermediate objectives by: $IO\_List = \{p_i^{14}, G\}$. Finally the robot can reach G by following this list.

Remark 3.3.2. *Although the optimization method can be applied to any kind of mobile robots to drive the robot from one point to another one, however the approach described in this chapter might not be applicable for general non-holonomic robot (like car-like mobile robot) if the approaching direction to the intermediate point is not considered. The reason is that, after achieving the intermediate point, some kinds of robots will not have enough space to turn (due to the non-holonomic constraint) in order to achieve the next trajectory. However, the unicycle model considered here has not such a problem since it allows turning in space.*

From the above description, the proposed algorithm contains the following three aspects:

1. Select the intermediate objectives from local information to generate a temporary list IO_List;
2. Be sure that the robot can reach another region (for example, R_s in Fig. 3.6);
3. Judge when the robot arrived at this region.

Before explaining these three aspects, let us give some notations which will be used in the sequel. Define $\mathbb{P} = \{P_i\}$ for $1 \leq i \leq N$ to be all the set of the obstacle boundaries detected by the equipped sensors, where N is the number of detected obstacle boundaries. Note $P_i = \{p_i^j\}$ for $1 \leq j \leq N_i$ being the set of joint points to represent the i^{th} obstacle boundary, where N_i is the number of points. Each segment S_i^j is defined by its endpoints $S_i^j = (p_i^j, p_i^{j+1})$. Let $IO\_List = \{p_k, G\}$ for $1 \leq k \leq m$ save the selected intermediate objectives. Denote $dis(p_k p_{k+1})$ for

$1 \leq k \leq m-1$ the function to calculate the distance between point p_k and point p_{k+1}, and note

$$dis(\{O\} \cup IO\_List) = dis(Op_1) + \sum_{k=1}^{m-1} dis(p_k p_{k+1}) + dis(p_m G)$$

as the function to compute the complete path cost from robot's current position O to the final target G by following IO_List.

For the reached intermediate points which have already treated, we removed them from IO_List and save them into a $CloseList$ which is used to avoid unnecessary returns. Thus for an endpoint belonging to $CloseList$, the path cost from this path is set to be $+\infty$. $List\_H$ and $List\_T$ are defined to save two possible lists of intermediate objectives from the head and the tail of IO_List, being initialized as $List\_H = List\_T = \{G\}$.

3.3.5.1 The intermediate objectives selection

Whenever the robot detects several segments obstacles around its surrounding, it always chooses the one with which the begin-final segment OG has an intersection. If OG has no intersection with all obstacles, then the robot can see the target G directly and thus the optimal path is the straight line OG. Otherwise, OG can have only one intersection with all obstacles, since it can detect only visible part of obstacles. For example, in Fig. 3.7, OG intersects with P_i, and the possible optimal trajectory might from p_i^1 or p_i^5, but it is absolutely not possible from the obstacles P_{i+1} or P_{i+2} since those paths from P_{i+1} or P_{i+2} are obviously larger than the ones from P_i.

After determining the exact obstacle (In Fig. 3.7, it is P_i since segment $p_i^2 p_i^3$ in $P_i = \{p_i^1, \cdots, p_i^5\}$ intersects with OG), we search the intermediate objectives from both sides of OG. Take the right region of OG for example, one has the segment $p_i^3 p_i^4$, and check whether the segment Gp_i^4 has an intersection with P_i. If not, that means the robot can see the final objective G after passing over the point p_i^4, thus the point p_i^3 does not need to be added on $List\_T$. If segment Gp_i^4 has an intersection with P_i, which implies the robot is not able to see G after passing over p_i^4, thus the robot needs to go to point p_i^3 in order to see G. Consequently, p_i^3 should be added on $List\_T$. Iteratively search next segment

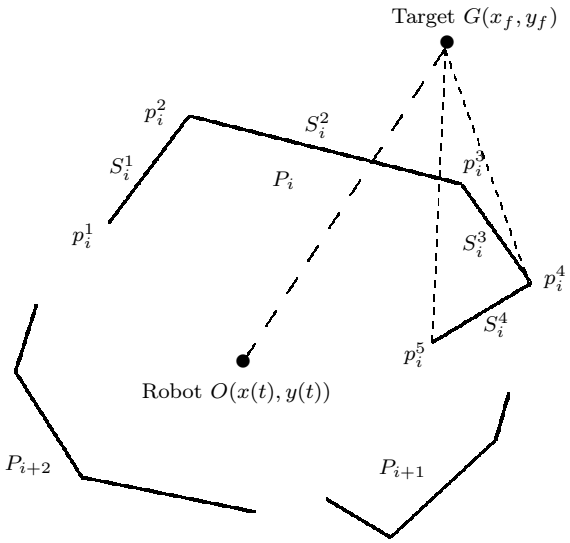


Figure 3.7: Intermediate Objectives Generation

($p_i^4 p_i^5$ in Fig. 3.7) until the end of the segment of P_i, one obtains

$$List\_T = \{p_i^5, p_i^4, G\}$$

By applying the same procedure for the left region of OG, one gets

$$List\_H = \{p_i^1, p_i^2, G\}$$

Finally, the temporary list of intermediate objectives is then determined by the path cost of these two lists, i.e. if

$$dis(O, List\_T) > dis(O, List\_H)$$

then $IO\_List = List\_H$. Otherwise $IO\_List = List\_T$

The routine to generate the list of intermediate objectives is given in Algorithm 1 in section 3.6.

3.3.5.2 Reach switching region

In order to clearly explain the algorithm, let consider the following simple segment obstacle depicted in Fig. 3.8, and suppose that one has obtained the following

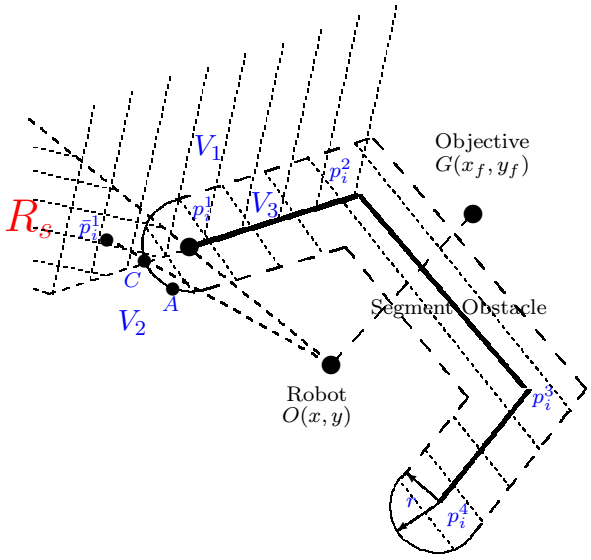


Figure 3.8: Intermediate Objectives Selection

list of intermediate objectives:

$$IO\_List = \{p_i^1, p_i^2, G\}$$

Thus robot is guided to reach the first element in IO_List, i.e. p_i^1, then p_i^2 and finally G. Then one can solve the optimal problem with constraints $C_1 - C_4$ by minimizing the cost function with respect to current intermediate objective, i.e.

$$\min \int_{T_p} \left\| O - p_i^1 \right\|^2 dt, \text{ s.t. } C_1 - C_4 \tag{3.6}$$

The solution to this optimal problem yields optimal trajectories, then one can get optimal control (v, ω) according to (3.4).

However, choosing directly p_i^1 as final target in cost function (3.6) result in local minima again. For example, as shown in Fig. 3.8, where r is the collision free distance, the point A is on the boundary satisfying the obstacle avoidance criterion. In this situation, the robot may stop at point A since no other trajectories are more optimal than to stay at this point (it has already minimized the cost function (3.6) and satisfied as well $C_1 - C_4$). Finally, the robot cannot pass over the segment $p_i^1 p_i^2$ to see the second intermediate objective p_i^2.

In order to make sure that the robot can always achieve this, let us give the following notations. Denote V_1 the top region of line $p_i^1 p_i^2$ (the region where the second intermediate objective p_i^2 is visible), and V_2 the left region of line Op_i^1 (the region where robot can reach freely). Define V_3 the collision constraint region:

$$V_3 = \{(x, y) : (x - x_i)^2 + (y - y_i)^2 \leq r^2, \forall (x_i, y_i) \in p_i^1 p_i^2\}$$

Then define the switching region R_s as follows:

$$R_s = (V_1 \cap V_2) \cap \overline{V}_3$$

where $\overline{V}_3$ is the complement of V_3.

As the switching region is defined, one can see that the optimal path for the robot is to go directly into the switching region, as a result one can choose $\bar{p}_i^1 \in R_s$ and replaces p_i^1 by $\bar{p}_i^1$ in (3.6) to ensure the robot go into the switching region as directly as possible, and avoid detours around the endpoint of the obstacles, thus this optimal problem can be solved without local minima, since the robot can always pass over the segment $p_i^1 p_i^2$ to see the second intermediate objective p_i^2.

Here the modified intermediate objective $\bar{p}_i^1$ is determined as follows (see Fig. 3.8): firstly find out the point C at a distance of r to the point p_i^1 on the extension of segment from the endpoint p_i^2 to the endpoint p_i^1, and then select $\bar{p}_i^1$ at a distance of r to the point C on the extension of segment from the robot to the point C.

It is worth noting that the connection between the modified intermediate point $\bar{p}_i^1$ and robot position may crosses with another obstacle, if the line connected between $\bar{p}_i^1$ and robot crosses with another obstacle, the intermediate objective should be selected from the obstacle that crosses with the line, add the new intermediate objective to the IO_List, and generate new modified intermediate objective $\bar{p}_i^1$ from new IO_List. (See line 10-16 in Algorithm 4)

The routine to select intermediate objectives is given in Algorithm 2 in section 3.6.

3.3.5.3 Judge the switching time

Suppose that one has $IO\_List = \{p_i^1, p_i^2, \cdots, G\}$ and the associated switching region R_s. Since the robot reinitializes and solves the optimal problem after every T_c, thus one can use the position of robot at $t = 0$ and $t = T_c$ (named as $(x(0), y(0))$ and $x(T_c), y(T_c)$) to judge whether it enters R_s. For this, note the function $f(x, y) = 0$ representing the first segment $p_i^1 p_i^2$ in IO_List. If

$$f(x(0), y(0)) f(x(T_c), y(T_c)) < 0$$

one can judge that the robot has already entered in the region R_s, implying that the robot has passed over p_i^1 and now can see p_i^2. Thus we move p_i^1 from IO_List into $CloseList$.

The routine to judge the switching time is given in Algorithm 3 in section 3.6.

3.3.6 Algorithm description

Given the temporary list of intermediate objectives IO_List, which enables to define the switching region R_s and calculate the modified intermediate objective $\bar{p}_i^1$, one can solve optimal problem over T_p to get optimal trajectories, which yields the optimal controls v and ω for robot over $[0, T_p]$. Then one applies those optimal controls only for an interval $[0, T_c]$. When $t = T_c$, one iterates the same procedure as before, i.e., scans the surroundings to get segments obstacles $\mathbb{P}$, generates the list of intermediate objectives IO_List, calculates the modified intermediate objective $\bar{p}_i^1$ and solves the optimal problem over T_p and implements the optimal controls for $[0, T_c]$. The algorithm stops when the robot reaches the final target G. The routine is detailed in Algorithm 4 in section 3.6.

3.4 Simulation results

In order to show the feasibility and the efficiency of the proposed algorithm, three simulations for different scenarios are made, and the comparisons with visibility graph with expanded obstacles are made in the latter two simulations. The simulation settings are as follows: the range of robot sensors is 3 m; the maximum speed of robot is 1.0 m/s, the maximum acceleration is 1.0 m/s^2, the

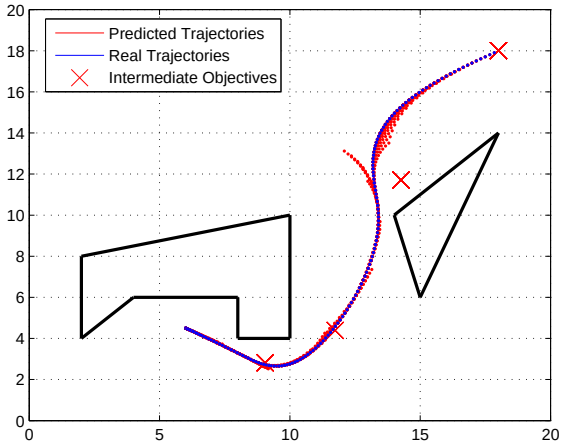


Figure 3.9: Scenario 1: Simple environment

maximum angular velocity is 1.0 rad/s, the maximum angular acceleration is 1.0 rad/s^2. The planning horizon interval T_p is 2 s, and the update period T_c is $0.2s$.

For the simple scenario, depicted in Fig. 3.9, black polygons represent obstacles, containing a concave obstacle and a triangle obstacle. In this scenario, there exists a broad zone of local minima. The robot starts from the initial point $(6, 4.5)$ to the target $(18, 18)$. The red crosses shown in the figure are the intermediate objectives chosen by the proposed algorithm, the red trajectories are the predicted ones planned by the receding horizon planner, and the blue trajectories are real trajectories. It can be seen that the proposed algorithm generates a safe and optimal path of intermediate objectives and avoids the local minima successfully.

Two more complex scenarios are shown in Fig. 3.10 and Fig. 3.11, arrows in the figures indicate the orientation of the robot, and the red crosses shown in the figure are the intermediate objectives chosen by the proposed algorithm. Comparisons with visibility graph with expanded obstacles are made, the blue trajectories are generated by the path planning algorithm proposed in this chapter, and the pink ones are generated by visibility graph.

One can see that in Fig. 3.10 several local minima exist, the robot starts from $(7, 2)$ to $(10, 20)$ and avoids all the local minima by choosing intermediate objectives and reaches the target successfully by using only local sensor informa-

Table 3.1: Comparison of simulation results

	Method	Running Time (s)	Time-saving	Trajectory Length (m)
Scenario 2	Our method	3.02	19.2%	22.2
	Visibility Graph	3.74		22.1
Scenario 3	Our method	3.21	20.1%	30.2
	Visibility Graph	4.02		29.8

tion. In Fig. 3.11, where there is a long winding corridor, the robot starts from $(7, 2)$ to $(25, 10)$. It can be seen that the robot manages to walk through the long corridor and reach the target successfully while avoiding local minima and all the obstacles.

Normally visibility graph is used in global planning when the map is completely known, in order to use visibility graph in local planning with unknown map, the algorithm needs to generate expanded polygon for each obstacle in the local map and search for the shortest path among all the obstacles, then iterate until the robot reaches the target. In our proposed algorithm the robot search for the shortest path only in some obstacles (normally one or two) in each iteration, which reduces the computational complexity compared to visibility graph approach.

As we can see in the figure 3.10, 3.11 and table 3.1, there are no big differences between the trajectories generated by two different methods, however it costs less time by using our method.

Implementations in a wifibot and real environment can be found in the following link: Video Link

3.5 Conclusion

This chapter presents a path planning algorithm for navigation of non-holonomic mobile robots in unknown complex environments. The new algorithm takes into account irregular obstacles which are impossible to be approximated by circles. In order to avoid local minima problems, an algorithm of choosing intermediate objectives is proposed. The robot can reach the target and avoid obstacles by choosing appropriate intermediate objectives. Efficiency of the proposed algorithm is shown thereafter via different simulations and implementations in a

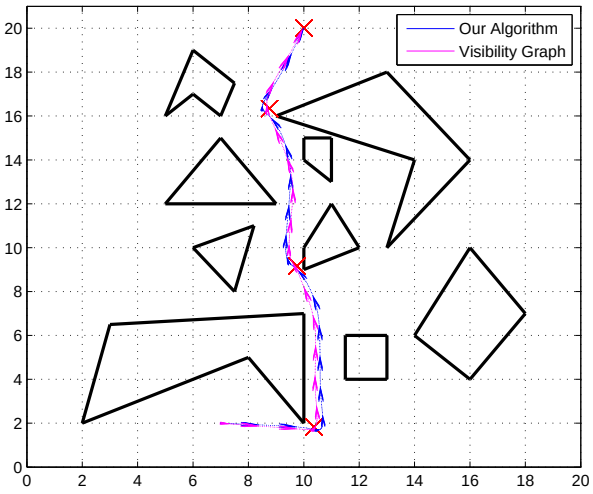


Figure 3.10: Scenario 2: Complex environment

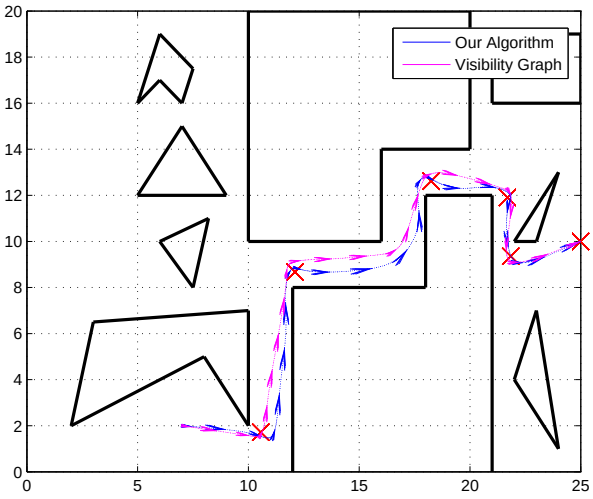


Figure 3.11: Scenario 3: Environment with corridor

wifibot, the simplicity of the algorithm is shown via the comparison with visibility graph.

3.6 Pseudocode

Algorithm 1 The Intermediate Objectives List Generation Function

Function $IO\_Generation\ (G(x_f, y_f), O(x(t), y(t)), \mathbb{P})$
for each P_i **do**
  if $\exists p_i^j, p_i^{j+1} \in P_i$ s.t. $p_i^j p_i^{j+1} \bigcap OG \neq \emptyset$ **then**
    select P_i, break
  end if
end for
$List\_T = List\_H = \{G\}$
for k=j+1: 1:m-1 **do** ▷ m is the number of points on P_i
  if $Gp_i^{j+2} \bigcap P_i \neq \emptyset$ **then**
    $List\_T = \{p_i^{j+1}\} \bigcup List\_T$
  end if
end for
$List\_T = \{p_i^m\} \bigcup List\_T$
for k=j:-1:2 **do**
  if $Gp_i^{j-1} \bigcap P_i \neq \emptyset$ **then**
    $List\_H = \{p_i^j\} \bigcup List\_H$
  end if
end for
$List\_H = \{p_i^1\} \bigcup List\_H$
if $p_i^m \in CloseList$ **then**
  $dist(List\_T) = +\infty$
end if
if $p_i^1 \in CloseList$ **then**
  $dist(List\_H) = +\infty$
end if
if $dist(\{O\} \bigcup List\_T) \leq dist(\{O\} \bigcup List\_H)$ **then**
  Return $List\_T$
else
  Return $List\_H$
end if

Algorithm 2 Selection of the Intermediate Objective

Function $IO\_Selection\ (O(x(t), y(t)), IO\_List)$
Get the first segment $p_i^1 p_i^2$ from IO_List
Get the function $f(x, y) = 0$ for this segment;
Compute point C s.t. $f(x_c, y_c) = 0$, $dis(Cp_i^1) = r\ dis(Cp_i^2) = r + dis(p_i^1 p_i^2)$;
Determine the function $g(x, y) = 0$ for the segment OC;
Select $\bar{p}_i^1$ s.t. $g(x_{\bar{p}_i^1}, y_{\bar{p}_i^1}) = 0$, $dis(C\bar{p}_i^1) = r\ dis(O\bar{p}_i^1) = r + dis(OC)$;
Return $\bar{p}_i^1$

Algorithm 3 Switching Time

Function $Switch\ ((x(0), y(0)), (x(T_c), y(T_c)), IO\_List)$
Get the first segment $p_i^1 p_i^2$ from IO_List
Get the function $f(x, y) = 0$ for this segment;
if $f(x(0), y(0)) \times f(x(T_c), y(T_c)) < 0$ **then**
    Remove the 1st element p_i^1 from IO_List
    Add p_i^1 into $CloseList$
end if

Algorithm 4 Path Planning

Function $PathPlanning(G(x_f, y_f)$, Ini. conditions)
$t = 0$
while $(x(0) - x_f)^2 + (y(0) - y_f)^2 \geq \varepsilon$ **do**
    Get $\mathbb{P}$ from sensor
    $IO\_List = IO\_Generation(G(x_f, y_f), O(x(0), y(0)), \mathbb{P})$
    if $IO\_List = \{G\}$ **then** ▷ Can see G
        [x,y] = $Optimisation(G)$ over T_p
    else
        Get $\bar{p}_i^1 = IO\_Selection(IO\_List)$ ▷ Opt. with the 1^{st} intermediate objective
        for each P_i **do** ▷ Check $\bar{p}_i^1$ can be seen or not
            if $\exists p_i^j, p_i^{j+1} \in P_i$ s.t. $p_i^j p_i^{j+1} \bigcap \bar{p}_i^1 O \neq \emptyset$ **then**
                $add\_List = IO\_Generation(\bar{p}_i^1, O(x(0), y(0)), \mathbb{P})$
                $IO\_List = add\_List \cup IO\_List$
                Get $\bar{p}_i^1 = IO\_Selection(IO\_List)$
            end if
        end for
        [x,y] = $Optimisation(\bar{p}_i^1)$ over T_p
    end if
    for $t \in [0, T_c]$ **do**
        Get (v, ω) from (3.4) based on x and y
        Apply (v, ω) to the robot
    end for
    $Switch(x(0), y(0), x(T_c), y(T_c), IO\_List)$
    Reset $t = 0$
end while

Chapter 4

Control of non-holonomic wheeled mobile robots via i-PID controller

4.1 Introduction

In Chapter 3 we have obtained an optimal trajectory, and this chapter aims at controlling the robot to track the reference trajectory. The trajectory tracking problem is to determine a control to asymptotically stabilize the tracking error.

However as stated in section 1.2.1, according to the Brockett's necessary condition, the non-holonomic mobile robot systems are not asymptotically stabilizable by continuous state feedback. This has led to the development of other control strategies to solve this asymptotic stabilization problem. The use of time-varying continuous state feedback control for asymptotic stabilization of a mobile robot was originally developed by Samson Samson (1991), and a first result of the existence of such stabilizing control is given in Coron (1992). Thereafter, various approaches were developed to asymptotically stabilize the non-holonomic mobile robot, such as the methods based on the direct Lyapunov method Pomet (1992); Samson (1995), and the method based on the backstepping techniques Jiang & Nijmeijer (1999). Although these controllers have relatively good robustness properties, they also have practical disadvantages concerning the convergence rate and the delicate tuning of control parameters. As for the use of discontinues control for the mobile robot, a piecewise continuous control is used in Hespanha *et al.* (1999), and a sliding-mode control based on the transformation of the system into polar coordinates is proposed in Chwa (2004).

The i-PID controller introduced in Fliess & Join (2009) exhibits the robustness to the unmodeled dynamics and disturbance in the system Fliess *et al.* (2011), and it has been widely studied and applied to many electrical and mechanical processes Riachy *et al.* (2011); Villagra & Balaguer (2010). This chapter aims at applying the so-called i-PID controller to the non-holonomic robots in order to control the robot with measurement disturbance. However, due to the particularity of the non-holonomic systems, this controller can not be simply applied, for this a switching parameter is selected and a robust controller is proposed to control the robot in the presence of measurement disturbance.

4.2 Problem statement

In Chapter 3 we have proposed a path planning algorithm for non-holonomic mobile robots, however one can notice that in Chapter 3 section 3.2.3.4, the proposed controller is the open loop control deduced from equation (3.4), which is not robust to errors and noises in the system. Therefore a robust controller is required to control the robot with measurement disturbance.

This chapter still focus on the type (2.0) robot described in (3.1). Suppose that we can only measure the position (x, y) of the robot, which implies that the relative degree of those measurements is equal to 1, since the first part of system (3.1) is of the following form:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

It is clearly that the second control input ω is not involved in the above equation, since the decoupling matrix is singular. Due to this fact, one classic solution is to add an integrator to the first input in order to overcome the singularity of the decoupling matrix Ge (2010). For this, let us consider the following extended system:

$$\begin{cases} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \omega \\ \dot{v} &= \xi \end{cases}$$

with $u = [\xi, \omega]$ being the new input. One can check that, with the extended system, the relative degree for both output is equal to 2. Then one obtains:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = G(x, y, \dot{x}, \dot{y})u \tag{4.1}$$

where

$$\begin{aligned} G &= \begin{bmatrix} \cos\theta & -v\sin\theta \\ \sin\theta & v\cos\theta \end{bmatrix} \\ &= \begin{bmatrix} \cos(\arctan\frac{\dot{y}}{\dot{x}}) & -\sqrt{\dot{x}^2+\dot{y}^2}\sin(\arctan\frac{\dot{y}}{\dot{x}}) \\ \sin(\arctan\frac{\dot{y}}{\dot{x}}) & \sqrt{\dot{x}^2+\dot{y}^2}\cos(\arctan\frac{\dot{y}}{\dot{x}}) \end{bmatrix} \end{aligned} \tag{4.2}$$

which is invertible if $v = \sqrt{\dot{x}^2 + \dot{y}^2} \neq 0$.

If there is no disturbance in the measurement, a classical PID controller, which needs the exact value of G^{-1}, can be used to achieve non-vanishing Cartesian trajectories tracking (the linear velocity of the robot is assumed to be always non-zero), since G is singular when $v = 0$. It has been shown (Datta *et al.* (2000)) that this method cannot be used to stabilize the robot to a static point due to the same reason.

In addition, noises and disturbance are inevitable in real situations, thus the exact computation of G^{-1} in (4.2) cannot be obtained. Consider the output under disturbance as $Y = [x, y]^T + D$, where $D = [d_1, d_2]^T$ is the disturbance in the measurement. Thus the estimated values of θ, v and ω are disturbed:

$$\begin{cases} \theta_d &= \arctan\frac{\dot{y}+\tilde{d}_2}{\dot{x}+\tilde{d}_1} \\ \upsilon_d &= \sqrt{(\dot{x}+\tilde{d}_1)^2 + (\dot{y}+\tilde{d}_2)^2} \end{cases} \tag{4.3}$$

thus we have

$$\ddot{Y} = G(Y, \dot{Y})u + \tilde{D} \tag{4.4}$$

with

$$G(Y, \dot{Y}) = \begin{bmatrix} \cos(\theta_d) & -\upsilon_d\sin(\theta_d) \\ \sin(\theta_d) & \upsilon_d\cos(\theta_d) \end{bmatrix} \tag{4.5}$$

where θ_d and υ_d are defined in (4.3), and $\tilde{d}_1$, $\tilde{d}_2$ and $\tilde{D}$ represent respectively the associated disturbance in the first and second derivatives of the noisy outputs.

It is clear that the system (4.4) can not be controlled with the classical PID controller, since $G(Y, \dot{Y})$ defined in (4.5) can not be accurately estimated due

to the unknown disturbance. Moreover, $G(Y, \dot{Y})$ becomes singular when $v_d = 0$. In order to overcome the two drawbacks when applying the classical PID controller, we use the recently proposed i-PID controller to control the robot with measurement disturbance.

4.3 Determination of the controller

Since the controller proposed here is based on the i-PID controller, let us firstly present the basic idea of this controller, and then detail how to apply this controller into the control of the unicycle model with the measurement disturbance.

4.3.1 i-PID controller

Generally speaking, the method of i-PID controller locally approximates the system model by using a simple local model with an unknown term, and the unknown term can be estimated by the measurements of the input and output of the system, then a so-called i-PID controller can be deduced to realize the control goal.

In this chapter the system model (4.4) is approximated by the following local model over a small time interval $T = [t_k, t_{k+1}]$ with $k \in Z^+$:

$$\ddot{Y}(t) = F(t) + \alpha(Y, \dot{Y})u(t) \tag{4.6}$$

where u and Y are known input and output signals with disturbance, $\alpha(Y, \dot{Y})$ is a non-singular 2×2 dimensional matrix which should be well chosen in order to achieve the control goal. $F \in \mathbb{R}^2$ represents all unknown terms including the disturbances, which can be estimated by using the information of Y, u and α.

For the above-mentioned locally approximated continuous model over time interval T, one can estimate F by discretizing it. Precisely, denote T_s the sampling period, so at each sampling time $k = t/T_s$, one has

$$\ddot{Y}_k = F_k + \alpha(Y_k, \dot{Y}_k)u_k$$

For the sake of simplicity, we denote $\alpha(Y_k, \dot{Y}_k)$ as $\alpha(Y, \dot{Y})$ in the following, then it yields $F_k = \ddot{Y}_k - \alpha(Y, \dot{Y})u_k$, where Y_k and u_k are measurable signals at time k, and $\ddot{Y}_k$ is the 2^{nd} order differentiation of the output Y at sampling time k. If it is

assumed that T_s is small enough such that $F_{k-1} \to F_k$, then the i-PID controller can be designed as follows:

$$u_k = \alpha_k^{-1}(Y, \dot{Y})(-F_{k-1} + e_k) \tag{4.7}$$

where $e_k = \ddot{Y}_{ref,k} - K_2(\dot{Y}_k - \dot{Y}_{ref,k}) - K_1(Y_k - Y_{ref,k})$ with Y_{ref} being the references of the output to be tracked, and K_1 and K_2 being the freely chosen coefficients such that the polynomial $s^2 + K_2 s + K_1$ is Hurwitz.

Remark 4.3.1. *We want to emphasize that this approach can treat not only the kinematic model, but also the dynamic model. The robot dynamic model can be expressed in the following form Rashid & Sidek (2011):*

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) = B(q)u_d \tag{4.8}$$

where $q \in \mathbb{R}^{n\times 1}$ is a vector of generalized coordinates and $\dot{q} \in \mathbb{R}^{n\times 1}$ is a vector of velocities of generalized coordinates. $M(q) \in \mathbb{R}^{n\times n}$ is a symmetric positive definite inertia matrix of the system, $C(q, \dot{q}) \in \mathbb{R}^{n\times n}$ is the centripetal and coriolis matrix. $G(q)$ is the gravitational terms, $B(q) \in \mathbb{R}^{n\times r}$ is the input transformation matrix, and $u_d \in \mathbb{R}^{r\times 1}$ is the input vector.

Equation (4.8) can be rearranged of the following form:

$$\dot{\rho} = f(\rho) + H(\rho)u_d \tag{4.9}$$

where $\rho = [q^T, \dot{q}^T]$, and $f(\rho)$ and $H(\rho)$ are given as:

$$f(\rho) = \begin{bmatrix} \dot{q} \\ -M(q)^{-1}C(q, \dot{q})\dot{q} - M(q)^{-1}G(q) \end{bmatrix}$$
$$H(\rho) = \left[0, (M(q)^{-1})^T\right]^T$$

As one can see from equation (4.9), it will just introduce some additional terms in (4.4), which in fact can be integrated into the term $\tilde{D}$ in (4.4). Therefore, we can use the method proposed in this chapter to treat the stabilization problem for the dynamic model of robot.

As one can see in the controller (4.7), there are two parameters to be determined, $\alpha_k(Y, \dot{Y})$ and F_k, which will be discussed in the following.

4.3.2 Discussion on $\alpha(Y, \dot{Y})$

The determination of $\alpha(Y, \dot{Y})$ is the most important issue when applying such a controller. A good parameter $\alpha(Y, \dot{Y})$ should well approximate $G(Y, \dot{Y})$ defined in (4.5), and be always invertible, and change as few times as possible as time goes on, so it is best that $\alpha(Y, \dot{Y})$ is time-invariant.

In Sira-Ramírez *et al.* (2011) and Youcef-Toumi & Wu (1991), similar controllers are presented, which use an unknown term to represent unknown parameters and disturbance in the system model. However, in Sira-Ramírez *et al.* (2011) the similar parameter $G(Y, \dot{Y})$ in the system is a time-invariant scalar. In Youcef-Toumi & Wu (1991) the determination of $\alpha(Y, \dot{Y})$ is discussed, nevertheless, the similar parameter $G(Y, \dot{Y})$ in the system is assumed to be always invertible and time-invariant. Thus in their controller $\alpha(Y, \dot{Y})$ can be set as a fixed number or a fixed invertible matrix.

We aim at finding out an invertible time-invariant α to well approximate $G(Y, \dot{Y})$ in the controller. However, let us take a look at $G(Y, \dot{Y})$ in our system (4.4): it is a matrix whose entries vary as time goes on, and the sign of all entries in $G(Y, \dot{Y})$ is changing, which makes it impossible to use a time-invariant α to approximate $G(Y, \dot{Y})$.

In order to have $G(Y, \dot{Y})$ well approximated by $\alpha(Y, \dot{Y})$, $\alpha(Y, \dot{Y})$ needs to change according to $G(Y, \dot{Y})$. One can of course set that

$$\alpha(Y, \dot{Y}) = \begin{bmatrix} \cos\hat{\theta} & -\hat{v}\sin\hat{\theta} \\ \sin\hat{\theta} & \hat{v}\cos\hat{\theta} \end{bmatrix}$$

where $\hat{\theta} = \arctan\frac{\dot{\hat{y}}}{\dot{\hat{x}}}$ is the estimation of θ with noises, $\dot{\hat{x}}$ and $\dot{\hat{y}}$ are the estimations of $\dot{x}$ and $\dot{y}$ with noises, and $\hat{v}$ is the estimation of v defined in (3.4). In this way, this method is in fact equivalent to the controller linked to exact linearization by feedback with the estimate of θ and v. However, one can notice that $\alpha(Y, \dot{Y})$ will be singular when $\hat{v} = 0$. In order to make $\alpha(Y, \dot{Y})$ being invertible and well approximate $G(Y, \dot{Y})$, another intuitive choice is to remove $\hat{v}$ in the above matrix and one obtains:

$$\alpha(Y, \dot{Y}) = \begin{bmatrix} \cos\hat{\theta} & -\sin\hat{\theta} \\ \sin\hat{\theta} & \cos\hat{\theta} \end{bmatrix}$$

The above selected $\alpha(Y, \dot{Y})$ is suitable for the controller, since it is always invertible and it can well approximate $G(Y, \dot{Y})$. However, since this choice of $\hat{\theta}$

is always time-varying when robot moves, which will increase the computation complexity of the controller. In order to make the selected $\alpha(Y, \dot{Y})$ change as few times as possible when robot moves, we propose to choose it as follows:

$$\alpha(Y, \dot{Y}) = \begin{bmatrix} sgn(\cos\hat{\theta}) & -sgn(\sin\hat{\theta}) \\ sgn(\sin\hat{\theta}) & sgn(\cos\hat{\theta}) \end{bmatrix}$$

where $sgn(\sigma)$ is the sign function which extracts the sign of the real number σ, and it is assumed that $sgn(0) = 1$. For this proposed $\alpha(Y, \dot{Y})$, let us define the following switching signal $i(\hat{\theta}) : \mathbb{R} \to \mathcal{I}$ with $\mathcal{I} = \{1, 2, 3, 4\}$:

$$i(\hat{\theta}) = \begin{cases} 1 & if\ \hat{\theta} \in (2k\pi, 2k\pi + \frac{\pi}{2}) \\ 2 & if\ \hat{\theta} \in (2k\pi + \frac{\pi}{2}, 2k\pi + \pi) \\ 3 & if\ \hat{\theta} \in (2k\pi + \pi, 2k\pi + \frac{3\pi}{2}) \\ 4 & if\ \hat{\theta} \in (2k\pi + \frac{3\pi}{2}, 2(k+1)\pi) \end{cases} \tag{4.10}$$

where $k \in Z$. The corresponding constant matrices can then be defined as follows:

$$\alpha_1 = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \quad \alpha_2 = \begin{bmatrix} -1 & -1 \\ 1 & -1 \end{bmatrix}$$
$$\alpha_3 = \begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix} \quad \alpha_4 = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

Therefore, the proposed matrix $\alpha_{i(\hat{\theta})}$ satisfies:

$$\alpha(Y, \dot{Y}) = \alpha_{i(\hat{\theta})}, \forall \hat{\theta} \in \mathbb{R}$$

Summarily, the selected $\alpha_{i(\hat{\theta})}$ has several advantages. Firstly it has only 4 values when $\hat{\theta}$ changes in $[2k\pi, 2(k+1)\pi]$, which is able to make $\alpha(Y, \dot{Y})$ change as few times as possible. Secondly it is always invertible, so that the controller can stabilize the robot at a static point with the robot velocity being zero.

It can be seen that the choice of $\alpha(Y, \dot{Y})$ involves the estimation values $\hat{\theta}$, $\dot{\hat{x}}$ and $\dot{\hat{y}}$, thus the efficient estimation of these values is important, the numerical differentiation method used for estimation and its advantages can be found in section 2.4.3.

4.3.3 Algebraic estimation of F

Now there is only one parameter F left in the i-PID controller to be determined. The calculation of F also uses the algebraic technique described in section 2.4.3.

Let us consider the local approximated model:

$$\ddot{Y} = F + \alpha u \tag{4.11}$$

where F can be considered as a constant between two sampling time if the sampling time is fast enough with respect to the time variation of F. Then by taking the Laplace transformation of both sides of equation (4.11), one obtains

$$s^2\mathcal{Y}(s) - s\mathcal{Y}(0) - \mathcal{Y}'(0) = \frac{F}{s} + \alpha\mathcal{U}(s) \tag{4.12}$$

where $\mathcal{Y}$ and $\mathcal{U}$ are the Laplace transforms of $y(t)$ and $u(t)$ respectively.

In order to eliminate the unknown terms $\mathcal{Y}(0)$ and $\mathcal{Y}'(0)$ which are linked to unknown initial conditions, we take the 2^{nd} order derivative of both sides with respect to s:

$$s^2\mathcal{Y}''(s) + 4s\mathcal{Y}'(s) + 2\mathcal{Y}(s) = \frac{2F}{s^3} + \alpha\mathcal{U}''(s) \tag{4.13}$$

By dividing both sides of equation (4.13) with s^3, one has:

$$\frac{\mathcal{Y}''(s)}{s} + \frac{4\mathcal{Y}(s)'}{s^2} + \frac{2\mathcal{Y}(s)}{s^3} = \frac{2F}{s^6} + \frac{\alpha\mathcal{U}''(s)}{s^3} \tag{4.14}$$

Take the inverse Laplace transformation of both sides of equation (4.14), one obtains:

$$\begin{aligned}\int_0^T (-\tau)^2 Y d\tau + \int_0^T 4(T-\tau)(-\tau)Y d\tau + \int_0^T (T-\tau)^2 Y d\tau \\ = \frac{2FT^5}{5!} + \alpha\int_0^T \frac{(T-\tau)^2}{2!}(-\tau)^2 u d\tau\end{aligned} \tag{4.15}$$

where $[0, T]$ is a short time window, and the window is sliding in order to get the estimate at each time instant. Let $\tau = \delta T \in [0, T]$, where $\delta \in [0, 1]$, after simplification equation (4.15) becomes:

$$T^3\int_0^1 (6\delta^2 - 6\delta + 1)Y d\delta = \frac{FT^5}{60} + \frac{T^5\alpha}{2}\int_0^1 (1-\delta)^2\delta^2 u d\delta \tag{4.16}$$

Hence, at sampling step k, the numerical estimate value of F_k can be expressed as:

$$F_k = \frac{60}{T^2}\int_0^1 (6\delta^2 - 6\delta + 1)Y d\delta - 30\alpha \int_0^1 (1-\delta)^2\delta^2 u d\delta \tag{4.17}$$

4.4 Simulation results

In the simulation, the parameters are set: $K_2 = 20$, $K_1 = 100$, time window $T = 3s$, sampling time $T_s = 0.01$. The reference trajectory is set as:

$$\begin{cases} x_r &= \sin 2t \\ y_r &= \sin \frac{t}{2} \end{cases}$$

Fig. 4.1 to Fig. 4.7 show the simulation result of the designed control applied on the non-holonomic wheeled mobile robot without noise. Fig. 4.1, Fig. 4.2 and Fig. 4.3 show the trajectory tracking result, with the blue curves representing the reference trajectory, the red ones representing the robot trajectory. $i(\hat{\theta})$ during the simulation is shown in Fig. 4.4, the control inputs are shown in Fig 4.5 and Fig. 4.6, and the tracking errors are shown in Fig. 4.7. One can see from the simulation that the robot is well controlled with the stabilization of the robot system and α selected appropriately.

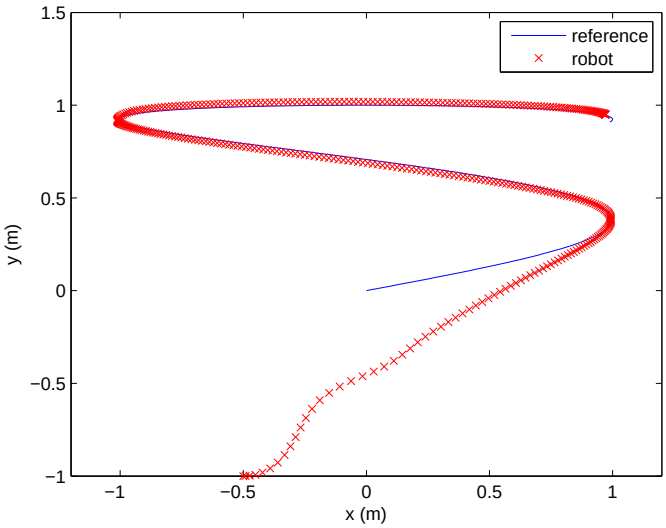


Figure 4.1: Trajectory tracking result without noise

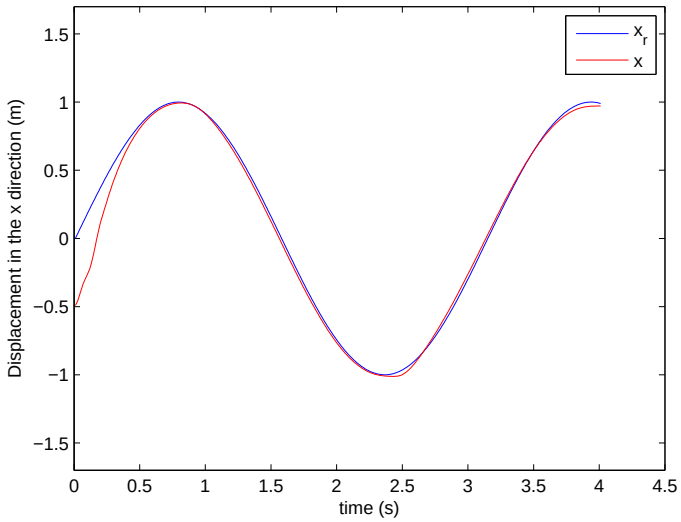


Figure 4.2: Tracking of position x without noise

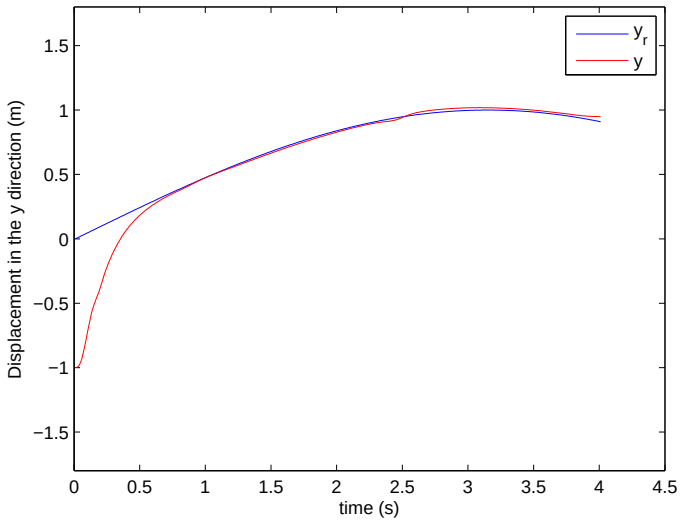


Figure 4.3: Tracking of position y without noise

When adding disturbance of white Gaussian noise of $SNR = 30dB$ (signal-to-noise ratio) to the measurement (as shown in Fig. 4.8 and Fig. 4.9), the simulation result is shown from Fig. 4.10 to Fig. 4.16. Fig. 4.10, Fig. 4.11 and Fig. 4.12 show the tracking result, $i(\hat{\theta})$ is shown in Fig. 4.13, the control inputs

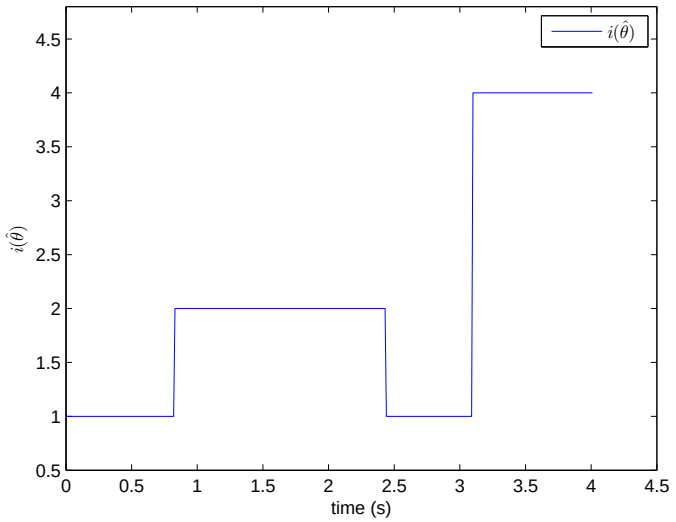


Figure 4.4: $i(\hat{\theta})$ in the noise free case

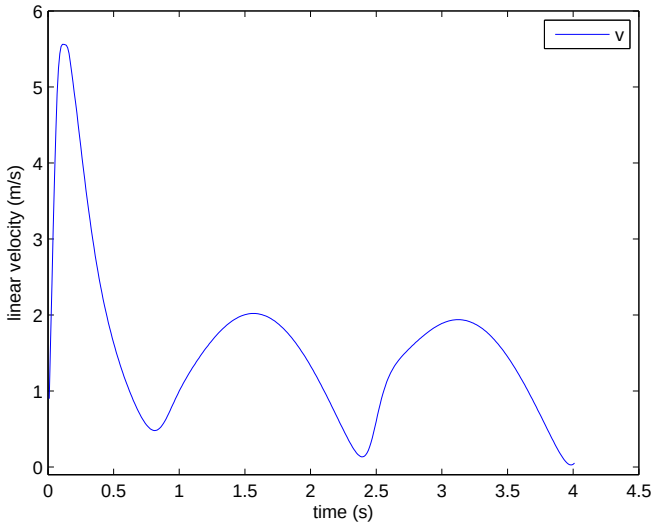


Figure 4.5: Linear velocity control without noise

are shown in Fig. 4.14 and Fig. 4.15, and tracking errors are shown in Fig. 4.16. As we can see that the robot is able to track the trajectory with measurement noises, and the designed controller is effective and robust to the noises.

One can notice that there are frequent changes of $\alpha_{i(\hat{\theta})}$ in the Fig. 4.13, this

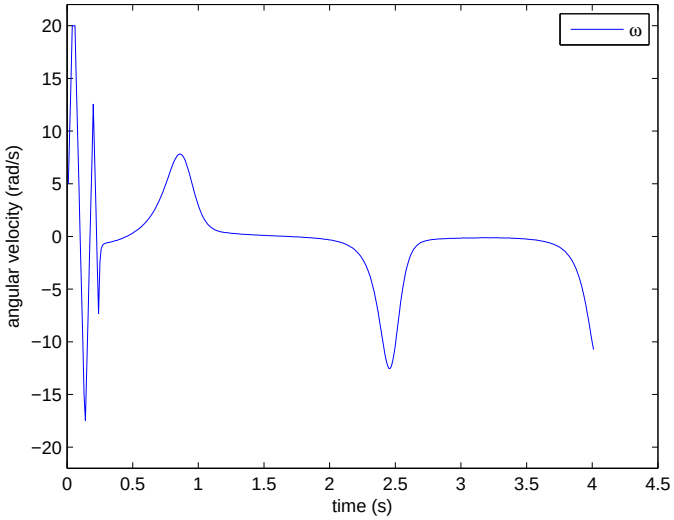


Figure 4.6: Angular velocity control without noise

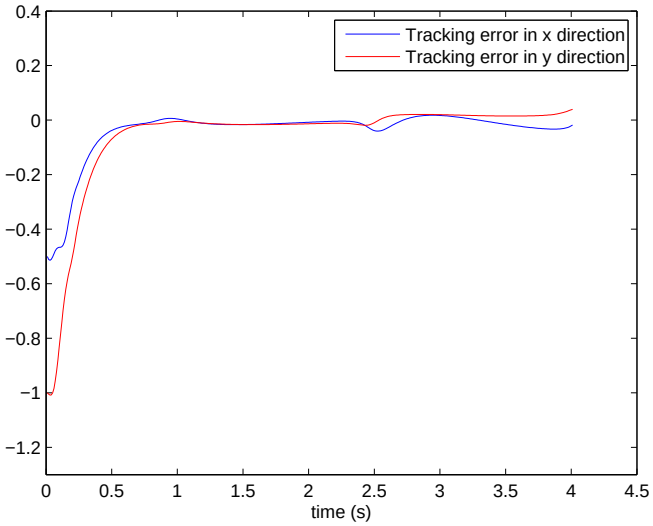


Figure 4.7: Tracking errors without noise

is because when the angle of the robot θ is very close to $\frac{k\pi}{2}(k = 0, 1, 2...)$, the estimated $\hat{\theta}$ under noises may vacillate around $\frac{k\pi}{2}$. While this error is not significant and can be eliminated by introducing a hysteresis zone. When $\hat{\theta}$ vacillates around $\frac{k\pi}{2}$ within the hysteresis zone, we consider that $\alpha_{i(\hat{\theta})}$ does not change.

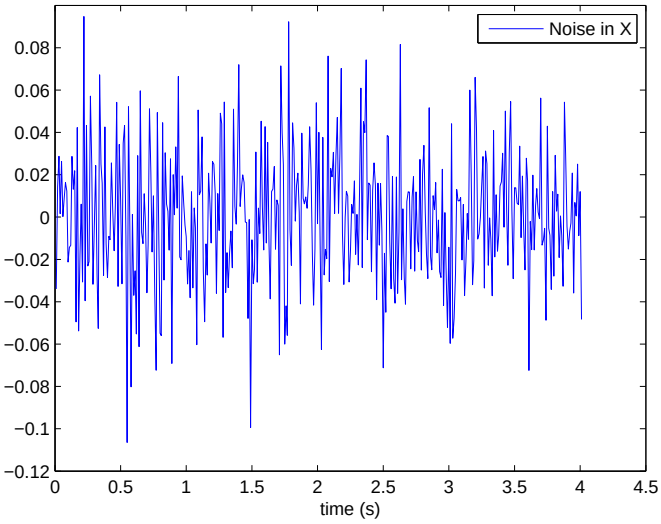


Figure 4.8: Noise imposed in x $SNR = 30dB$

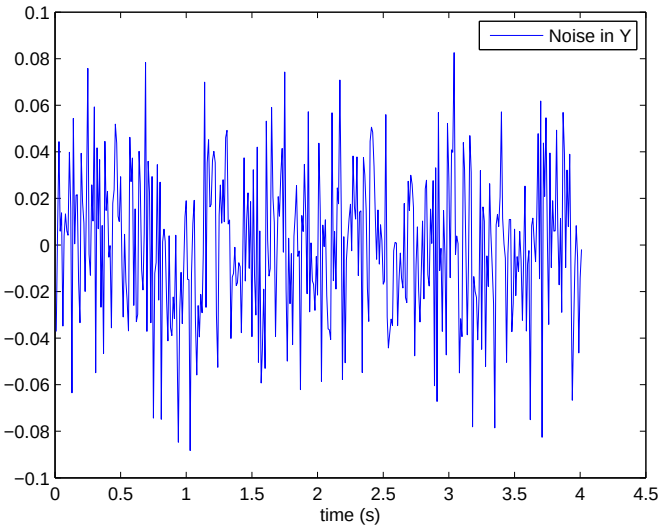


Figure 4.9: Noise imposed in y $SNR = 30dB$

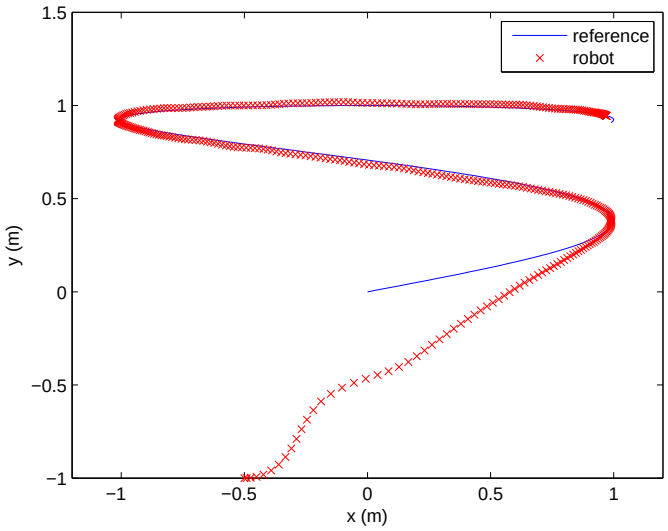


Figure 4.10: Trajectory tracking result with white Gaussian noise $SNR = 30dB$

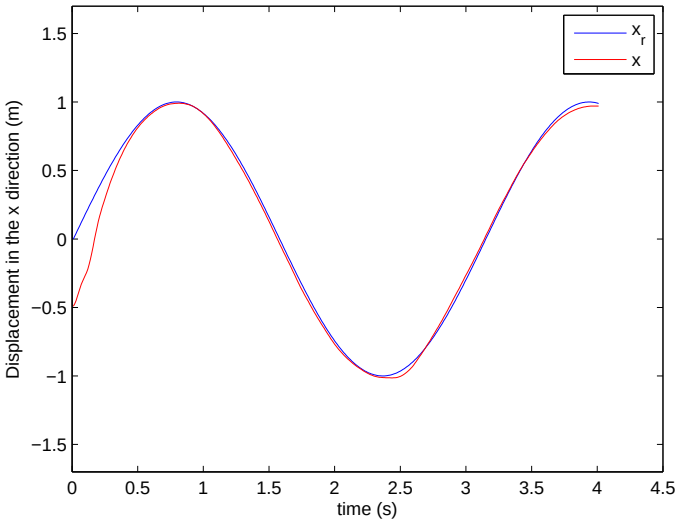


Figure 4.11: Tracking of position x with white Gaussian noise $SNR = 30dB$

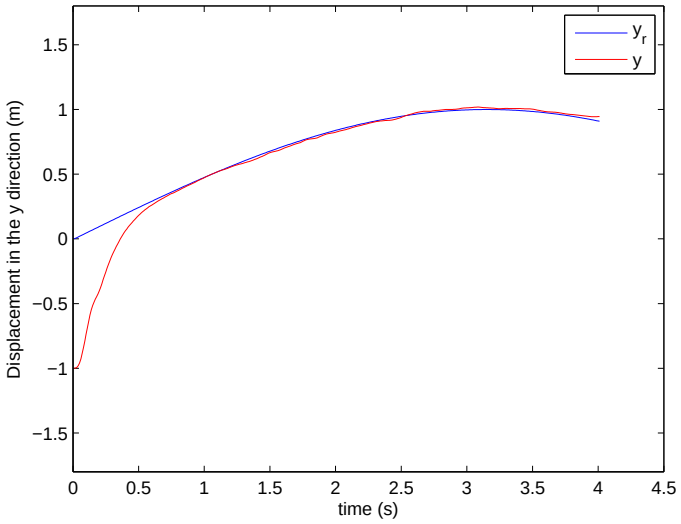


Figure 4.12: Tracking of position y with white Gaussian noise $SNR = 30dB$

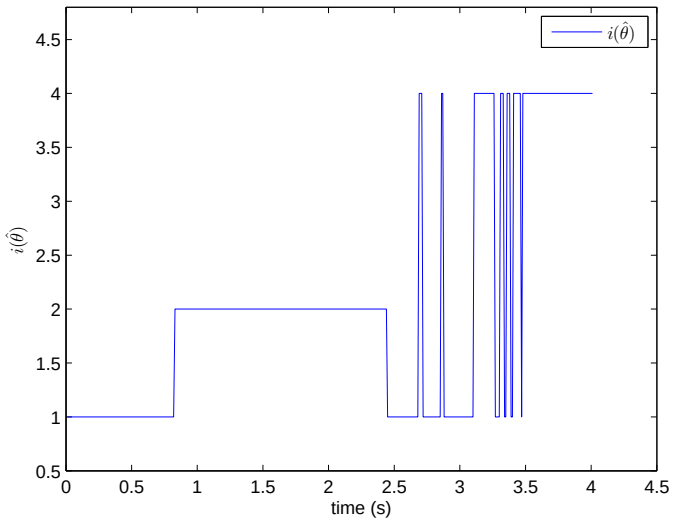


Figure 4.13: $i(\hat{\theta})$ with white Gaussian noise $SNR = 30dB$

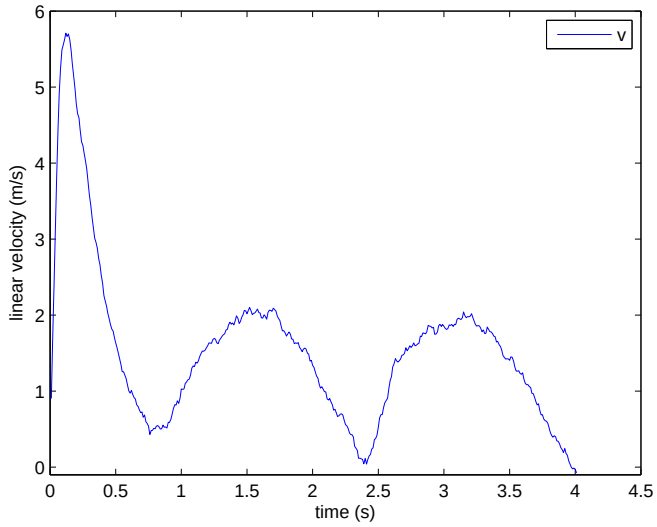


Figure 4.14: Linear velocity control with white Gaussian noise $SNR = 30dB$

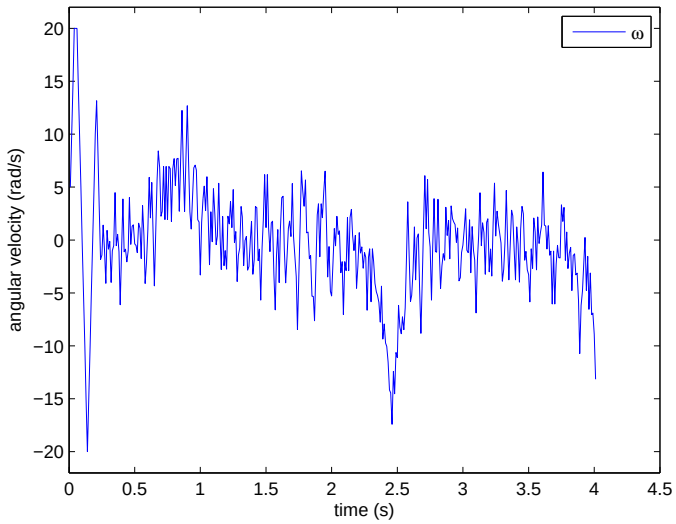


Figure 4.15: Angular velocity control with white Gaussian noise $SNR = 30dB$

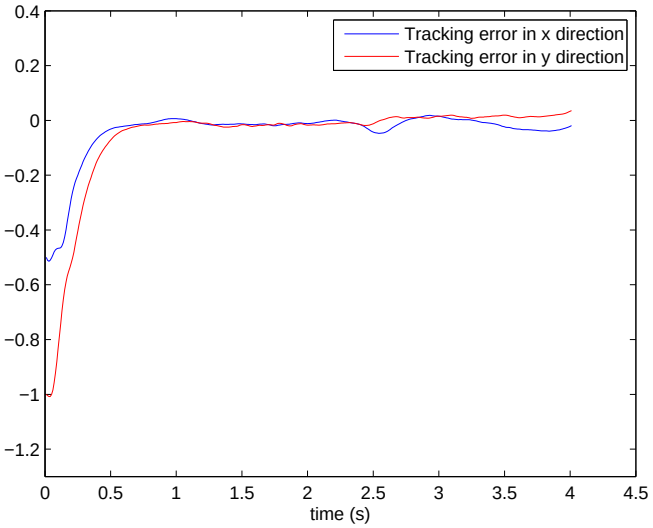


Figure 4.16: Tracking errors with white Gaussian noise $SNR = 30dB$

The following simulation (Fig. 4.17 to Fig. 4.23) is made with a hysteresis zone of $\frac{\pi}{40}$ and a white Gaussian noise of $SNR = 30dB$ adding to the measurement. One can see in the Fig. 4.20 that frequent changes of $\alpha_{i(\hat{\theta})}$ are eliminated by adding the hysteresis zone.

Fig. 4.24 to Fig. 4.28 illustrate the simulation of stabilize the robot at point $(4, 1)$, a white Gaussian noise of $SNR = 30dB$ is added to the measurement as well. Tracking result is shown in Fig. 4.24 and Fig. 4.25, and control inputs are shown in Fig. 4.27 and Fig. 4.28. As we can see, the controller is able to stabilize the robot at a static point with the robot velocity being 0.

Two more real-time 3D simulations are made in the following link by using ROS (Robot Operating System): Video Link. One is reference tracking simulation and the other one is the stabilization of the robot at a static point with the robot velocity being zero.

4.5 conclusion

This chapter presents the i-PID controller applied to the non-holonomic wheeled mobile robot. After the study of the system, the parameter α in the controller is selected as a switching function according to the information of the system.

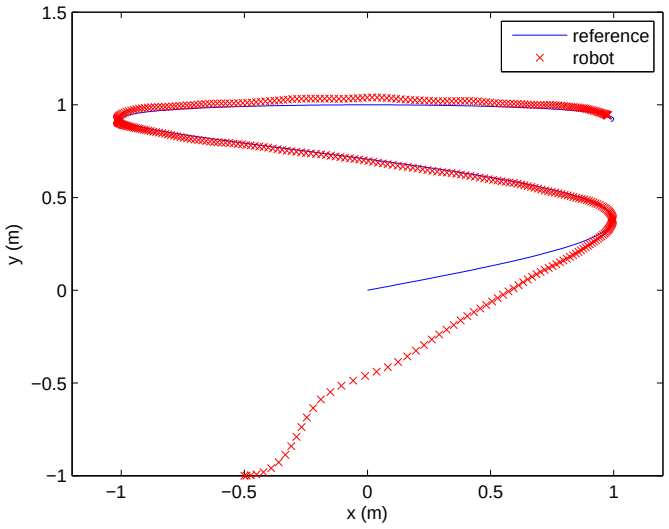


Figure 4.17: Trajectory tracking result with a hysteresis zone

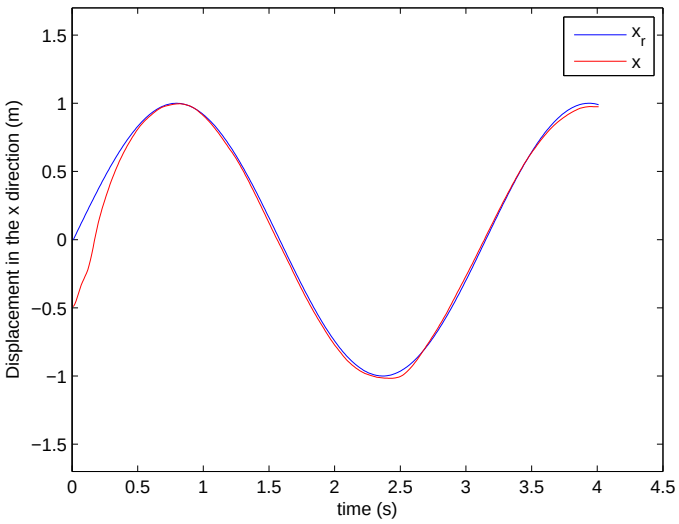


Figure 4.18: Tracking of position x with a hysteresis zone

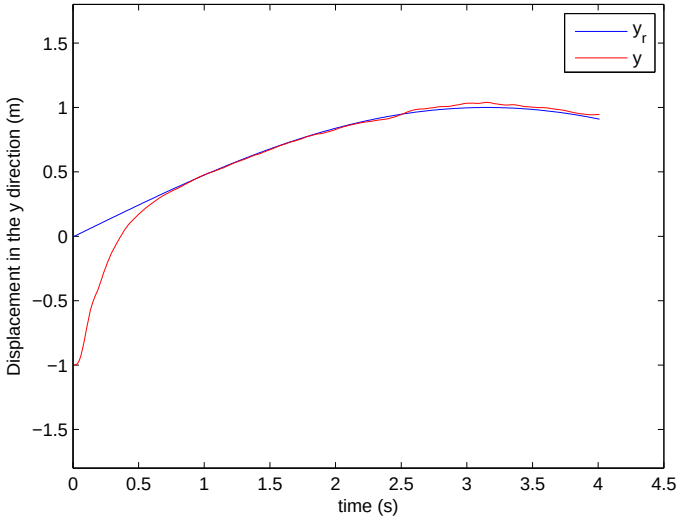


Figure 4.19: Tracking of position y with with a hysteresis zone

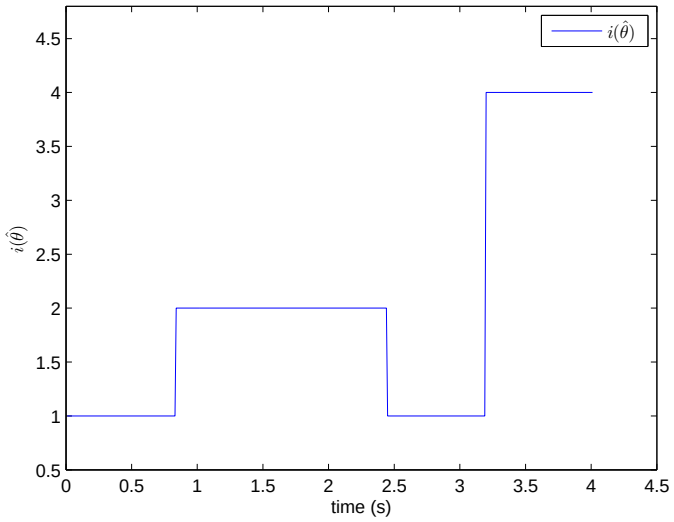


Figure 4.20: $i(\hat{\theta})$ with a hysteresis zone

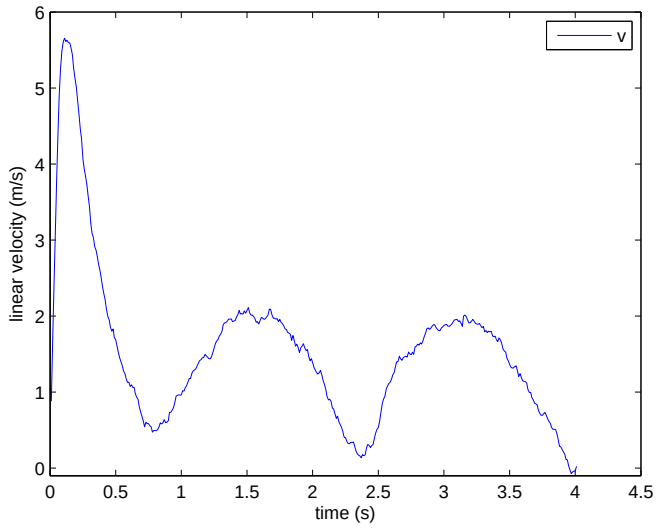


Figure 4.21: Linear velocity control with a hysteresis zone

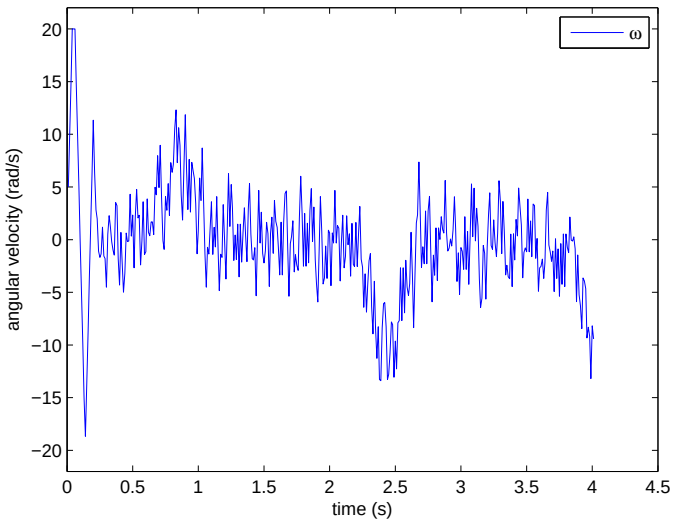


Figure 4.22: Angular velocity control with a hysteresis zone

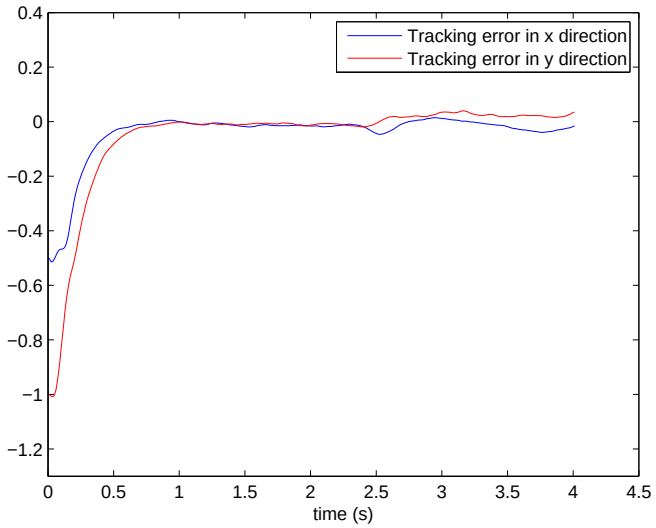


Figure 4.23: Tracking errors with a hysteresis zone

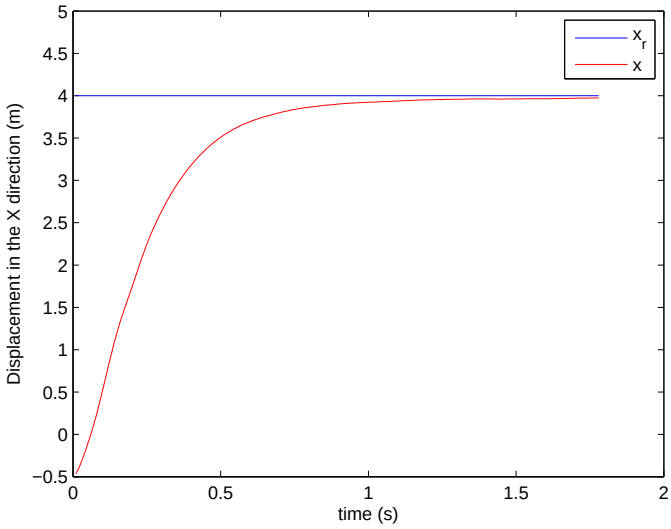


Figure 4.24: Stabilization of position x

The presented i-PID controller is robust to the measurement disturbance of the robot, and it can even stabilize the robot at a static point with the robot velocity being zero with the proposed parameter α. The effectiveness and robustness of the designed controller were shown thereafter via several different simulations.

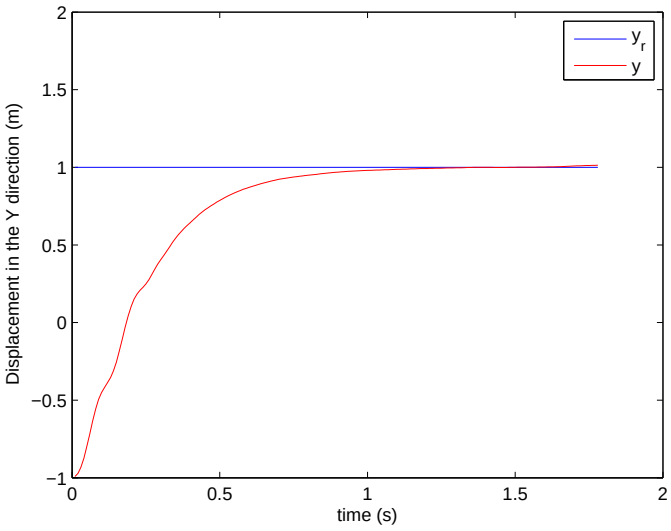


Figure 4.25: Stabilization of position y

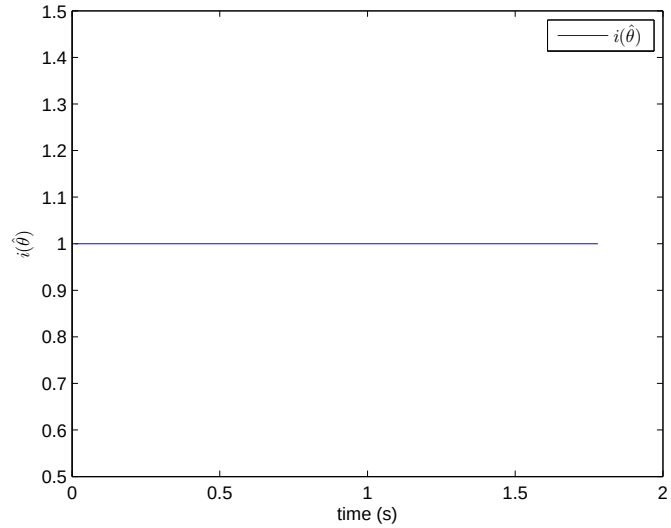


Figure 4.26: $i(\hat{\theta})$ of stabilization

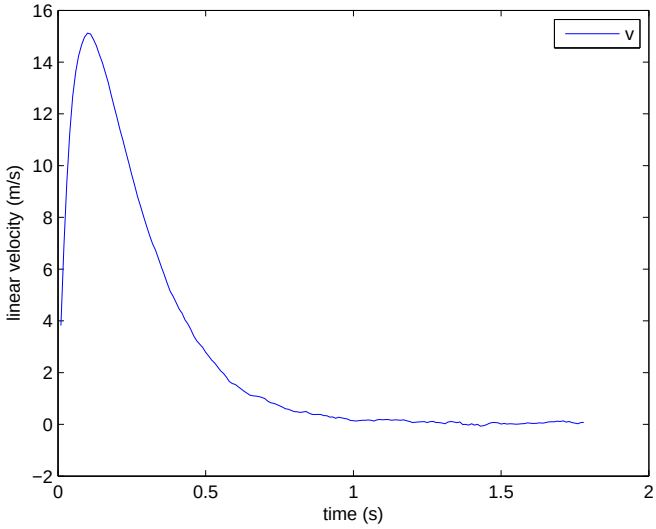


Figure 4.27: Linear velocity control of stabilization

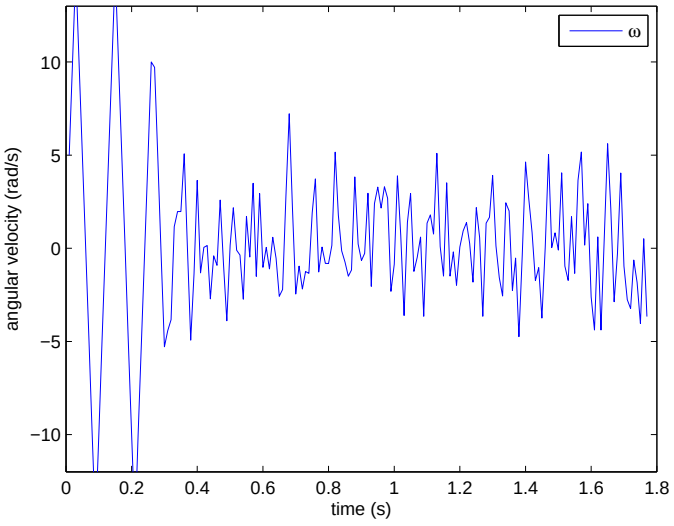


Figure 4.28: Angular velocity control of stabilization

Chapter 5

Motion planning for mobile robots using potential field and the i-PID controller

5.1 Introduction

Artificial potential field approach is widely studied because of its simplicity and interesting mathematical analysis Latombe (1991); Rezaee & Abdollahi (2012); Sfeir *et al.* (2011). The basic idea of this approach is to fill the robot workplace with potential fields, the attractive potential field is caused by the target to attract the robot moving towards the target, and the repulsive potential field is caused by obstacles to repulse the robot away from obstacles. There are also some extensions for the potential field method. In Conn & Kam (1998) moving obstacles are considered with the time are considered as one of the dimensions of the model workplace. In Ge & Cui (2002) the velocity of obstacles and target are taken into consideration, and the unreachable target with obstacle nearby problem is discussed in Ge & Cui (2000). Although the basic idea of potential field approach in path planning seems easy, the problems of local minima and oscillatory movements sometimes make the method less efficiency.

Local minima is the case when the total force imposed on the robot is zero though robot has not reached its goal, or the direction of the total force passes through obstacles. In order to solve local minima problems, Charifa & Bikdash (2011) combines the potential field with boundary following algorithm, which in-

troduces long detours for the robot. Laue & Röfer (2005) proposed to combine potential field approach with A^* search, which however decreases the continuity and increases the computational complexity because of the necessity of dividing the workplace into grids. There are also other solutions, such as tangential potential fields Kim & Kim (2003), object grouping method Zhang *et al.* (2006), while these methods are not effective when the total force passes through the center of the obstacle. Moreover, the total force imposed on the robot is calculated by summing up all the forces, this may cause some oscillations in the robot motion. This problem can be solved by assigning distance criteria in the potential field function.

In this chapter, we propose a new potential field approach for robot motion planning to solve local minima and oscillation problems. The new potential function is defined by taking into account the robot orientation and angular velocity besides the position and the linear velocity, and distance criteria is added to the original potential function to decrease oscillations. The new potential function allows the robot to avoid local minima and decreases the oscillations in robot motion. At last, the i-PID controller, which is robust to the disturbances in the system, is used for robot motion planning. This method only needs the online measurements of the robot velocity and information of obstacles.

5.2 Problem statement

In Chapter 3 we have proposed a path planning algorithm using optimal control. Leaving the complexity of programming, the efficiency of the method largely depends on the efficiency of the optimization algorithm, and sometimes there is even no solution to the optimization problem. Therefore in this chapter we propose another robot motion planning method using potential field and i-PID controller. When there is no solution to the optimization problem, the robot can switch to this method which guarantees the obstacle avoidance, as shown in Fig. 1.5(b).

The potential field approach is widely used in robot motion planning, however as we all know that the original potential field approach suffers from local minima problems. As shown in Fig. 5.1, the direction of the total force imposed on the

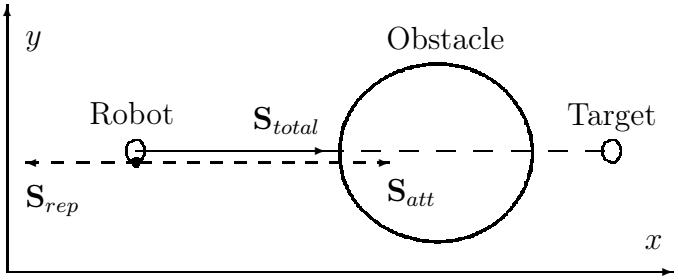


Figure 5.1: Local minima problems in original potential field method

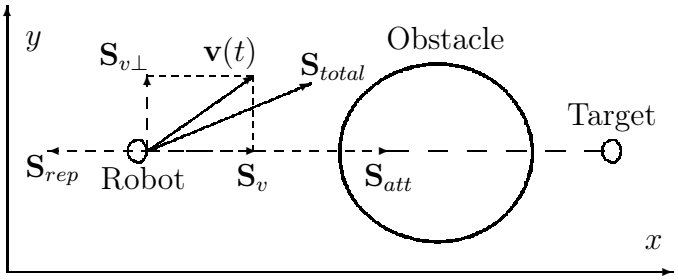


Figure 5.2: Modified potential field method

robot pass through the center of the obstacle, thus the generated force $\mathbf{S}_{total}$* can not navigate the robot away from the obstacle. A modified method was proposed in Ge & Cui (2002), which takes the robot velocity into consideration, thus another force is introduced in the direction of the velocity to navigate the robot away from the obstacle, as seen in Fig. 5.2. However when the direction of the velocity also passes through the center of the obstacle, as seen in Fig. 5.3, the modified method can not navigate the robot to avoid the obstacle either. This motivates us to think whether we can introduce a force in the direction perpendicular to the direction of the robot when the robot is approaching the obstacle, thus the local minima problem can be solved. Therefore we take the orientation and the angular velocity of the robot into consideration to propose a new potential field function (see the next section), which is able to solve local minima problems, which is the first contribution of this chapter.

Moreover, in the robot motion planning problem, the potential forces need to change as smoothly as possible in order to achieve good tracking performance for the controller, since it is the derivative of the potential field will be considered

*The boldface type is used for vectors in this chapter.

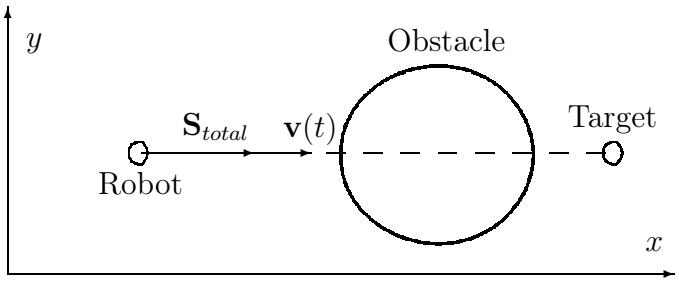


Figure 5.3: Local minima problems in modified potential method

to design the controller. Generally speaking, the total force (i.e. the derivative of the total potential filed) is used for the controller design. A common way is to set a positive linear velocity for the robot and design an angular controller to make the orientation of the robot converging to the angle of the total force. While this method is not robust to the disturbances in the system, therefore the second contribution of this chapter is to propose the i-PID controller, which is robust to the disturbance, in order to improve the motion planning performance.

5.3 Potential field function

5.3.1 Attractive potential function

Firstly, let us define the attractive potential field function. Like other potential field methods, it is defined as a function of relative distance between the robot and the target:

$$U_{att}(\mathbf{P}) = K_{att}\|\mathbf{P}_{tar} - \mathbf{P}(t)\|^2 \tag{5.1}$$

where $\mathbf{P}(t)$ and $\mathbf{P}_{tar}$ denote the positions of the robot and the target at time t respectively, $\|\mathbf{P}_{tar} - \mathbf{P}(t)\|$ is the Euclidean distance between the robot and the target, and K_{att} is a positive parameter.

Therefore the desired virtual force towards the target is defined as the negative gradient of the attractive potential function with respect to the robot position:

$$\begin{aligned}\mathbf{S}_{att}(\mathbf{P}) &= -\nabla_{\mathbf{P}} U_{att}(\mathbf{P}) = \frac{\partial U_{att}(\mathbf{P})}{\partial \mathbf{P}} \\ &= 2K_{att}\|\mathbf{P}_{tar} - \mathbf{P}(t)\|\mathbf{n}_{RT}\end{aligned} \tag{5.2}$$

where $\mathbf{n}_{RT}$ is the unit vector pointing from the robot to the target.

5.3.2 Repulsive potential function

Conventionally, the repulsive function is defined as a function of relative distance between the robot and obstacles. Some papers also take the robot velocity into consideration, like Ge & Cui (2002). In this chapter a new repulsive function is presented which makes fully use of the position, velocity, orientation and angular velocity of the robot, and it is able to avoid local minima problems.

In Ge & Cui (2002) the potential function of robot position and velocity is considered as follows. Define $P_d(\mathbf{P}, \mathbf{P}_{obs})$ as the distance between the robot and obstacle, and $V_{RO}(t) = \mathbf{v}(t)^T \mathbf{n}_{RO}$ as the velocity in the direction from the robot to the obstacle, where $\mathbf{v}(t)$ is the robot velocity and $\mathbf{n}_{RO}$ is a unit vector point from the robot to the obstacle. Define a_{max} as the maximum deceleration of the robot, thus the distance traveled by the robot before V_{RO} reduces to 0 is $P_m(V_{RO}) = \frac{V_{RO}^2}{2a_{max}}$. Then the repulsive potential function is defined as

$$U_{rep}(\mathbf{P}, \mathbf{v}) = \begin{cases} 0, \quad if\ P_d(\mathbf{P}, \mathbf{P}_{obs}) - P_m \geq p_0\ or\ V_{RO} \leq 0 \\ K_{pv}\left(\frac{1}{P_d(\mathbf{P}, \mathbf{P}_{obs}) - P_m} - \frac{1}{p_0}\right), \\ \quad if\ 0 < P_d(\mathbf{P}, \mathbf{P}_{obs}) - P_m < p_0\ and\ V_{RO} > 0 \end{cases} \tag{5.3}$$

where p_0 is the influence range of the obstacle, and K_{pv} is a positive constant.

However one can notice in (5.3) that if $V_{RO}(t)$ changes from positive to negative in one step, a sudden change will occur in the value of repulsive potential function $U_{rep}(\mathbf{P}, \mathbf{v})$. For example in Fig. 5.4, the relative velocity $V_{RO}(t_k) > 0$ at step t_k, however at step t_{k+1} the relative velocity $V_{RO}(t_{k+1}) < 0$, thus $U_{rep}(\mathbf{P}, \mathbf{v})$ will drop to 0 suddenly. When considering the repulsive force, i.e. the derivative of this repulsive potential, it will result in frequent oscillations in complex environment, which is not expected for controller design. To overcome this problem, we multiply the original repulsive potential function by $\cos\theta_d$, where $\theta_d = \|\theta - \theta_{RO}\|$, and θ is the orientation of the robot, θ_{RO} is the angle of the unit vector $\mathbf{n}_{RO}$. The new $U_{rep}(\mathbf{P}, \mathbf{v})$ is as follows:

$$U_{rep}(\mathbf{P}, \mathbf{v}) = \begin{cases} 0, \quad if\ P_d(\mathbf{P}, \mathbf{P}_{obs}) - P_m \geq p_0\ or\ V_{RO} \leq 0 \\ K_{pv}\left(\frac{1}{P_d(\mathbf{P}, \mathbf{P}_{obs}) - P_m} - \frac{1}{p_0}\right)\cos\theta_d, \\ \quad if\ 0 < P_d(\mathbf{P}, \mathbf{P}_{obs}) - P_m < p_0\ and\ V_{RO} > 0 \end{cases} \tag{5.4}$$

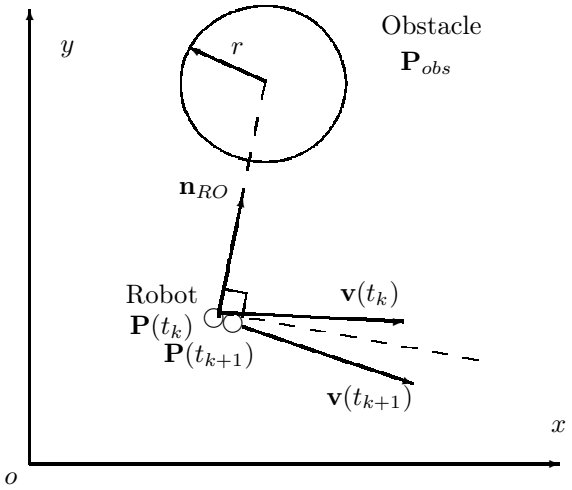


Figure 5.4: Sudden change of V_{RO}

which guarantees the repulsive potential function reduce to zero smoothly while $V_{RO}(t)$ become negative. Moreover as stated in the problem statement, the above defined potential function suffers from local minima problems. Therefore in this chapter a new repulsive potential function is presented to solve local minima problems.

Let us define that if a maximum deceleration β_{max} is applied to the robot to reduce its angular velocity, the angle traveled by the robot before angular velocity $w(t)$ reduces to zero is:

$$\theta(\omega) = \frac{\omega(t)^2}{2\beta_{max}}$$

Then the repulsive potential function of robot orientation and angular velocity is defined as follows:

$$U_{rep}(\theta,\omega) = \begin{cases} 0, \quad if\ V_{RO} < 0\ or\ P_d > p_\theta\ or\ \theta_d > \theta_0 \\ \\ (-K_{\theta 1}(\theta_d - \theta(\omega))^2 + K_{\theta 2})^2 (p_\theta - P_d(\mathbf{P}, \mathbf{P}_{obs}))^2 (\theta_0 - \theta_d)^2, \\ \quad if\ V_{RO} > 0\ and\ P_d \le p_\theta\ and\ \theta(\omega) < \theta_d \le \theta_0 \\ \\ K_{\theta 2}^2 \big(p_\theta - P_d(\mathbf{P}, \mathbf{P}_{obs})\big)^2 \big(\theta_0 - \theta_d\big)^2, \\ \quad if\ V_{RO} > 0\ and\ P_d < p_\theta\ and\ \theta_d < \theta(\omega) \end{cases} \tag{5.5}$$

where p_θ is a positive constant describing the influence range of obstacle on $U_{rep}(\theta,\omega)$, and $p_\theta > p_0$, θ_0 is the influence angle, $K_{\theta 1}$ and $K_{\theta 2}$ are positive con-

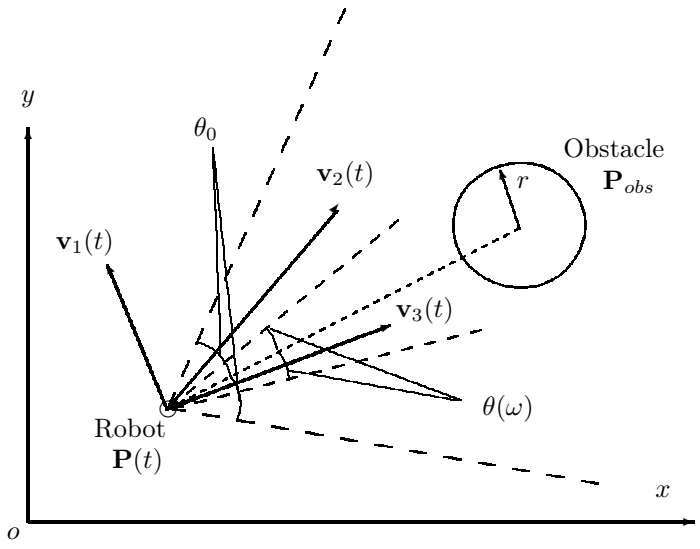


Figure 5.5: Different cases of $U_{rep}(\theta, \omega)$

stants. As shown in Fig. 5.5, the direction of $\mathbf{v}_1(t)$ is out of the influence angle θ_0, thus $U_{rep}(\theta, \omega) = 0$. In the second case, the direction of $\mathbf{v}_2(t)$ is in the influence angle θ_0 but out of the range of $\theta(\omega)$, the potential function is of paraboloidal profile. In the last case, the direction of $\mathbf{v}_3(t)$ is inside angle $\theta(\omega)$, the potential function is defined as the maximum value of the parabola.

One can notice that in (5.5), $\big(p_\theta - P_d(\mathbf{P}, \mathbf{P}_{obs})\big)^2$ enables the value of $U_{rep}(\theta, \omega)$ change smoothly when robot reaches and leaves the influence range of the obstacle, and $\big(\theta_0 - \theta_d\big)^2$ enables the value of $U_{rep}(\theta, \omega)$ change smoothly when θ_d reaches and exceeds the influence angle. The smooth change of $U_{att}(\theta, \omega)$ is very beneficial for tracking. Therefore the new repulsive potential function is defined as

$$U_{rep} = U_{rep}(\mathbf{P}, \mathbf{v}) + U_{rep}(\theta, \omega) \tag{5.6}$$

Thus the corresponding repulsive force $\mathbf{S}_{rep}$ is defined as the negative gradient of the repulsive potential function with respect to robot position and velocity. Since $\nabla_{\mathbf{v}} U(\theta, \omega) = 0$, we have

$$\begin{aligned} \mathbf{S}_{rep} &= \mathbf{S}_{rep}(\mathbf{P}, \mathbf{v}) + \mathbf{S}_{rep}(\theta, \omega) \\ &= -\nabla_{\mathbf{P}} U_{rep}(\mathbf{P}, \mathbf{v}) - \nabla_{\mathbf{v}} U_{rep}(\mathbf{P}, \mathbf{v}) - \nabla_{\mathbf{P}} U_{rep}(\theta, \omega) \end{aligned} \tag{5.7}$$

The robot velocity in the direction from the robot to the obstacle V_{RO} can be

written as

$$\begin{aligned} V_{RO}(t) &= \mathbf{v}(t)^T \mathbf{n}_{RO} \\ &= \mathbf{v}(t)^T \frac{\mathbf{P}_{obs} - \mathbf{P}(t)}{\|\mathbf{P}_{obs} - \mathbf{P}(t)\|} \end{aligned}$$

then the gradients of $V_{RO}(t)$ with respect to velocity and position are calculated as

$$\nabla_{\mathbf{v}} V_{RO}(t) = \mathbf{n}_{RO} \tag{5.8}$$

$$\nabla_{\mathbf{P}} V_{RO}(t) = \frac{V_{RO}(t)\mathbf{n}_{RO} - \mathbf{v}(t)}{\|\mathbf{P}_{obs} - \mathbf{P}(t)\|} \tag{5.9}$$

For clarity, as shown in Fig. 5.6, denote $V_{RO\perp}(t)\mathbf{n}_{RO\perp}$ as the velocity component perpendicular to $V_{RO}(t)\mathbf{n}_{RO}$, which is given in the following equation:

$$V_{RO\perp}(t)\mathbf{n}_{RO\perp} = \mathbf{v}(t) - V_{RO}(t)\mathbf{n}_{RO} \tag{5.10}$$

where

$$\begin{aligned} &V_{RO\perp}(t) = \sqrt{\mathbf{v}(t)^2 - V_{RO}(t)^2} \\ &\mathbf{n}_{RO\perp}^T \mathbf{n}_{RO} = 0 \end{aligned}$$

thus equation (5.9) can be written as

$$\nabla_{\mathbf{P}} V_{RO}(t) = \frac{-V_{RO\perp}\mathbf{n}_{RO\perp}}{\|\mathbf{P}_{obs} - \mathbf{P}(t)\|}$$

Denote $\mathbf{n}_{\theta_{RO\perp}}$ as the unit vector perpendicular to $\mathbf{n}_{RO}$, and $\theta_{RO} - \theta_{RO\perp} = \pi/2$, where $\theta_{RO\perp}$ is the argument of unit vector $\mathbf{n}_{\theta_{RO\perp}}$. Thus we have

$$\nabla_{\mathbf{P}} \theta_{RO} = \frac{\mathbf{n}_{\theta_{RO\perp}}}{P_d(\mathbf{P}, \mathbf{P}_{obs})} \tag{5.11}$$

Therefore

$$\nabla_{\mathbf{P}} \theta_d = \nabla_{\mathbf{P}} \|\theta - \theta_{RO}\| \tag{5.12}$$

If $\theta > \theta_{RO}$, we obtain

$$\nabla_{\mathbf{P}} \theta_d = \nabla_{\mathbf{P}} (\theta - \theta_{RO}) = \frac{-\mathbf{n}_{\theta_{RO\perp}}}{P_d(\mathbf{P}, \mathbf{P}_{obs})} \tag{5.13}$$

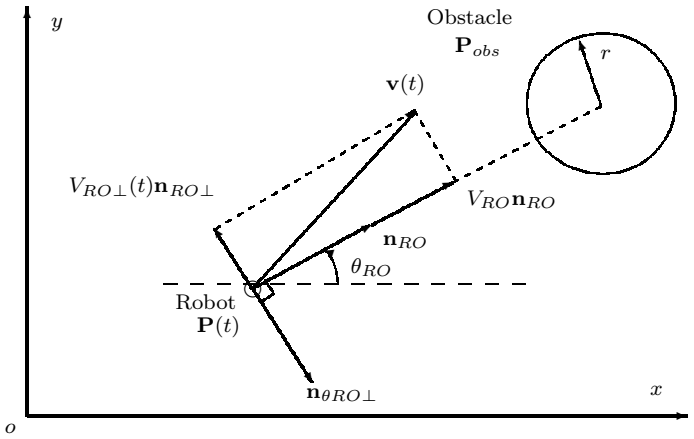


Figure 5.6: Vectors for defining repulsive force

If $\theta \leq \theta_{RO}$, we have

$$\nabla_{\mathbf{P}}\theta_d = \nabla_{\mathbf{P}}(\theta_{RO} - \theta) = \frac{\mathbf{n}_{\theta_{RO\perp}}}{P_d(\mathbf{P}, \mathbf{P}_{obs})} \tag{5.14}$$

However one can notice that when $\theta > \theta_{RO}$, $\mathbf{n}_{\theta_{RO\perp}} = -\mathbf{n}_{RO\perp}$, and when $\theta \leq \theta_{RO}$, $\mathbf{n}_{\theta_{RO\perp}} = \mathbf{n}_{RO\perp}$, thus one can obtain that

$$\nabla_{\mathbf{P}}\theta_d = \frac{\mathbf{n}_{RO\perp}}{P_d(\mathbf{P}, \mathbf{P}_{obs})}$$

Therefore we have

$$\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v}) = \begin{cases} 0, \ if \ P_d(\mathbf{P}, \mathbf{P}_{obs}) - P_m \geq p_0 \ or \ V_{RO} \leq 0 \\ \\ \mathbf{S}_{rep1} + \mathbf{S}_{rep2}, \\ \quad if \ 0 < P_d(\mathbf{P}, \mathbf{P}_{obs}) - P_m < p_0 \ and \ V_{RO} > 0 \end{cases} \tag{5.15}$$

where

$$\mathbf{S}_{rep1} = \frac{-K_{pv}\cos\theta_d}{(P_d(\mathbf{P}, \mathbf{P}_{obs}) - P_m)^2}\left(1 + \frac{V_{RO}}{a_{max}}\right)\mathbf{n}_{RO} \tag{5.16}$$

and

$$\begin{aligned} \mathbf{S}_{rep2} = & \frac{K_{pv}V_{RO}V_{RO\perp}\cos\theta_d}{a_{max}P_d(\mathbf{P}, \mathbf{P}_{obs})(P_d(\mathbf{P}, \mathbf{P}_{obs}) - P_m)^2}\mathbf{n}_{RO\perp} \\ & + K_{pv}\sin\theta_d\left(\frac{1}{P_d(\mathbf{P}, \mathbf{P}_{obs}) - P_m} - \frac{1}{p_0}\right)\frac{\mathbf{n}_{RO\perp}}{P_d(\mathbf{P}, \mathbf{P}_{obs})} \end{aligned} \tag{5.17}$$

We also have

$$\mathbf{S}_{rep}(\theta,\omega) = \begin{cases} 0, \quad if\ V_{RO} < 0\ or\ P_d(\mathbf{P},\mathbf{P}_{obs}) > p_\theta\ or\ \theta_d > \theta_0 \\ \\ \mathbf{S}_{rep3} + \mathbf{S}_{rep4} \\ \quad if\ V_{RO} > 0\ and\ P_d(\mathbf{P},\mathbf{P}_{obs}) \le p_\theta\ and\ \theta(\omega) < \theta_d \le \theta_0 \\ \\ \mathbf{S}_{rep5} + \mathbf{S}_{rep6}, \\ \quad if\ V_{RO} > 0\ and\ P_d(\mathbf{P},\mathbf{P}_{obs}) < p_\theta\ and\ \theta_d < \theta(\omega) \end{cases} \tag{5.18}$$

where

$$\mathbf{S}_{rep3} = -2M^2H(\theta_0-\theta_d)^2\mathbf{n}_{RO} \tag{5.19}$$

$$\begin{aligned} \mathbf{S}_{rep4} = {} & 4K_{\theta1}MH^2(\theta_d-\theta(\omega))(\theta_0-\theta_d)^2\frac{\mathbf{n}_{RO\perp}}{P_d(\mathbf{P},\mathbf{P}_{obs})} \\ & + 2M^2H^2(\theta_0-\theta_d)\frac{\mathbf{n}_{RO\perp}}{P_d(\mathbf{P},\mathbf{P}_{obs})} \end{aligned} \tag{5.20}$$

and

$$\mathbf{S}_{rep5} = -2K_{\theta2}^2H(\theta_0-\theta_d)^2\mathbf{n}_{RO} \tag{5.21}$$

$$\mathbf{S}_{rep6} = 2K_{\theta2}^2H^2(\theta_0-\theta_d)\frac{\mathbf{n}_{RO\perp}}{P_d(\mathbf{P},\mathbf{P}_{obs})} \tag{5.22}$$

where

$$M = -K_{\theta1}(\theta_d-\theta(\omega))^2 + K_{\theta2}$$

and

$$H = p_\theta - P_d(\mathbf{P},\mathbf{P}_{obs}).$$

The relationship among the repulsive reference velocity components is shown in Fig. 5.7 and Fig. 5.8. The repulsive forces $\mathbf{S}_{rep1}$, $\mathbf{S}_{rep3}$ and $\mathbf{S}_{rep5}$ are in the opposite direction of $\mathbf{n}_{RO}$, which will keep the robot away from the obstacle. The repulsive forces $\mathbf{S}_{rep2}$, $\mathbf{S}_{rep4}$ and $\mathbf{S}_{rep6}$ are in the direction of $\mathbf{n}_{RO\perp}$, which will drive the robot for detouring. One can see that because of the new repulsive function $U_{rep}(\theta,\omega)$, another force is generated to repulse the robot away from the obstacle, shown as the red arrows in Fig. 5.7 and Fig. 5.8. When there are multiple obstacles, the repulsive force can be calculated by

$$\mathbf{S}_{rep} = \sum_{j=1}^{n_{obs}} \mathbf{S}_{rep,j} \tag{5.23}$$

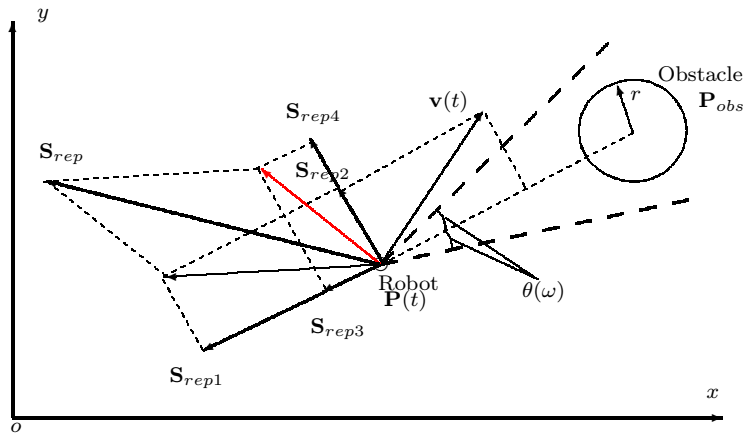


Figure 5.7: Relationship among $\mathbf{S}_{rep1}$, $\mathbf{S}_{rep2}$, $\mathbf{S}_{rep3}$ and $\mathbf{S}_{rep4}$

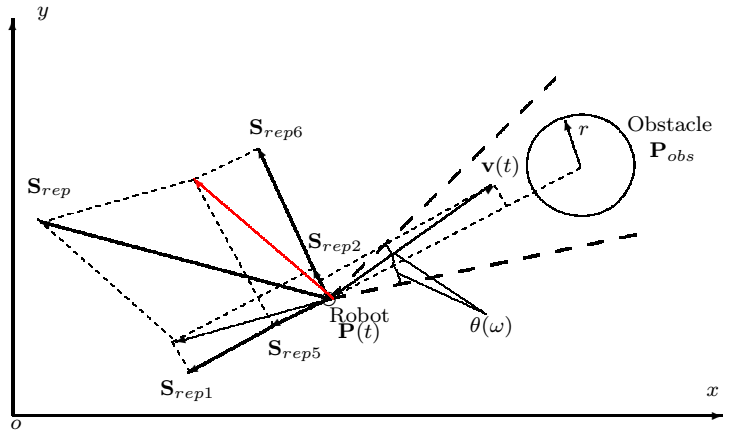


Figure 5.8: Relationship among $\mathbf{S}_{rep1}$, $\mathbf{S}_{rep2}$, $\mathbf{S}_{rep5}$ and $\mathbf{S}_{rep6}$

where n_{obs} is the number of obstacles in the sensor, and $\mathbf{S}_{rep,j}$ is the repulsive force generated by the j^{th} obstacle.

After calculation of attractive and repulsive forces, the total force can be obtained by

$$\mathbf{S}_{total} = \mathbf{S}_{att} + \mathbf{S}_{rep} \tag{5.24}$$

and the total force $\mathbf{S}_{total}$ can be used for motion planning.

The objective of motion planning is to control the robot follow the direction of the total force θ_s, then one common solution is to design an angular controller (a PID one) to force the orientation of the robot converging to the desired angle θ_S, see Ge & Cui (2002). However, one can also regard this problem as a desired velocity tracking problem, since $\frac{v_{r,y}}{v_{r,x}} = \frac{\mathbf{S}_y}{\mathbf{S}_x} = \tan\theta_s$, where $v_{r,x}$ and $v_{r,y}$ are the desired velocity on x and y axis respectively, and $\mathbf{S}_x$ and $\mathbf{S}_y$ are the component of $\mathbf{S}_{total}$ on x and y axis respectively. Thus, if one can force the robot's velocity on x (which is equal to $\dot{x}$) to track $v_{r,x}$, and make the robot's velocity on y (which is equal to $\dot{y}$) to track $v_{r,y}$, then one can ensure the robot moving with the desired angle θ_s. In fact, this new interpretation by tracking desired velocity has an important advantage, i.e. it can take into account some physical constraints of the robots, such as maximal velocity. For example, since $v_{r,y} = \frac{\mathbf{S}_y}{\mathbf{S}_x} v_{r,x} = K_s v_{r,x}$, where $K_s = \frac{\mathbf{S}_y}{\mathbf{S}_x}$ is known and deduced from potential field function. If $K_s \geq 1$, one can set the desired velocities as $v_{r,y} = v_{my}$, $v_{r,x} = \frac{v_{my}}{K_s}$, where $v_{my} = v_{max}\sqrt{\frac{K_s^2}{1+K_s^2}}$. If $K_s < 1$, one can define the desired velocities as $v_{r,x} = v_{mx}$, $v_{r,y} = K_s v_{mx}$, where $v_{mx} = \frac{v_{max}}{\sqrt{1+K_s^2}}$. In both cases, we can always ensure that the physical constraints are satisfied. The corresponding controller will be detailed in the following section by using a so-called i-PID technique.

5.4 Motion planning for non-holonomic mobile robots via i-PID controller

5.4.1 Robot model

This chapter also considers the type (2.0) robot model, let us recall it here:

$$\begin{cases} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \omega \end{cases} \tag{5.25}$$

Different methods have been proposed to use the total force for robot motion planning, for instance, in Ge & Cui (2002) the angle of the total force is used to design a PID controller to control θ, and set the linear velocity as a positive one. However this method is not robust when there are noises in the system. In De Luca & Oriolo (1994), a simple controller is obtained to track the desired velocities $v_{r,x}$ and $v_{r,y}$, by using pseudoinversion. Since in robot model (5.25) we have

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = G(\theta) \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{5.26}$$

where $G(\theta) = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{bmatrix}$, thus the controller proposed in De Luca & Oriolo (1994) is as follows:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \left[G^T G\right]^{-1} G^T \begin{bmatrix} v_{r,x} \\ v_{r,y} \end{bmatrix} \tag{5.27}$$

where $v_{r,x}$ and $v_{r,y}$ are the desired velocity on x and y axis respectively. Thus, if the robot is able to track both $v_{r,x}$ and $v_{r,y}$, then the robot is able to track the direction of the total force. However, as the author of De Luca & Oriolo (1994) has pointed out that if the initial condition of the robot is different with the reference, this approach can not converge the velocity to the reference, since the controller just solves the tracking problem in the least square sense, and the proposed controller in De Luca & Oriolo (1994) cannot guarantee the asymptotical convergence: $\dot{x} \to v_{r,x}$ and $\dot{y} \to v_{r,y}$. In order to ensure the convergence, an intuitive solution is to consider the dynamics of $\ddot{x}$ and $\ddot{y}$, and design a controller to make $\dot{x}$ and $\dot{y}$ tracking the desired velocity $v_{r,x}$ and $v_{r,y}$.

For this, let us consider the following dynamics deduced from the system

(5.25) as follows:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \tilde{G}\tilde{u} \tag{5.28}$$

where

$$\tilde{G} = \begin{bmatrix} \cos\theta & -v\sin\theta \\ \sin\theta & v\cos\theta \end{bmatrix} \tag{5.29}$$

and $\tilde{u} = [\dot{v}, \omega]^T$ being the new control input. The following section is then devoted to using the robust i-PID controller to achieve the tracking tasks.

5.4.2 i-PID controller

In this chapter, we use the controller proposed in Chapter 4 to control the robot. Different from Chapter 4, the reference to be tracked here is the desired velocity, thus the controller is a bit different:

$$\tilde{u}_k = \alpha^{-1}(x, y, \dot{x}, \dot{y})(F_{k-1} + e_k) \tag{5.30}$$

where

$$e_k = \begin{bmatrix} \ddot{v}_{r,x} \\ \ddot{v}_{r,y} \end{bmatrix} + K_p \begin{bmatrix} \dot{x} - v_{r,x} \\ \dot{y} - v_{r,y} \end{bmatrix} + K_I \int_T \begin{bmatrix} \dot{x} - v_{r,x} \\ \dot{y} - v_{r,y} \end{bmatrix} \tag{5.31}$$

and K_p and K_I are usual tuning gains. The determination of $\alpha(x, y, \dot{x}, \dot{y})$ and the calculation of F_k are exhaustively discussed in Chapter 4, and the numerical differentiation method used here is stated in section 2.4.3. Here we just note that F_k is calculated as

$$F_k = \frac{6}{T}\int_0^1 (2\delta - 1)\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} d\delta + 6\alpha(x, y, \dot{x}, \dot{y})\int_0^1 (\delta^2 - \delta)\tilde{u}d\delta \tag{5.32}$$

where $\delta \in [0, 1]$.

5.5 Simulation results

In the simulation, the parameters are set as follows: maximum linear acceleration $a_{max} = 2.0\ m/s^2$, maximum angular acceleration $\beta_{max} = 1.0\ rad/s$, obstacle influence range on $U_{rep}(\mathbf{P}, \mathbf{v})$ $p_0 = 0.3\ m$, obstacle influence range on $U_{rep}(\theta, \omega)$ $p_\theta = 0.6\ m$, time window $T = 3\ s$, influence angle $\theta_0 = \pi/4$. $K_p = 50$, $K_I = 100$,

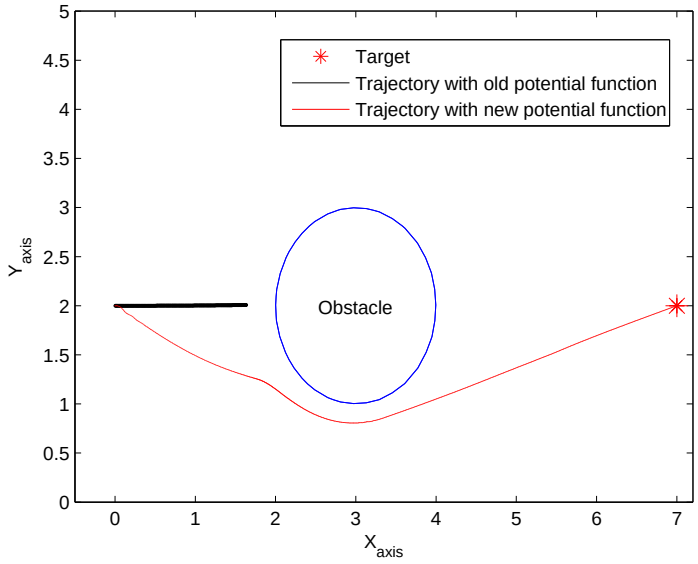


Figure 5.9: Situation with local minima

$K_{att} = 0.04$, $K_{pv} = 0.8$ and $K_{\theta 1} = K_{\theta 2} = 0.8$,

$$\alpha(x, y, \dot{x}, \dot{y}) = \begin{bmatrix} sgn(\cos\hat{\theta}) & -sgn(\sin\hat{\theta}) \\ sgn(\sin\hat{\theta}) & sgn(\cos\hat{\theta}) \end{bmatrix},$$

where $sgn(\sigma)$ is the sign function which extracts the sign of real number σ, and $\hat{\theta}$ is the estimation of θ.

The first simulation is made with the situation of local minima, in which the connection between the robot and the target passes through the center of the obstacle, and the velocity direction also passes through the center of the obstacle. As we can see in Fig. 5.9, with the classical potential function defined in Ge & Cui (2002), the robot stops in front of the obstacle. However with our new potential function, there are no local minima problems due to the introduce of new potential fields.

The second simulation is made using the original repulsive function of Ge & Cui (2002) in comparison with our new repulsive function where no local minima occurs. One can see that although the robot is able to avoid obstacles (Fig. 5.10), there are many unexpected oscillations in the generated repulsive force (Fig. 5.11), which eventually result in oscillations in the generated references

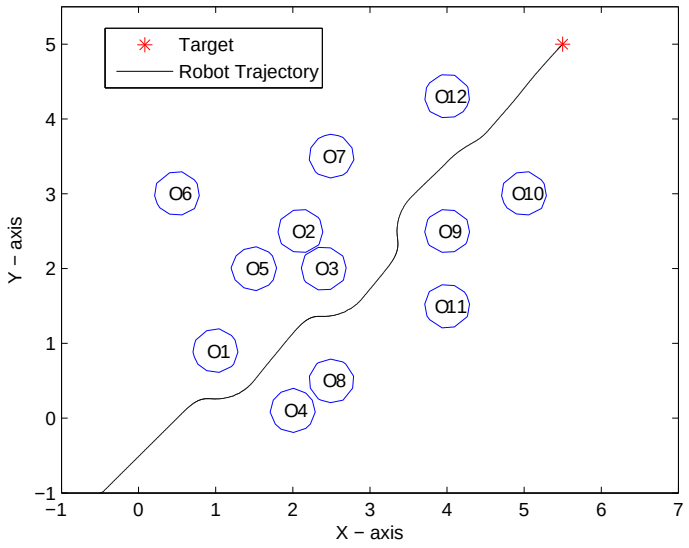


Figure 5.10: Robot trajectory with original $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$

(Fig. 5.12 and Fig. 5.13), this is detrimental for velocity tracking, the tracking errors are shown in Fig 5.14. One can see that there are also many oscillations in the generated controls (Fig. 5.15 and Fig. 5.16).

In the third simulation, only the repulsive force $\mathbf{S}_{rep}(\theta, \omega)$ is imposed on the robot, one can see in Fig. 5.17 that the robot is also able to avoid all the obstacles with only $\mathbf{S}_{rep}(\theta, \omega)$. We can see from Fig. 5.18 that $\mathbf{S}_{rep}(\theta, \omega)$ changes smoothly without sudden change, velocity tracking results are shown in Fig. 5.19 and Fig. 5.20, reference velocities also changes smoothly, the controller is able to track the velocity references, one can see in Fig. 5.21 that the tracking errors are reduced. The control inputs calculated by the i-PID controller are shown in Fig. 5.22 and Fig. 5.23. However one can notice that there is no distance constraints in the repulsive function $U_{rep}(\theta, \omega)$, so the trajectory in Fig. 5.17 is close to the obstacle.

The last simulation is shown from Fig. 5.24 to Fig. 5.31, where both repulsive forces $\mathbf{S}_{rep}(\theta, \omega)$ and $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$ are imposed on the robot to avoid obstacles. The robot trajectory is shown in Fig. 5.24, the forces generated by the potential field function are shown in Fig. 5.25 and Fig. 5.26, velocity tracking results are shown in Fig. 5.27 and Fig. 5.28, and tracking errors are shown in Fig. 5.29. The

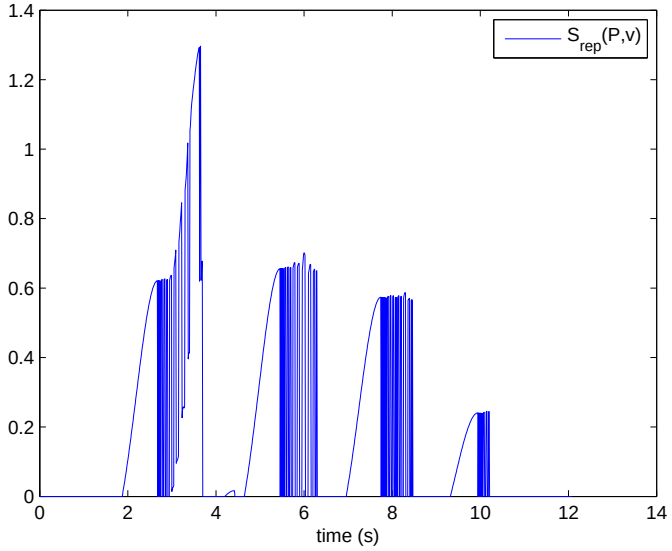


Figure 5.11: Repulsive force of original $\mathbf{U}_{rep}(\mathbf{P}, \mathbf{v})$

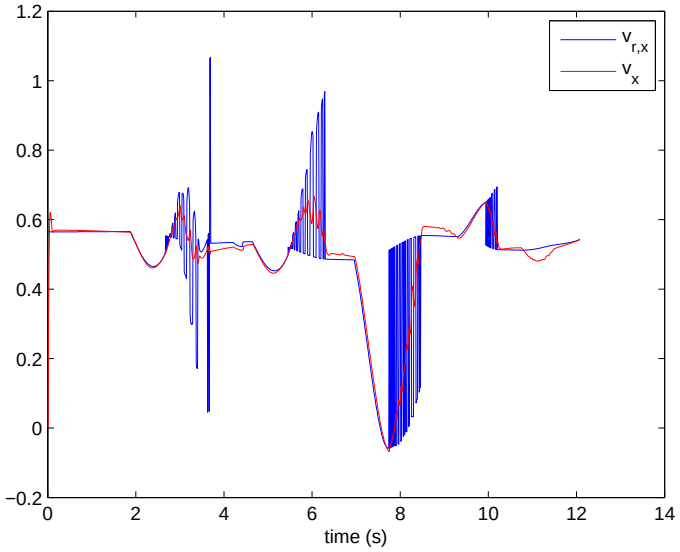


Figure 5.12: Velocity tracking in x direction with original $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$

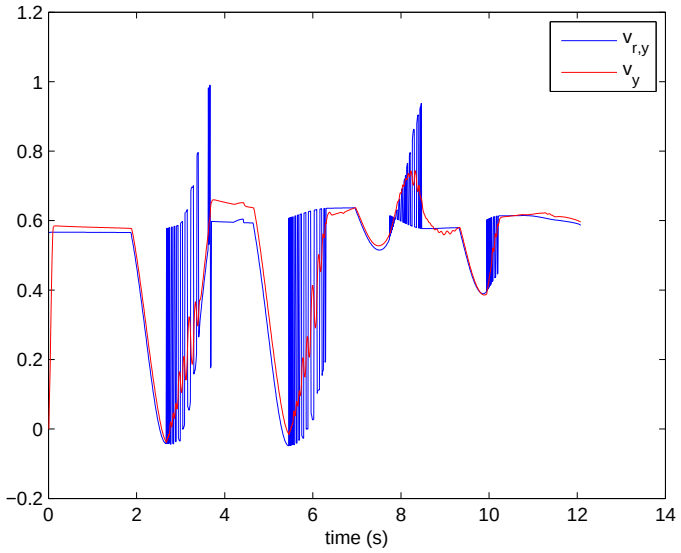


Figure 5.13: Velocity tracking in y direction with original $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$

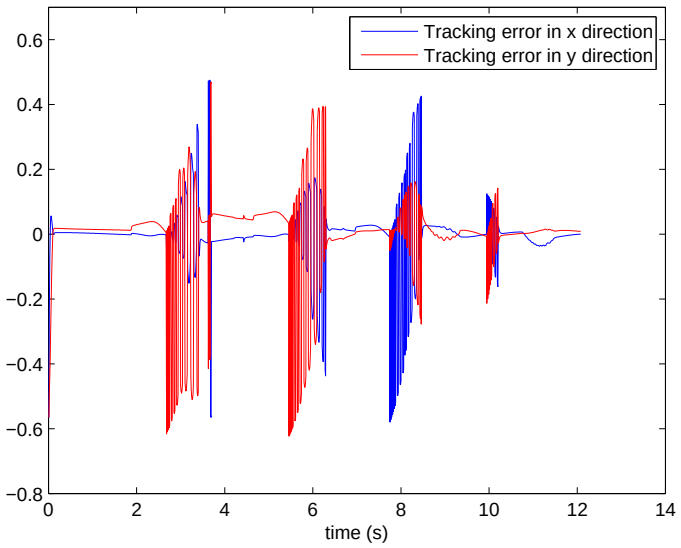


Figure 5.14: Tracking errors with original $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$

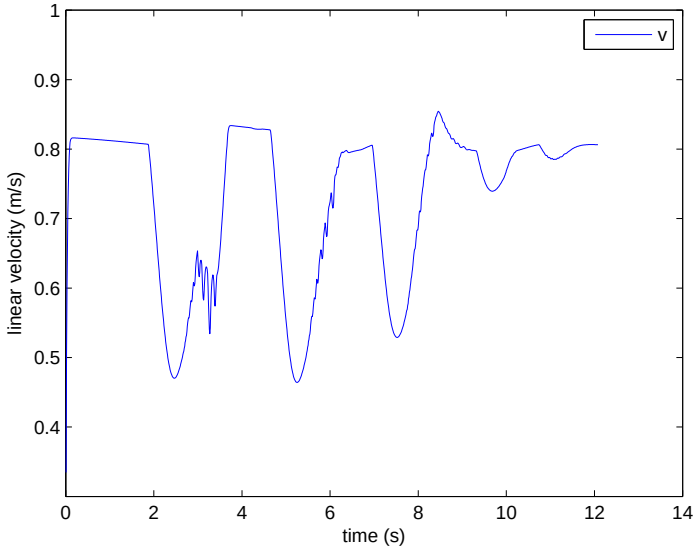


Figure 5.15: Linear velocity control with original $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$

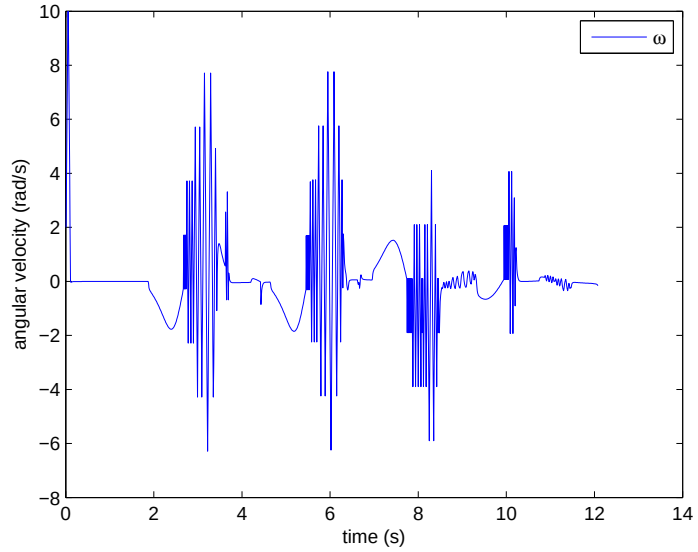


Figure 5.16: Angular velocity control with original $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$

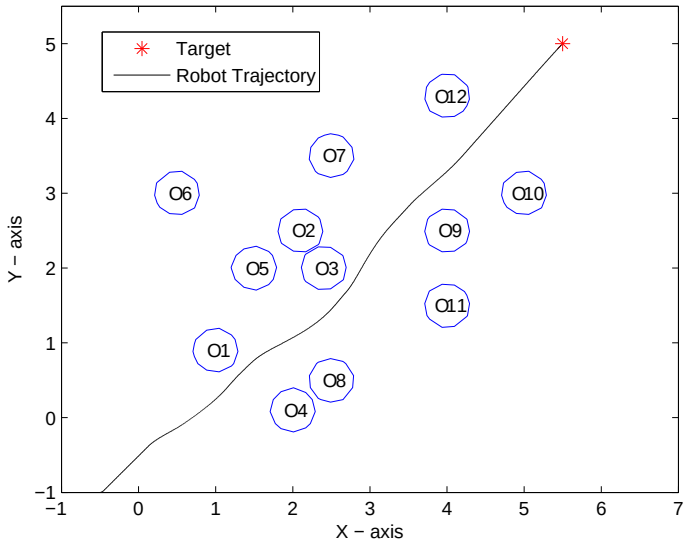


Figure 5.17: Robot trajectory with only $\mathbf{S}_{rep}(\theta, \omega)$

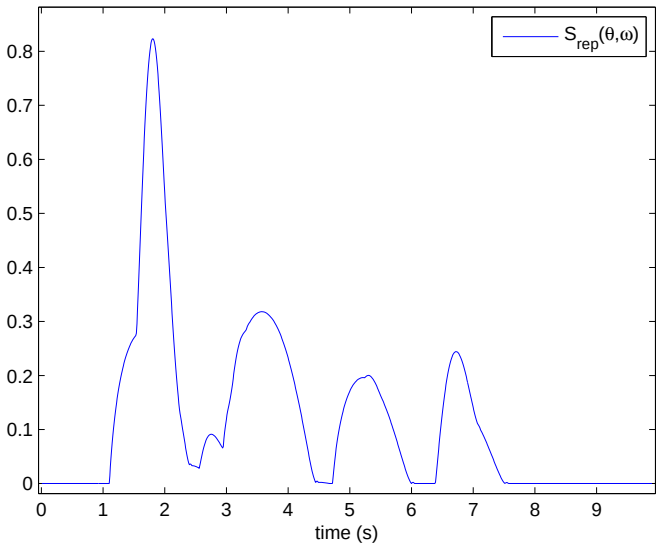


Figure 5.18: Repulsive force $\mathbf{S}_{rep}(\theta, \omega)$ with only $\mathbf{S}_{rep}(\theta, \omega)$

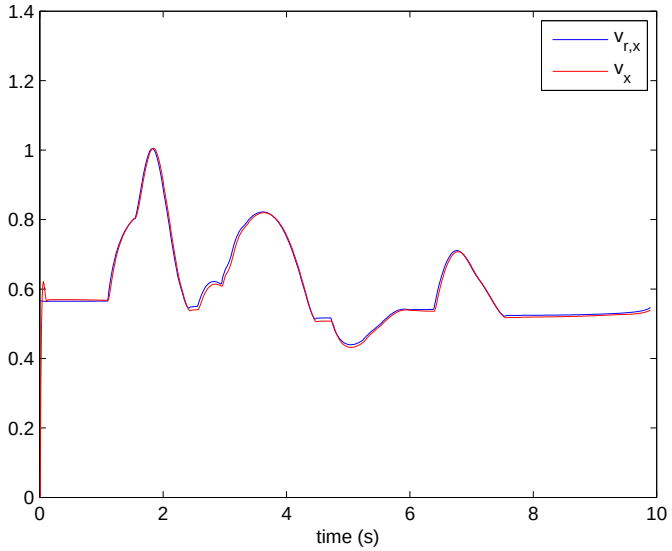


Figure 5.19: Velocity tracking in x direction with only $\mathbf{S}_{rep}(\theta, \omega)$

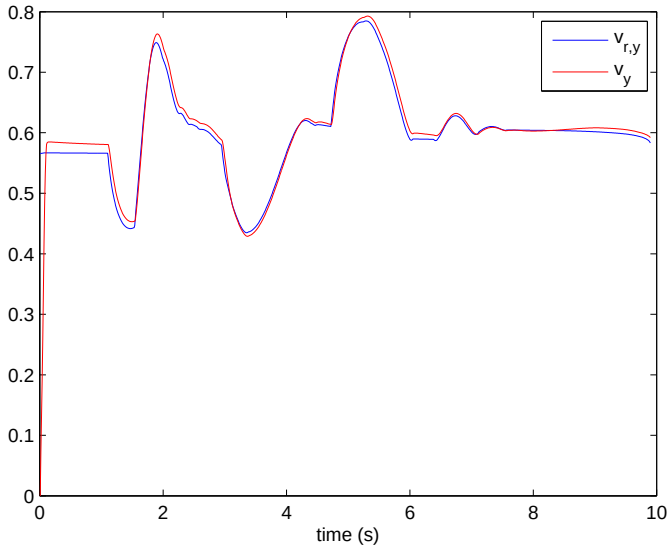


Figure 5.20: Velocity tracking in y direction with only $\mathbf{S}_{rep}(\theta, \omega)$

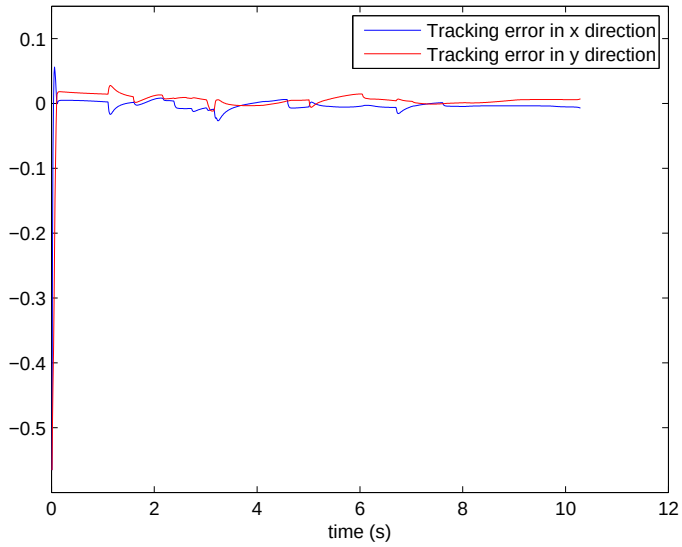


Figure 5.21: Tracking errors with only $\mathbf{S}_{rep}(\theta, \omega)$

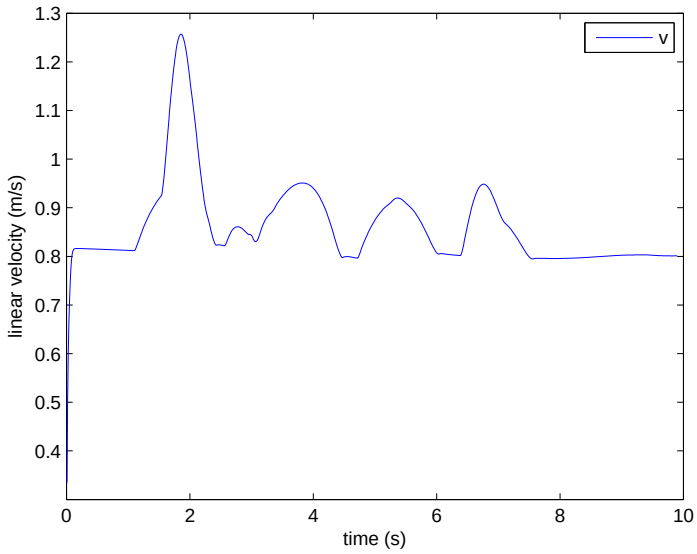


Figure 5.22: Linear velocity control with only $\mathbf{S}_{rep}(\theta, \omega)$

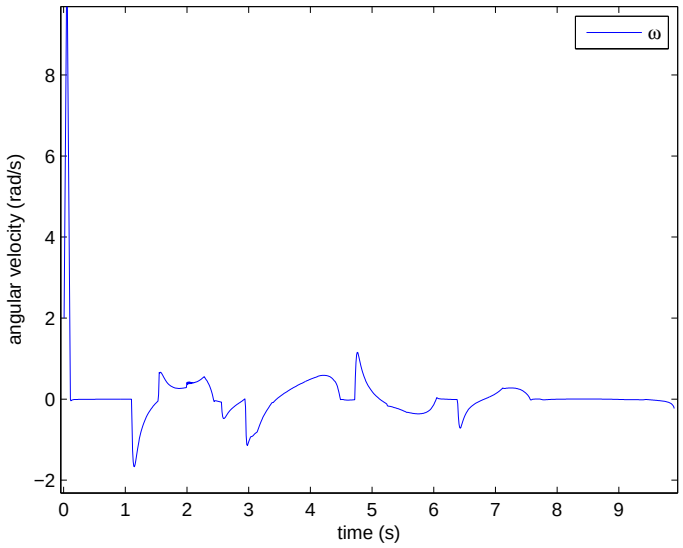


Figure 5.23: Angular velocity control with only $\mathbf{S}_{rep}(\theta, \omega)$

control inputs calculated by the i-PID controller are shown in Fig. 5.30 and in Fig. 5.31. One can see that with both repulsive forces $\mathbf{S}_{rep}(\theta, \omega)$ and $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$, the robot is able to avoid all the obstacles and keep a predefined distance from the obstacles, and there is no oscillations in repulsive forces and input controls.

A $3D$ simulation made by using ROS (Robot Operating System) and an implementation in a wifibot can be found in the following link: Video Link.

5.5.1 Switching strategy

As stated in section 5.2, when the robot is very close to obstacles, there is possibly no solution to the optimization problem. The path planning strategy proposed in this chapter always has solutions, but the robot trajectories may not be optimal. Therefore we can combine the two planning methods, when there is no solution to the optimization problem, the robot can switch to the method proposed in this chapter to guide the robot away from obstacles, and then switch back to plan optimal trajectories (see Fig. 1.5(b)).

The obstacles are assumed as circles in the previous simulations, and the potential field functions are calculated from the center of the circle. However in complex environment the obstacles are expressed as polygons as in Chapter

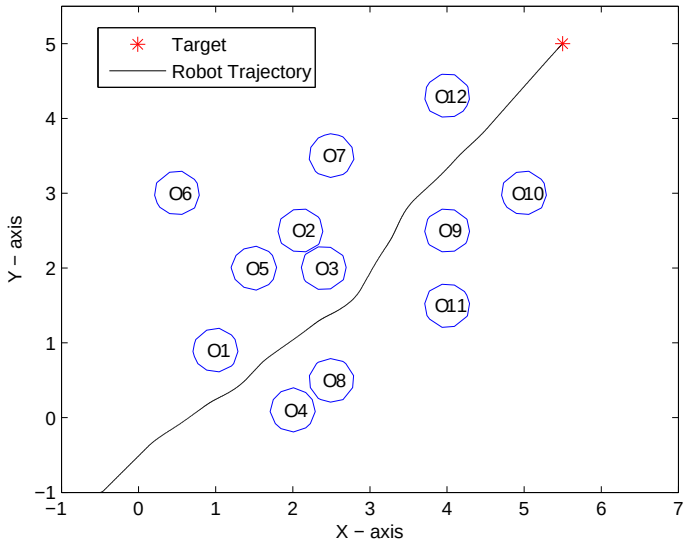


Figure 5.24: Robot trajectory with both $\mathbf{S}_{rep}(\theta, \omega)$ and $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$

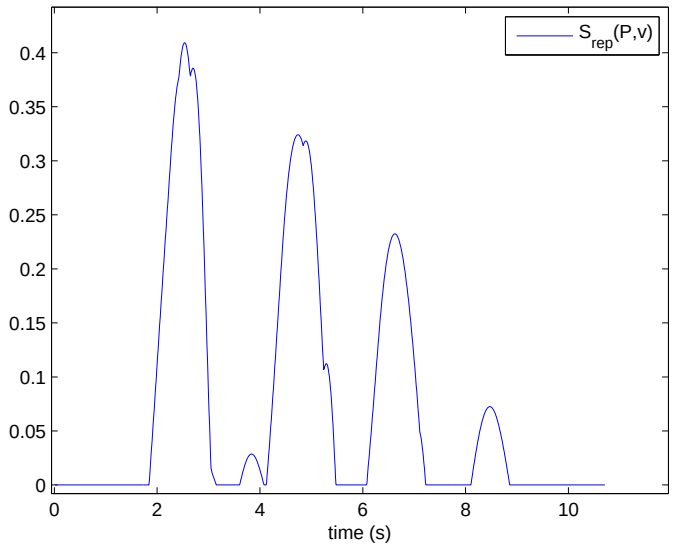


Figure 5.25: Repulsive force $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$ with both $\mathbf{S}_{rep}(\theta, \omega)$ and $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$

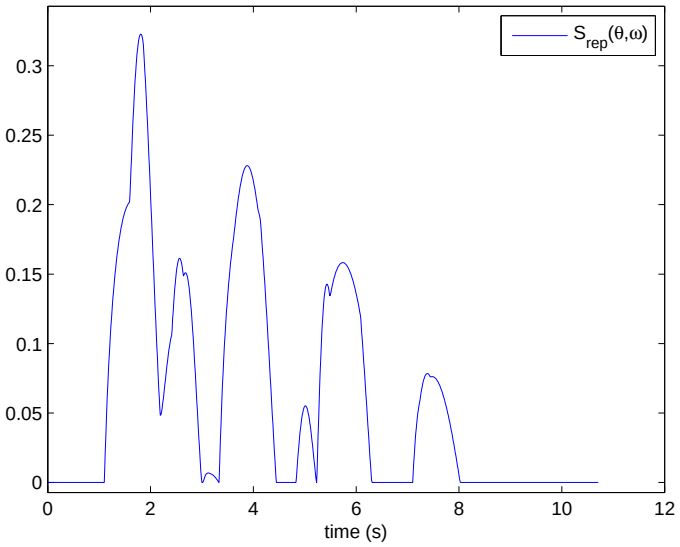


Figure 5.26: Repulsive force $\mathbf{S}_{rep}(\theta, \omega)$ with both $\mathbf{S}_{rep}(\theta, \omega)$ and $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$

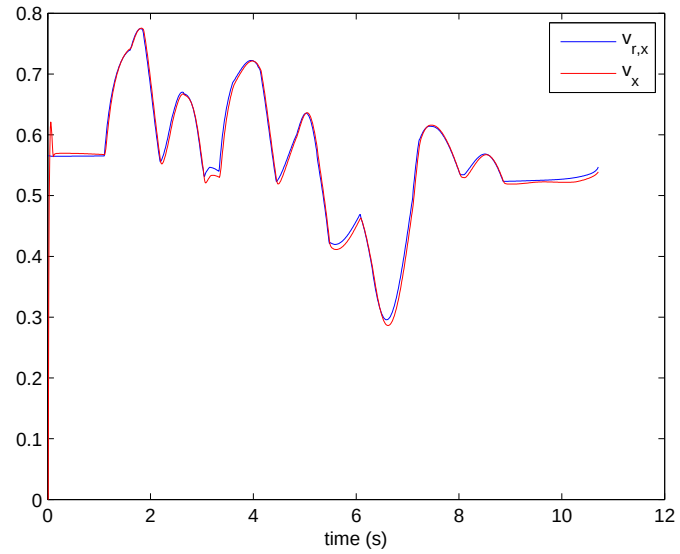


Figure 5.27: Velocity tracking in x direction with both $\mathbf{S}_{rep}(\theta, \omega)$ and $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$

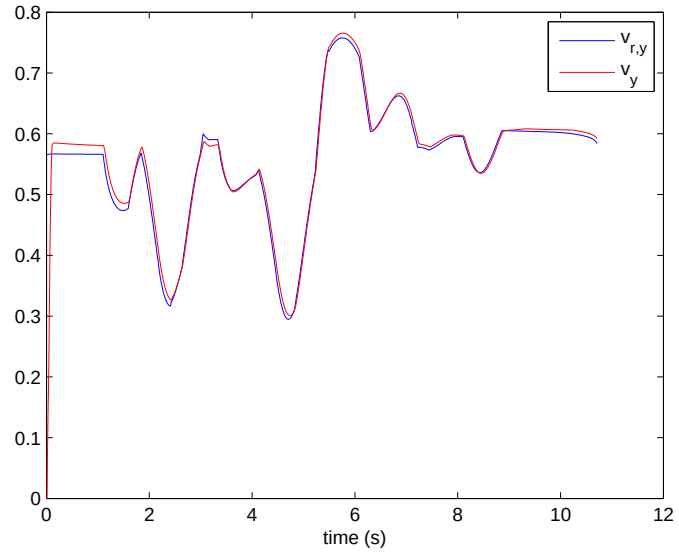


Figure 5.28: Velocity tracking in y direction with both $\mathbf{S}_{rep}(\theta, \omega)$ and $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$

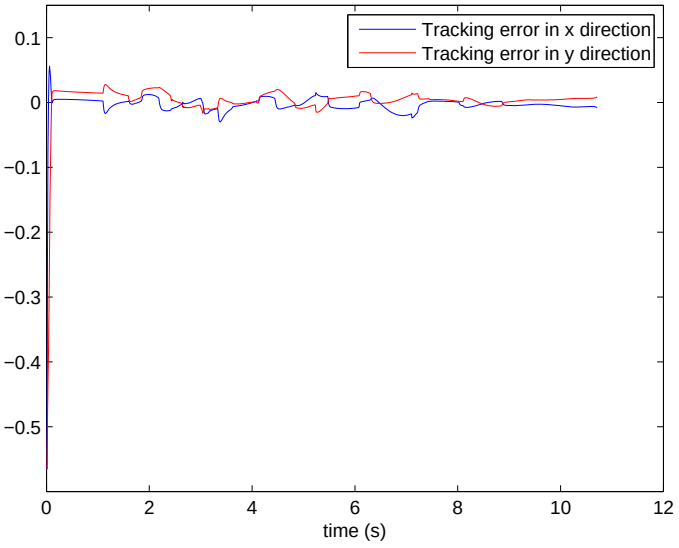


Figure 5.29: Tracking errors with both $\mathbf{S}_{rep}(\theta, \omega)$ and $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$

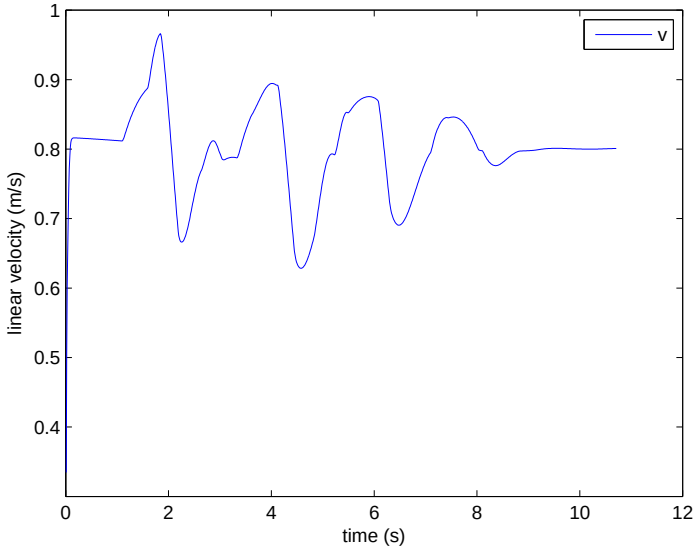


Figure 5.30: Linear velocity control with both $\mathbf{S}_{rep}(\theta, \omega)$ and $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$

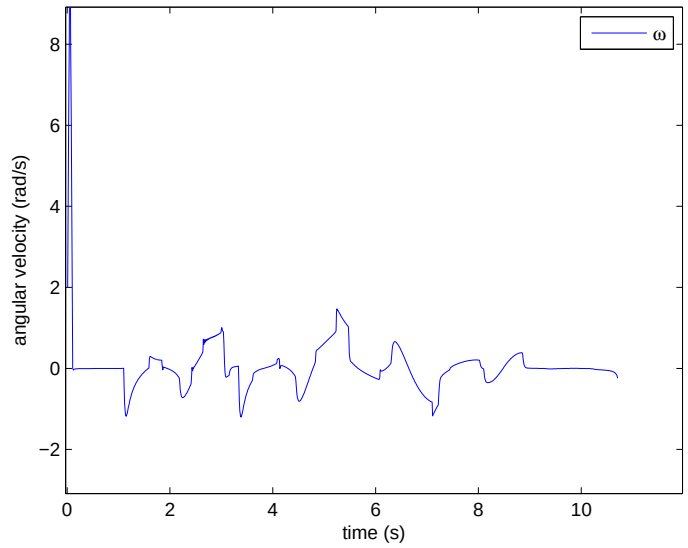


Figure 5.31: Angular velocity control with both $\mathbf{S}_{rep}(\theta, \omega)$ and $\mathbf{S}_{rep}(\mathbf{P}, \mathbf{v})$

3, thus in the following simulation, the potential functions are defined to be calculated from the closest point on the obstacle, and the distance between the robot and an obstacle is defined as the distance between the robot and the closest point on the obstacle. For example in Fig. 5.32, if the robot is the position of R_1, then closest point is o_1, and the distance between the robot and the obstacle is R_1o_1. If the robot is the position of R_2, then closest point is o_2, and the distance between the robot and the obstacle is R_2o_2.

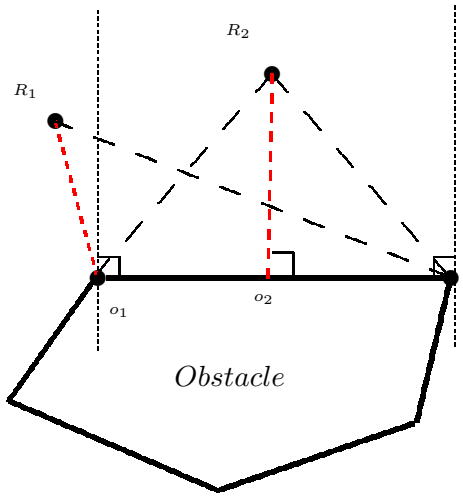


Figure 5.32: Distance between the robot and an obstacle

A simulation which combines the two methods is shown in Fig. 5.33, the blue trajectories are planned by optimization, and the red trajectories in zone A and zone B are planned by potential field functions. Zoom of zone A and zone B are shown in Fig. 5.34 and Fig. 5.35 respectively. One can see that, the robot needs to pass a narrow corridor, and there is no solution to the optimization problem when the robot is very close to the obstacle, thus the robot switch to the potential field function method, and the robot is able to move away from the obstacle, then the robot switch back and continue. The switching time is shown in Fig. 5.36.

5.6 Conclusion

This chapter proposes a new potential field method for robot motion planning. The new potential field function, which takes into account the robot orientation

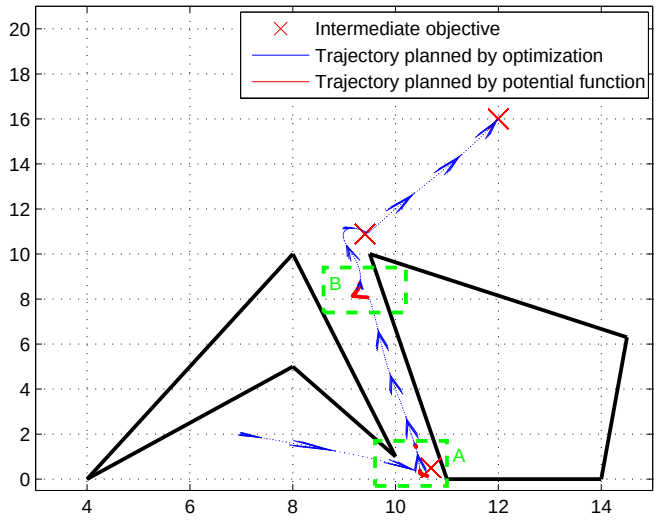


Figure 5.33: Path planning using optimization control and potential field

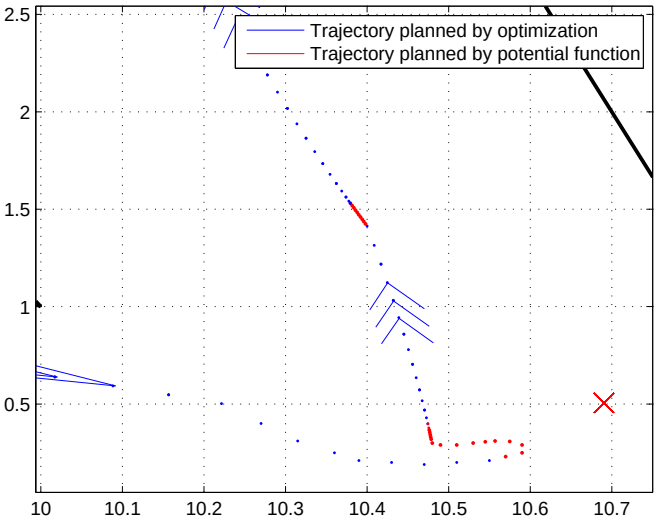


Figure 5.34: Zoom of zone A

and angular velocity, is able to solve local minima problems and produce smooth repulsive force in complex environment to avoid oscillations. Then the i-PID controller is used for robot motion planning, the force generated by potential field

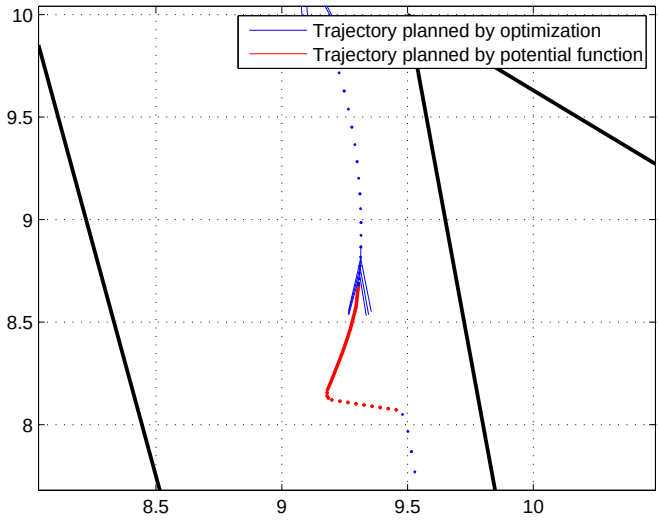


Figure 5.35: Zoom of zone B

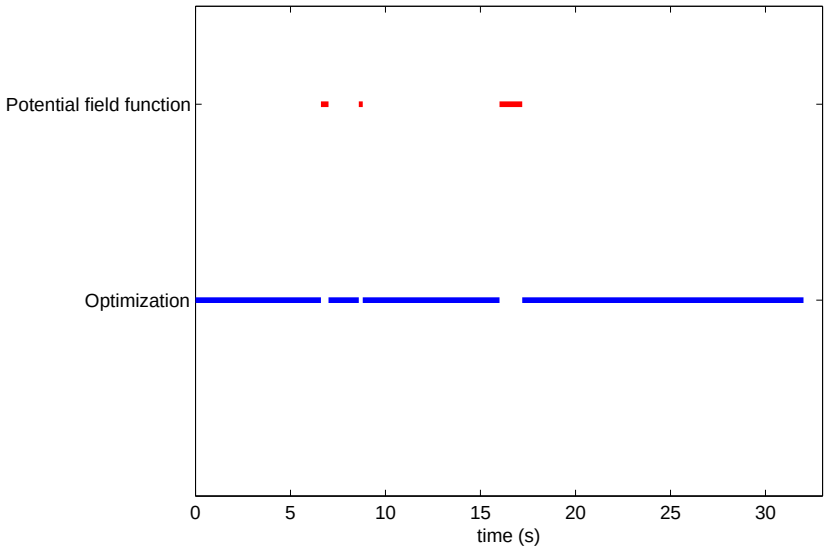


Figure 5.36: Switching Time

function is used as reference. The advantages and the efficiency of the proposed algorithm and the controller is shown thereafter via different simulations and an implementation in a real robot.

Chapter 6

Cooperative path planning for mobile robots based on visibility graph

6.1 Introduction

Multi-robot systems are currently a major focus of research in the field of robotics. The research effort into the cooperative path planning relies on the fact that the cooperative path planning between robots have the possibility to solve problems more efficiently than a single robot, since a single robot will ultimately be spatially limited. More recently, multiple autonomous mobile robots have been proposed for rescue missions Murphy *et al.* (2009); Nagatani *et al.* (2011), exploration Al Khawaldah *et al.* (2012); Kim *et al.* (2011), and even entertainments Camacho *et al.* (2006); Kim (2004).

In this chapter, the problem of interest is the cooperative path planning of autonomous mobile robots evolving in environment with obstacles. As stated in section 1.2.3, the path planning methods can be divided into two broad categories: global path planning and local path planning. However, both global path planning and local path planning have their own drawbacks. Completed environment information is required for the global path planning, which is not possible when the environment is partially detected by the robots. In local path planning, the path planned may not be globally optimized, since the environment is only partially known.

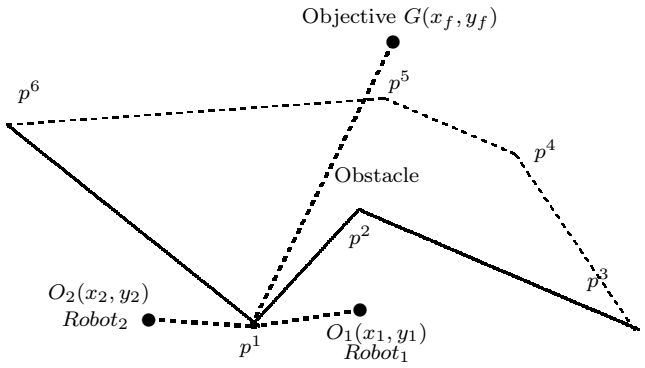


Figure 6.1: Disadvantage of Local Path Planning

The problem of path planning of multi-robot was investigated in Harinarayan & Lumelsky (1994), but the algorithm proposed does not provide optimal path. Some cooperative path planning approaches based on A^* search Chiddarwar & Babu (2011), geometric method Leroy *et al.* (1999) or artificial potential field Warren (1990) are able to provide optimal path, while however, as stated before, the constraints of the robot are not taken into account. In this chapter, the cooperative path planning is considered based on the shared map information between robots, and robots are able to reach the target by achieving the intermediate objectives generated by visibility graph. The reach of the intermediate objectives is ensured by the optimal control problem.

6.2 Problem statement

As stated above, the cooperative path planning between robots has the possibility to solve problems more efficiently than a single robot, since a single robot will ultimately be spatially limited.

For example in Fig. 6.1, there is an obstacle which is represented by its vertices $\{p^1, p^2, ..., p^6\}$. Robots are supposed to go to the desired objective $G(x_f, y_f)$ while avoid the detected obstacles. With only local information, $Robot_1$ only sees the obstacle $p^1p^2p^3$, and $Robot_2$ only sees the obstacle p^1p^6 (the solid line in the figure), and one can see that the quasi-optimal* path for for $Robot_2$ is $\{O_2p^1G\}$, and when $Robot_2$ pass over the point p^1, another new obstacle $p^1p^2p^3$ will be

*means that shortest path without considering collisions and kinematic constraints

found, then it will have to make a long detour to reach the objective, $Robot_1$ will encounter similar problems too. Nevertheless if planning with the shared information between the two robots, the path can be more quasi-optimal (O_1p^3G for $Robot_1$ and O_2p^6G for $Robot_2$), thus unnecessary detours are avoided. This motivates us to study the path planning problem based on shared information of multi-robots.

Moreover, this chapter proposes to solve the path planning problem by using visibility graph and optimal control. Visibility graph approach has the advantage of calculating the shortest collision-free quasi-optimal trajectory quickly and easy implementing Maron & Lozano-Pérez (1996), however the trajectory generated by visibility graph consists of the vertices of the obstacles, that means robots will have to touch these vertices when following the trajectory. As we know in practice, the robot can not touch the obstacles and not even close to them. Moreover, the visibility graph is an off-line path planning algorithm, and the velocity constraints, robot kinematic constraints are not considered. These drawbacks prevent the implementing of visibility graph on real mobile robots. For the problems stated above, this chapter proposes to convert the detected information of obstacles into polygons to avoid collisions of the vertices and merge two polygons when they intercross, which will be explained in the next section, and then the constraints of the robot are considered as a real time optimal control problem.

6.3 Generation of intermediate objectives based on visibility graph

The visibility graph approach has the advantage of calculating the shortest collision-free quasi-optimal trajectory quickly and easy implementing, however, as stated in the previous section, its drawback prevents its implementing on real mobile robots. To overcome these drawbacks, the polygon generation and mergence algorithms are proposed in the following subsections.

6.3.1 Polygon generation

In our research, we take into account the general shape of obstacles. Considering the limited sensor range, as stated in Chapter 3, only a portion of obstacles can

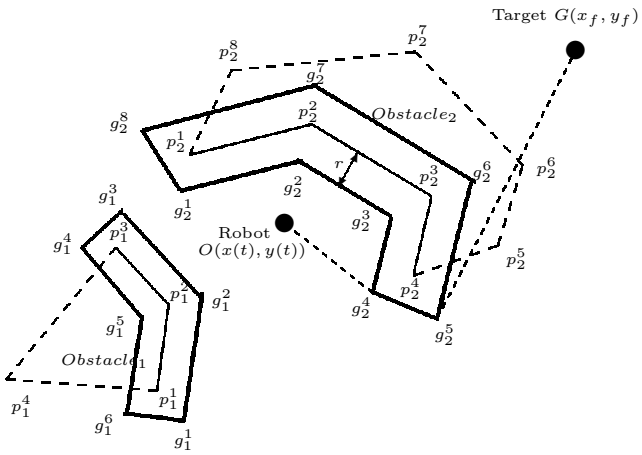


Figure 6.2: Polygon generation

be captured, thus the obstacles can be described neither as circles nor as complete polygons.

The representation of obstacles is defined as same as in Chapter 3, as shown in Fig 3.3, the visible portion of the i^{th} obstacle contour can be approximated by a succession of segments S_i^j, where $j = 1, 2, ..., q$, and q is the number of segments on an obstacle. Each segment is represented by its two end points p_i^j and p_i^{j+1}, and each point has their coordinates $(x_{p_i^j}, y_{p_i^j})$ and $(x_{p_i^{j+1}}, y_{p_i^{j+1}})$ respectively.

As stated in section 6.2, the robot can not touch the obstacles, not even close to them, so the obstacles need to be expanded to provide safe path. Let us take an example, as shown in Fig. 6.2, there are two obstacles $Obstacle_1$ and $Obstacle_2$ which are represented by their vertices respectively $\{p_1^1, ..., p_1^4\}$,$\{p_2^1, ..., p_2^8\}$. The dotted part $\{p_1^4\}$ and $\{p_2^5, p_2^6, ..., p_2^8\}$ are invisible for the moment, the robot assumes that there is no obstacle in the invisible part. Successions of joint segments are detected via the robot sensor: $\{p_1^1 p_1^2 p_1^3\}$, $\{p_2^1 p_2^2 p_2^3 p_2^4\}$. If applying the visibility graph with the visible segments, the shortest path $Op_2^4 G$ will be generated, which whereas will navigate the robot run into the obstacle. As a result, closed polygons $\{g_1^1, g_1^2, ..., g_1^6\}$ and $\{g_2^1, g_2^2, ..., g_2^8\}$ are generated according to the segments detected by the sensor. Thus a feasible safe quasi-optimal path $\{Og_2^4 g_2^5 G\}$ can be obtained via visibility graph.

Before explaining the polygon generation algorithm, let us give some notations which will be used in the sequel. Define $\mathbb{P} = \{P_i\}$ for $1 \leq i \leq N$ to be all sets of obstacle boundaries detected by the sensor, where N is the number of

detected obstacle boundaries. Note $P_i = \{p_i^j\}$ for $1 \leq j \leq N_i$ being the set of points to represent the i^{th} obstacle boundary, where N_i is the number of points. Define $List\_In$ saves all the vertices of the expanded polygon, which are on the same side of the obstacle with the robot. Define $List\_Out$ saves all the vertices of the expanded polygon, which are on the other side of the segment with the robot. Note $O(x(t), y(t))$ as the current robot position, g_{ia} and g_{ib} as two vertices to be calculated whose corresponding point on the obstacle is p^i, named as boundary vertices. Note $dis(Og_{ia})$ as the distance between the robot position $O(x(t), y(t))$ and g_{ia}. For g_{ia} and g_{ib}, if $dis(Og_{ia}) < dis(Og_{ib})$, i.e. the point g_{ia} is on the same side of the obstacle with the robot, and the point g_{ib} is on the other side of the obstacle with the robot, then we move g_{ia} onto $List\_In$, noted as $g_{ia} \rightarrow List\_In$, and $g_{ib} \rightarrow List\_Out$. For example in Fig. 6.2, for $Obstacle_2$, $List\_In_2 = \{g_2^1, g_2^2, g_2^3, g_2^4\}$, and $List\_Out_2 = \{g_2^5, g_2^6, g_2^7, g_2^8\}$.

The generation of the polygon is based on the calculation of the points detected on the obstacle. The points detected are distinguished as *disjoint points* and *joint points*, let us firstly start with the *disjoint points*.

6.3.1.1 Disjoint points

Let us take one obstacle ($p_1p_2p_3$) as an example (Fig. 6.3(a-b)), p^1 is the disjoint endpoint we concerned, r is the obstacle avoidance criterion, and point p' is at a distance of r to the point p^1 on the extension of segment from p^2 to p^1, whose coordinate can be easily obtained as (p'_x, p'_y), there are two cases to be distinguished.

Case 1. *If the slope of line p^1p^2 not exists, shown in Fig. 6.3(a), the coordinate of the boundary vertices $g_{1a}(p'_x + x, p'_y)$ and $g_{1b}(p'_x - x, p'_y)$ can be obtained easily.*

Case 2. *If the slope k of the segment $p_i^1p_i^2$ exists, shown in Fig.6.3(b), the coordinate of boundary vertices $g_{1a}(\frac{r \times k}{\sqrt{k^2+1}} + p'_x, \frac{-r}{\sqrt{k^2+1}} + p'_y)$, and point $g_{1b}(\frac{-r \times k}{\sqrt{k^2+1}} + p'_x, \frac{r}{\sqrt{k^2+1}} + p'_y)$ can be obtained.*

After calculation, if $dis(Og_{ia}) < dis(Og_{ib})$, then $g_{ia} \rightarrow List\_In$, $g_{ib} \rightarrow List\_Out$, and vice versa.

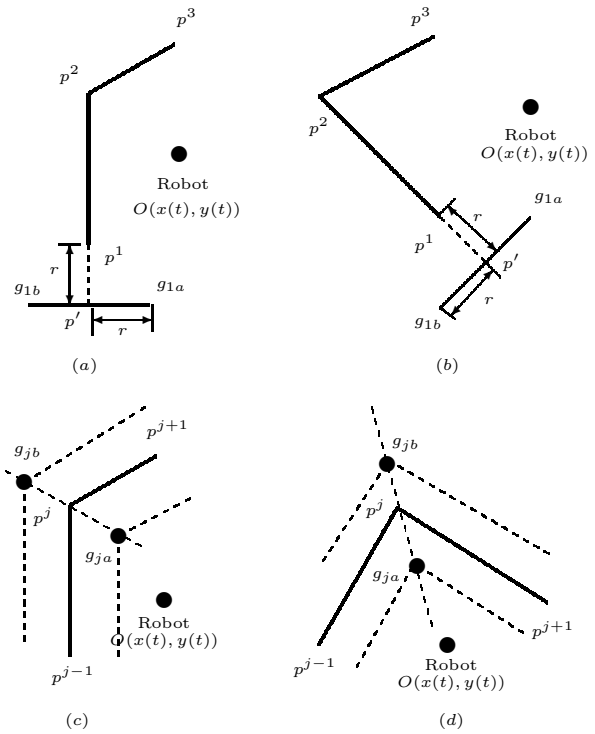


Figure 6.3: Polygon generation

6.3.1.2 Joint points

Take the joint point $p^j(x_j, y_j)$ as an example (seen in Fig. 6.3(c-d), whose previous point is $p^{j-1}(x_{j-1}, y_{j-1})$ and following point is $p^{j+1}(x_{j+1}, y_{j+1})$. One can notice that the boundary vertices g_{ja}, g_{jb} to be calculated are at a distance of r to both line $p^{j-1}p^j$ and p^jp^{j+1}, thus the point can be calculate by this distance condition. There are also two cases for the joint points.

Case1: If the slop of one segment connected with the joint point p^j not exists, for example the slop of $p^{j-1}p^j$ not exists and the slop of p^jp^{j+1} is k_{j+1}, shown in Fig. 6.3(c). With the above distance condition we have 4 solutions:

$$\begin{cases} g_{x1} = x_j + r \\ g_{y1} = y_j + rk_{j+1} \pm r\sqrt{k_{j+1}^2 + 1} \end{cases}$$

$$\begin{cases} g_{x2} = x_j - r \\ g_{y2} = y_j - rk_{j+1} \pm r\sqrt{k_{j+1}^2 + 1} \end{cases}$$

Case2: If both slop k_{j+1} and k_{j-1} exist, shown in Fig. 6.3(d), we also have 4

solutions:

$$\begin{cases} g_{x1} = \frac{x_j k_{j+1} - x_j k_{j-1} + r\sqrt{k_{j+1}^2+1} \pm r\sqrt{k_{j-1}^2+1}}{k_{j+1}-k_{j-1}} \\ g_{y1} = \frac{x_j k_{j+1} - x_j k_{j-1} + r k_{j-1}\sqrt{k_{j+1}^2+1} \pm r k_{j+1}\sqrt{k_{j-1}^2+1}}{k_{j+1}-k_{j-1}} \end{cases}$$

$$\begin{cases} g_{x2} = \frac{x_j k_{j+1} - x_j k_{j-1} - r\sqrt{k_{j+1}^2+1} \pm r\sqrt{k_{j-1}^2+1}}{k_{j+1}-k_{j-1}} \\ g_{y2} = \frac{x_j k_{j+1} - x_j k_{j-1} - r k_{j-1}\sqrt{k_{j+1}^2+1} \pm r k_{j+1}\sqrt{k_{j-1}^2+1}}{k_{j+1}-k_{j-1}} \end{cases}$$

After having 4 solutions, find out g_{ia} with $dis(Og_{ia}^j)$ of the smallest value, and g_{ib} with $dis(Og_{ib}^j)$ of the largest value, then $g_{ia}^j \rightarrow List\_In$, $g_{ib}^j \rightarrow List\_Out$.

After calculating the points on the obstacles one by one as stated above, $List\_In$ and $List\_Out$ are obtained for each obstacle, thus the closed boundary of obstacles are obtained.

6.3.2 Polygon mergence algorithm

When there are more than one robot in the system, there will be a shared map between robots. Since non-intersecting closed polygons are required for the visibility graph, two intercrossed polygons generated from two obstacles need to be merged as a new closed polygon.

Assuming that obstacles $P_i\{p_i^1, p_i^2, ..., p_i^n\}$ and $P_j\{p_j^1, p_j^2, ..., p_j^m\}$ intercross with each other, P_i and P_j are in the sensor range of $robot_1$ and $robot_2$ respectively. As defined before, $List\_In_i$ and the $List\_Out_i$ have all the boundary vertices of P_i, the generated boundary $G_i\{g_i^1, g_i^2, ..., g_i^{2n}\}$ is obtained by connecting the points on $List\_In_i$ and $List\_Out_i$ one by one in order, equivalently we have $List\_In_j$, $List\_Out_j$ and $G_j\{g_j^1, g_j^2, ..., g_j^{2m}\}$ of P_j. Denote $p_i^k \leftrightarrow p_j^m$ as connecting p_i^k and p_j^m, the segment p_1p_2 is noted as $\overline{p_1p_2}$, and function $Intersection(ab, cd)$ returns the intersection point of line ab and cd. Now let us consider the condition of boundary mergence.

Theorem 6.3.1. *Two obstacles P_i and P_j detected by two robots intercross with each other, there are 7 possible cases.*

Proof. The algorithm starts to search every point of the obstacle from the first point p_i^1 of P_i until the endpoint, and then search the points on P_j. The points of the obstacle can de divided into two types: disjoint points and joint point. For each type of points, they are either on the other obstacle or not, so there are 4

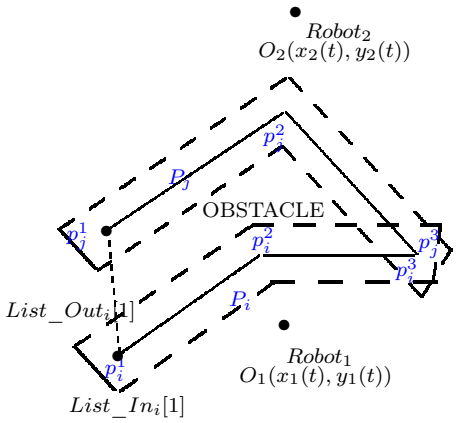


Figure 6.4: Polygon mergence of $Case1$

possible cases. However when the disjoint points is on the other obstacle, there are three different situations, so we have six cases now. Moreover, there is one more special case: the obstacle detected by one robot is a part of or exactly the other obstacle. To sum up, there are seven possible cases. □

Now let us clarify the seven possible cases.

Case1: Disjoint point is not on the other obstacle. Point p_i^1 is the disjoint point of obstacle P_i, if $p_i^1 \notin P_j$, then $List\_In_i[1] \leftrightarrow List\_Out_i[1]$, shown as the black solid line in Fig. 6.4.

Case2: Disjoint point is the disjoint point of the other obstacle. As shown in Fig. 6.5. Point p_i^1 is the disjoint point of obstacle P_i, and point p_j^1 is the disjoint point of the obstacle P_j, two obstacle jointed at the point $p_i^1(p_j^1)$, then if $p_i^1 p_i^2 \nparallel p_j^1 p_j^2$ (shown in Fig. 6.5(a)), calculate new point

$$\begin{aligned} T_1 &= Intersection(List\_Out_i[1]List\_Out_i[2], List\_Out_j[1]List\_Out_j[2]) \\ T_2 &= Intersection(List\_In_i[1]List\_In_i[2], List\_In_j[1]List\_In_j[2]) \end{aligned}$$

then let $List\_In_i[1] = List\_In_j[1] = T_2$, $List\_Out_i[2] \leftrightarrow T_1$, $List\_Out_j[2] \leftrightarrow T_1$, shown as the black solid segments in the figure.

Case3: Disjoint point is the disjoint point of the other obstacle. If $p_i^1 p_i^2 \parallel p_j^1 p_j^2$, as shown in Fig. 6.5(b), then $List\_In_i[1] \leftrightarrow List\_In_j[1]$, and delete point $List\_Out_i[1]$, $List\_Out_j[1]$.

Case4: Disjoint point is the joint point of the other obstacle.

Shown in Fig. 6.6, if point p_i^1 is the disjoint point of obstacle P_i, and $p_i^1 \in P_j$

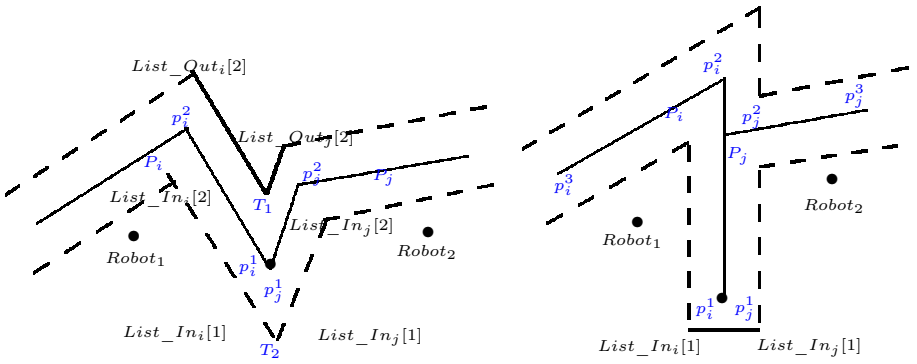


Figure 6.5: Polygon mergence of $Case2$ and $Case3$

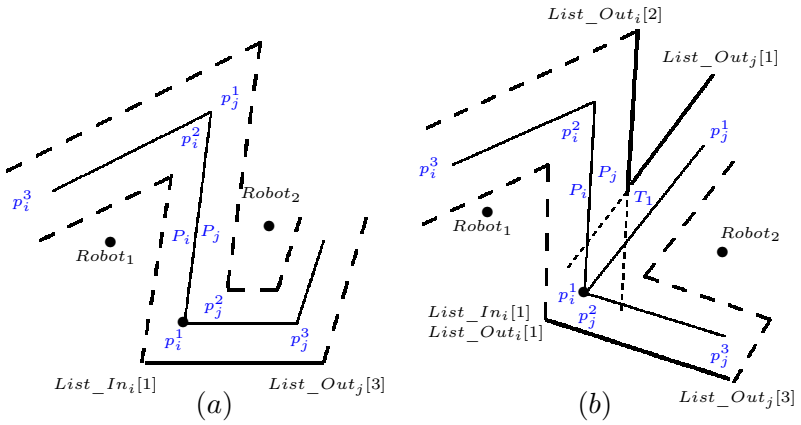


Figure 6.6: Polygon mergence of $Case4$

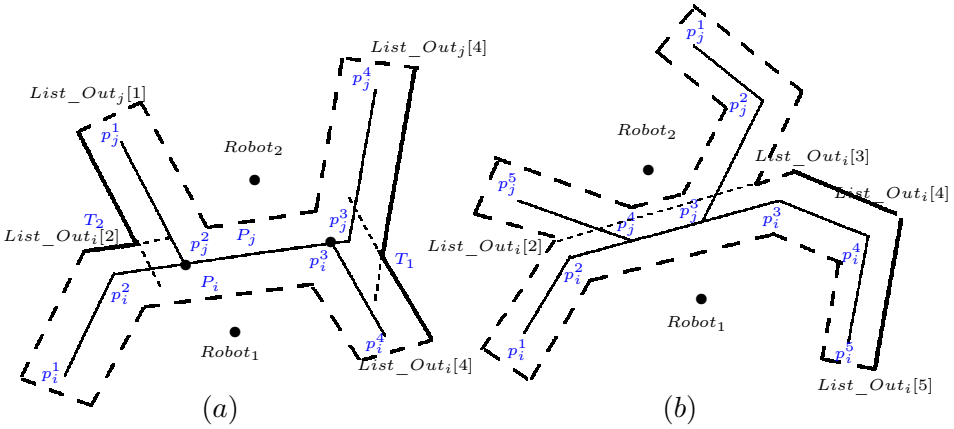


Figure 6.7: Polygon mergence of $Case5$ and $Case6$

but not the disjoint point of P_j. For example $p_j^2 = p_i^1$, then check the following point p_j^3 and previous point p_j^1 of point p_j^2, whether they satisfy the following conditions:

(1) Not on obstacle P_i;

(2) line $p_j^1p_j^2 \nparallel p_i^1p_i^2$ or $p_j^2p_j^3 \nparallel p_i^1p_i^2$;

(3) Segment $\overline{List\_Out_j[1]List\_Out_j[2]} \cap \overline{p_i^1p_i^2} = \emptyset$
or $\overline{List\_Out_j[2]List\_Out_j[3]} \cap \overline{p_i^1p_i^2} = \emptyset$.

If the point satisfies all the conditions stated above, for example, point p_j^3 in the Fig. 6.6(a) and 6.6(b), then $List\_In_i[1] \leftrightarrow List\_Out_j[3]$.

While if the point only satisfies the first two conditions, but not the third one, for example point p_j^1 in Fig. 6.6(b), segment $\overline{List\_Out_j[1]List\_Out_j[2]} \cap \overline{p_i^1p_i^2} \neq \emptyset$, then calculate new point

$$T_1 = Intersection(List\_Out_i[1]List\_Out_i[2], List\_Out_j[1]List\_Out_j[2])$$

and $List\_Out_i[2] \leftrightarrow T_1$, $List\_Out_j[1] \leftrightarrow T_1$, shown as the black solid segments in the figure.

Case5: Joint point is on the other obstacle.

If a jointed point of obstacle P_i is also on obstacle P_j, take the point p_j^2 in Fig. 6.7(a) as an example. Then find out the segment which intercross with the other obstacle but not on or parallel with the other obstacle, for example, i.e. $\overline{p_j^1p_j^2} \in P_j$, $\overline{p_j^1p_j^2} \cap p_i^2p_i^3 = p_j^2$, $\overline{p_i^3p_i^4} \in P_i$, $\overline{p_i^3p_i^4} \cap p_j^2p_j^3 = p_i^3$ and $\overline{List\_Out_j[1]List\_Out_j[2]} \cap$

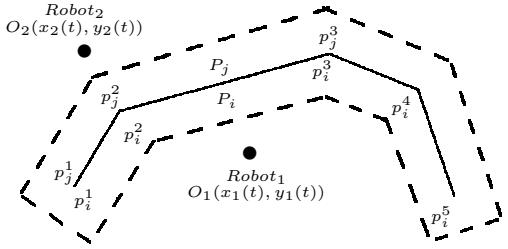


Figure 6.8: polygon mergence

$\overline{List\_Out_i[2]List\_Out_i[3]} \neq \emptyset$, then calculate new point

$$T_2 = Intersection(List\_Out_i[2]List\_Out_i[3], List\_Out_j[1]List\_Out_j[2])$$

and $List\_Out_i[2] \leftrightarrow T_2$, $List\_Out_j[1] \leftrightarrow T_2$.

Same with point p_i^3, calculate new point T_1 and $List\_Out_i[4] \leftrightarrow T_1$, $List\_Out_j[4] \leftrightarrow T_1$. Shown as the black solid segments in Fig. 6.7(a).

Case6: Point is not on the other obstacle. If the point is not on the other obstacle, p_i^k for example, then check

(1) p_i^k is not the first point of obstacle P_i;

(2) $\overline{p_i^k p_i^{k-1}} \cap P_j = \emptyset$, where the point p_i^{k-1} is the previous point of point p_i^k, which has been searched.

If the point satisfies the two conditions, then $List\_Out_i[k] \leftrightarrow List\_Out_i[k-1]$.

As we can see in Fig. 6.7(b), for example, point p_i^3 is not the first point of P_i and $p_i^3 \notin P_j$. However $\overline{p_i^2 p_i^3} \cap P_j \neq \emptyset$, thus point $List\_Out_i[2]$ and point $List\_Out_i[3]$ are not connected. Point p_i^4 and point p_i^5 satisfy all the three conditions, thus $List\_Out_i[3] \leftrightarrow List\_Out_i[4]$, $List\_Out_i[4] \leftrightarrow List\_Out_i[5]$.

Case7: When an obstacle in the sensor is a part of or exactly the other obstacle, for example obstacle $P_j \in P_i$, then take the generated polygon of obstacle P_i as the merged polygon.

Till now the the outer boundary is merged, then connect all the points in $List\_In$ one by one in order, $List\_In[k] \leftrightarrow List\_In[k+1]$, where $k = \{1, 2, ..., m_{Li}-1\}$, m_{Li} is the number of points in $List\_In$, thus the merged polygon is obtained.

6.3.3 Generation of intermediate objectives

Once the map is established after robots sharing their map information, the quasi-optimal path can be generated by visibility graph, and the path consist of a series of points which can be taken as intermediate objectives for robots. The robots can reach the final objective by achieving every intermediate objective in order. Let us define $IO\_List = \{p_k, G\}$ for $1 \leq k \leq m_{IO}$ saves the selected intermediate objectives, where m_{IO} is the number of intermediate objectives.

However as stated in previous section, the quasi-optimal paths generated by visibility graph consist of only straight line segments, the velocity constraints, kinematic constraints and non-holonomic constraints of the robot are not considered, so that the optimal path planning algorithm is applied to calculate the robot trajectory between the current robot position and the intermediate objective, and ensure all the constraints of the robot, which will be explained in the next section.

6.4 Path planning based on intermediate objectives

The reaching of the intermediate objectives can be ensured by either optimization algorithm proposed in Chapter 3 or the motion planning method proposed in Chapter 5. In this chapter we use the optimal path planning algorithm proposed in Chapter 3. The problem formulation, the representation of obstacles, the distance between the robot and obstacles, the system model and the optimal function are defined as same as those in Chapter 3. The problem is also considered as an optimal control problem over a receding horizon, with T_c being the update period, and $0 < T_c < T_p$, where T_p is the trajectory planning horizon. However the switching region is defined differently, let us clarify it in the following.

6.4.1 Reach switching region

Let take the following simple obstacle as an example (shown in Fig. 6.9) to explain the algorithm. For $Robot_1$, suppose that the list of intermediate objectives

is already obtained by the visibility graph approach:

$$IO\_List = \{g_i^2, g_i^1, G\}$$

Then the robot is navigated to reach the first point in IO_List, i.e. g_i^2, then g_i^1, then G. Thus the problem now becomes an optimal problem with constraints $C_1 - C_4$ (constraints stated in Chapter 3) by minimizing the cost function with respect to current intermediate objective, i.e.

$$\min \int_{T_p} \left\| O - g_i^2 \right\|^2 dt, \text{ s.t. } C_1 - C_4 \tag{6.1}$$

As stated in Chapter 3, the solution of this optimal problem yields optimal trajectories, then one can get optimal control (v, ω) according to the flatness property of the system, or use the i-PID controller to track optimal trajectories.

The optimal path planning method is able to guarantee the robot moving towards the intermediate objective, and we need to define when to switch the intermediate objective, and continue to go to the next intermediate objective or to replan the path.

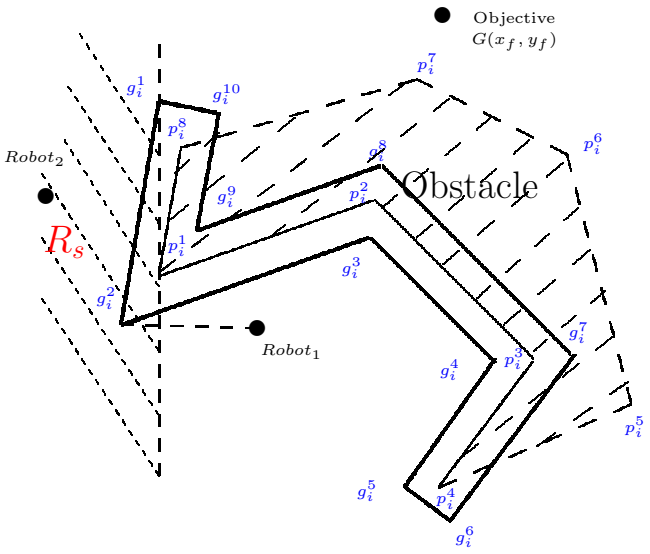


Figure 6.9: Reach Switching Region

Firstly, let define the switching region. We use the connection line of the next intermediate objective and the corresponding point of current intermediate

objective on the obstacle to define the switching region. As shown in Fig. 6.9, the intermediate objectives selected for $Robot_1$ are $\{g_i^2, g_i^1, G\}$, so point p_i^1 is the corresponding point of g_i^2, and the switching region R_s can be defined as the left region of line $p_i^1 g_i^1$, this is because:

(1) the current intermediate objective will always be in the switching region, thus the robot is able to enter the switching region when moving towards the intermediate objective;

(2) next intermediate objective can always be seen in this region if there is no new obstacle, robots will know whether to replan or go to next intermediate objective in this region.

The robot position at $t = 0$ and $t = T_c$ are used to judge whether the robot enters the switching region R_s or not. The function of the line $p_i^1 g_i^1$ can be obtained as $f(x, y) = 0$, if $f(x(0), y(0))f(x(T_c), y(T_c)) < 0$, one can judge that the robot has already reached the switching region R_s, and then the robot can see next intermediate objective, so p_i^1 has been passed over, then g_i^2 is removed from IO_List.

6.4.2 Algorithm description

Firstly robots scan the environment and expand the segments that are scanned, then robots share the information, and merge the obstacles boundaries if necessary. After that, visibility graph is applied to generate the intermediate objectives list IO_List. Given the IO_List, one can solve the optimal path planning problem over T_p to get the optimal trajectories over T_p. Then controls (the open loop control or the i-PID controller) are applied for the robot for only a time period over $[0, T_c]$. Every time when $t = T_c$, check whether the robot gets into the switching region or not, and repeat the procedures as before. The algorithm stops when the robots reach the final target G.

6.5 Simulation results

In order to show the feasibility and the efficiency of the proposed algorithm, two simulations of simple and complex environment are made in contrast with two simulations without cooperation between robots in the same environment. The

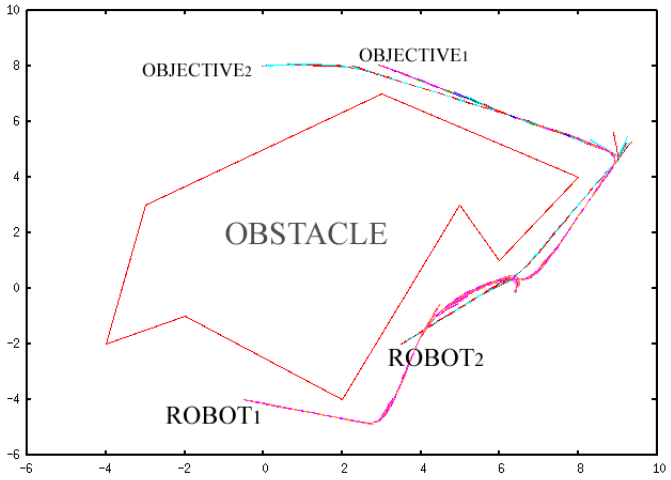
Table 6.1: comparison of Simulation result in simple environment

	Robot	Running Time (s)	Steps	Trajectory Length
Without Cooperation	R_1	10.99	292	33.80
	R_2	6.97	132	18.55
With Cooperation	R1	3.982	114	19.13
	R_2	4.083	121	18.59

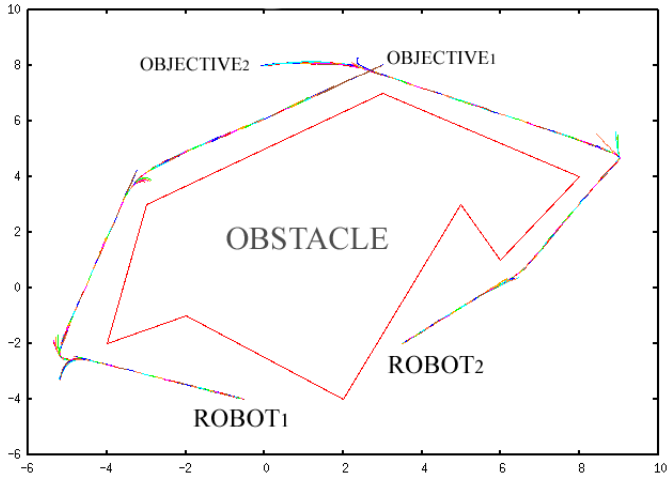
simulation settings are as follows: the range of the robot sensor is 3 meters; maximum velocity of the robot $v_{max} = 1.0\ m/s$, maximum acceleration $a_{max} = 1.0\ m/s^2$, maximum angular velocity $\omega_{max} = 1.0\ rad/s$, maximum angular acceleration $\dot{\omega} = 1.0\ rad/s^2$. The planning horizon interval $T_p = 2\ s$, and the update period $T_c = 0.2\ s$.

For the simple environment depicted in Fig. 6.10, $Robot_1$ starts at point $(-0.5, -4)$ going to the point $(3, 8)$, the $Robot_2$ starts at point $(3.5, -2)$, going to the point $(0, 8)$. In Fig. 6.10(a), there is no cooperation between the robots. In Fig. 6.10(b), the paths are planned with cooperation. It is clearly seen that, $Robot_1$ moved towards right because it is optimal when it plans locally, however there is a long way to travel. Instead, with the shared information, $Robot_1$ is able to calculate the path optimally, and long detour is avoided. A detail comparison is shown in Table 6.1, one can see that it takes $Robot_1$ less than half time to reach the target when planning with the shared information, the trajectories are well optimized.

More complex scenarios are depicted in Fig. 6.11, in which $Robot_1$ starts from $(-2.3, -4.5)$ to $(-2, 7)$, and $Robot_2$ starts from $(1.5, -5)$ to $(-5, 8)$. The dotted segments in the figure are the parts that are not detected by the robots. One can see that with only local information (shown in Fig. 6.11(a)), $Robot_1$ moved toward right first, and $Robot_2$ moved toward left first, which however made a long detour to reach the target. While in Fig. 6.11(b), the robots are able to plan efficiently with shared information, the unnecessary detours are avoided. A detail comparison is shown in Table 6.2, it is clearly seen that the time, step and length of the trajectories are well optimized when planning with shared information.



(a) Without cooperation

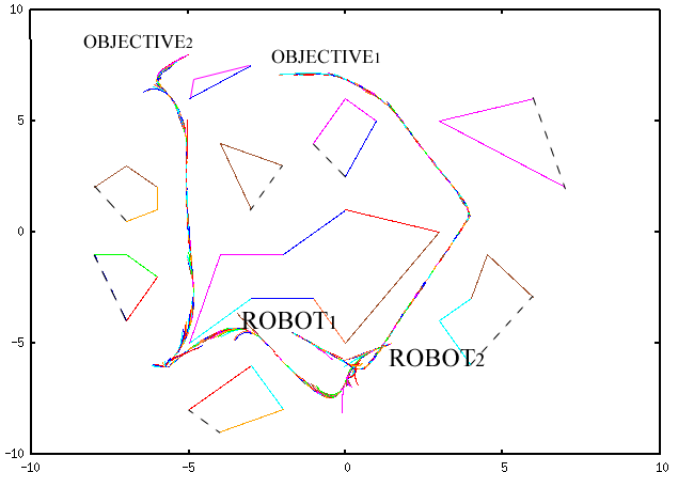


(b) With cooperation

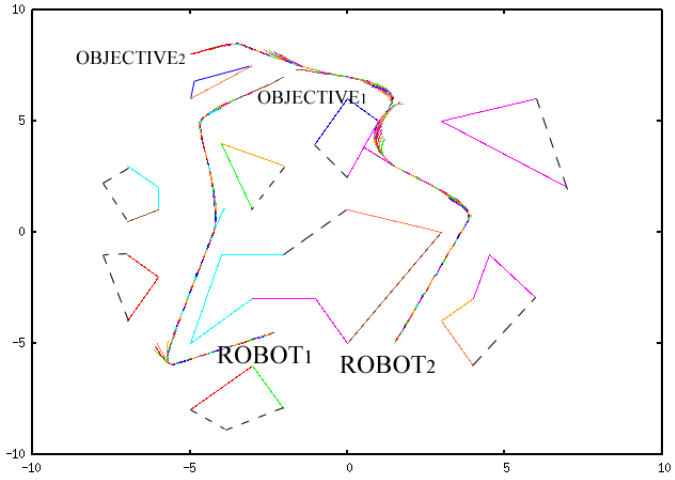
Figure 6.10: Path planning in simple environment

Table 6.2: comparison of Simulation result in complex environment

	Robot	Running Time (s)	Steps	Trajectory Length
Without Cooperation	R_1	8.74	127	20.04
	R_2	9.54	169	25.55
With Cooperation	R_1	7.285	116	18.15
	R_2	7.65	126	20.05



(a) Without cooperation



(b) With cooperation

Figure 6.11: Path planning in complex environment

6.6 Conclusion

This chapter presents a cooperative path planning algorithm for non-holonomic mobile robots based on visibility graphs. The algorithm proposes the enhancement of local path planning by sharing the information between robots, and the intermediate objectives for robots are generated by the visibility graph. To cope with the disadvantages of the visibility graph, an algorithm is proposed to generate polygons from a series of jointed segments and merge polygons when two polygons intercross. The robots can reach the target efficiently with the shared information while avoiding obstacles. The efficiency of the proposed algorithm are shown via different simulations.

Conclusions and Perspectives

The purpose of this chapter is to summarize the contributions presented in the thesis and introduce some perspectives for future work.

Conclusions

In this thesis, we studied the identification, path planning and motion control of non-holonomic mobile robots.

Real-time identification of robot kinematic models is discussed in Chapter 2, we consider the identification problem of the robot models as a detection of active mode of a special switched system, and each robot system is considered as a subsystem of the switched singular system. Then the distinguishability of the deduced switched singular system is discussed to give out the condition of the distinguishability. Moreover, the proposed method is quite robust to noises and disturbance in the system, one can see from the simulations that subsystems can be identified quickly in real time with noises.

Based on the identified robot kinematic model, Chapter 3 proposes a new path planning algorithm for non-holonomic robot in complex environment. In order to represent obstacles in a more accurate way, the viewed part of an obstacle is represented by a series of joint segments, and the algorithm generates intermediate objectives for the robot to avoid local minima in the environment. The reaching of intermediate objectives are ensured by optimal control problem and all the physical constraints are considered. The algorithm is based on flatness of the system, and can be extended to other types of robots stated in Chapter 2. Compared to visibility graph approach, the proposed algorithm reduces the computational complexity. The efficiency of the proposed algorithm is demonstrated via simulations and implementations in a wifibot.

It is well known that the open-loop control is not robust to the disturbance, thus in Chapter 4 we apply the i-PID controller, which is quite robust to the disturbance, to control the robot. In this chapter, the determination of the parameter α in the controller is discussed, and is selected as a switching function according to the information of the system. We show that with the proper parameter α, the controller can even stabilize the robot at a static point with the robot velocity being zero.

Because in some situations, there is no solution to the optimization algorithm used in Chapter 3, another path planning strategy is proposed in Chapter 5, which uses the potential field and i-PID controller. If there is no solution to the optimization algorithm, we can switch the planning strategy (as shown in Fig. 1.5(b)). The new potential field function defined in this chapter takes into account the robot orientation and angular velocity, and it is able to solve local minima problems and produce smooth repulsive force in complex environment to avoid oscillations. The virtual force generated by potential field function is used as the reference, the i-PID controller is used for robot motion control to achieve good performance.

Chapter 6 considers the cooperative path planning between robots. Robots can share their detected environment and their trajectories can be better planned. Visibility graph is used to generate intermediate objectives for each robot. In order to cope with the disadvantages of visibility graph, we propose an algorithm to generate expand polygons for the detected part of obstacles, and to merge boundaries of expanded obstacles after robots sharing their detected environment. The reaching of the intermediate objectives is ensured by either optimization algorithm proposed in Chapter 3 or the motion planning method proposed in Chapter 5, and a switching region is defined to judge whether to go to next intermediate objective or to replan the path.

Perspectives

At the end of this thesis, several issues remain unresolved, and some other methods can be developed. The theoretical concepts introduced in this thesis can lead to several extensions and future applications.

In Chapter 2, four robot kinematic models are discussed. Although as stated in Note 2.2.1, if the first four robot kinematic models can be identified, then the unidentified model is the fifth model (*Type* (1.2)), it is beneficial to study the *Type* (1.2) robot kinematic model and its input-output function and flat outputs. Therefore the path planning algorithm and the controller proposed in other chapters can be applied on the *Type* (1.2) robot.

In Chapter 3 and 4, the path planning algorithm and the controller are proposed for the (2.0) robot model. To be specific, in Chapter 3 all the constraints imposed on the robot are calculated as constraints of the flat output based on (2.0) robot model, and in Chapter 4 the parameter α in the controller (4.7) is selected according to the matrix $G(Y, \dot{Y})$ in the robot model (4.1). Therefore it is interesting to apply the proposed path planning algorithm and the i-PID controller to the robots of other types which are discussed in Chapter 2.

Chapter 5 proposes a path planning strategy using potential field functions, as we all know that the potential field approach can be used for lots of applications. For example, the new potential field strategy can be improved and used for robot formation control Barnes *et al.* (2007). The formation control is to control the robots moving cohesively in a specified geometrical pattern. The proposed potential field method in this thesis can be modified to keep robots with a desired distance and even a desired angle with other robots in the system, as well as avoid the obstacles in the environment. The method can also applied in applications like robot soccer games as in Vadakkepat *et al.* (2001), taking the ball as the target of attractive force and other robots as obstacles of repulsive forces.

In Chapter 6, a cooperative path planning method based on visibility graph is proposed for two mobile robots, and this method can be improved for more robots. In the proposed method, the polygon mergence algorithm is for two detected obstacles, and if there are more robots in the system, one can consider to merge two boundaries and then merge another obstacle boundary with the merged boundary. However this approach seems not very efficiency, it is beneficial to consider a more efficient polygon mergence algorithm of more than two robots.

In addition, in our study we focus on wheeled mobile robots, it will be interesting to try to apply the proposed approaches for other types of robots such as UAVs (Unmanned Aerial Vehicle). For example, with the flat output of the UAV

kinematic model Cowling *et al.* (2007), our path planning algorithm can be used to search for optimal collision free path.

Résumé en français

Motivation de la recherche

L'étude des robots mobiles a commencé à partir des années 1960, Nilsson et al. ont développé un robot autonome "Shakey" afin d'étudier l'intelligence artificielle, la planification autonome et le contrôle des systèmes robotiques. Dans les dernières décennies, les robots mobiles ont été largement étudiés, et a excité de plus en plus les intérêts des nombreuses recherches en raison de leurs applications dans l'industrie et leurs défis théoriques. Par exemple, la plate-forme robotique *SR4* sur Linux développé par la société *Smart Robots*, le robot mobile *Pioneer P3-DX* développé par *ActivMedia Robotics* pour la recherche et l'enseignement, la célèbre *Mars Rover Spirit* et etc. Plus récemment, les applications des robots mobiles sont devenus de plus en plus populaire, et les robots mobiles ont été utilisés dans les missions de sauvetage, les explorations, les guides touristiques, et même des divertissements tels que les jeux de football.

(a) Mars Rover (b) Jeux de football

Figure 1: Les applications des robots mobiles

Cependant, avec le développement rapide de l'automatisation et de la robotique, il y a des défis plus élevés pour les robots mobiles, et les demandes de la navigation autonome en environnement complexe deviennent importantes. C'est pourquoi nous nous concentrons sur la navigation autonome et le suivi des trajectoires des robots mobiles. L'objectif général de la recherche est de concevoir des nouveaux algorithmes de planification à naviguer robots dans l'environnement complexe et de proposer des contrôleurs robustes pour suivre la trajectoire désirée.

Identification des robots mobiles non-holonomes

La planification de trajectoire et le contrôle des robots mobiles sont les deux aspects principaux du problème de la navigation. Toutefois, pour presque toutes les approches du contrôle, les contrôleurs sont conçus sur la base du modèle de robot. En conséquence, les modèles des robots sont très importants dans les problèmes de la navigation, et les contrôleurs seront différents selon les différents modèles des robots (un exemple des différents types des robots est représenté sur la Fig. 2). Si nous avons un modèle de robot mobile inconnu, la première tâche est d'identifier le modèle, puis nous pouvons concevoir des contrôleurs pour le robot. Les modèles cinématiques des robots mobiles peuvent être classés en cinq catégories en introduisant les concepts de *degree of mobility* et de *degree of steerability*.

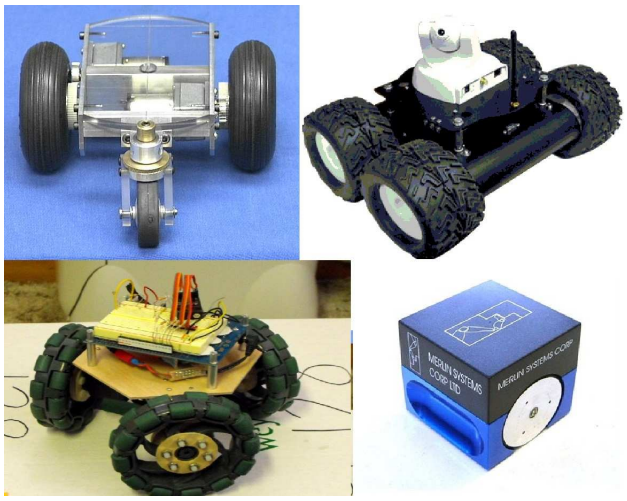


Figure 2: Différents types des robots

Pour ces différents modèles des robots mobiles non-holonomes, nous considérons le problème de l'identification comme une détection du sous-système d'un système à commutation spéciale. Si nous modélisons le sous-système comme un modèle possible des robots non-holonomes, alors le problème d'identification du modèle de robot devient une problème de l'identification des sous-systèmes de ce système à commutation.

Planification de trajectoire local pour les robots mobiles non-holonomes

Planification de trajectoire est très important, car il permet de la sélection et de l'identification d'une trajectoire possible pour les robots.

Lorsque nous considérons le problème de planification de trajectoire pour les robots mobiles non-holonomes, les contraintes physiques et les contraintes cinématiques doivent être pris en compte. Certains algorithmes considèrent les obstacles comme des cercles, ces algorithmes ne peuvent pas être utilisés dans un environnement complexe. Certains algorithmes considèrent l'environnement complexe, mais le "suivant la limite" mode qui n'est pas optimal est utilisé. Dans cette thèse, les obstacles irréguliers sont représentés par des segments. Le problème de planification de trajectoire pour des robots mobiles est décrit comme un problème de contrôle optimal. Minima locaux sont évités en choisissant des objectifs intermédiaires.

Contrôle de robots mobiles non-holonomes

Le problème de suivi de trajectoire pour un robot consiste à déterminer un contrôle à stabiliser asymptotiquement l'erreur de suivi. Selon le condition nécessaire de la Brockett, les systèmes des robots mobiles non-holonomes ne peuvent pas être asymptotiquement stabilisé par une rétroaction de l'Etat continue. Le contrôleur "i-PID", qui est proposé par *Fliess & Join*, présente la robustesse à des dynamiques non modélisés et des perturbations dans le système, et il a été étudié et appliqué à des nombreux procédés électriques et mécaniques. Nous appliquons le contrôleur "i-PID" pour les robots non-holonomes afin de contrôler les robots

avec les perturbations de mesure. Toutefois, en raison de la spécificité des systèmes non-holonomes, cet contrôleur ne peut pas être appliqué simplement, donc d'un paramètre de commutation est sélectionné et un contrôleur robuste est proposé de contrôle le robot avec les perturbations de mesure.

Planification de trajectoires pour les robots mobiles en utilisant de champs potentiels

Champ de potentiel est étudié largement en raison de la simplicité et de l'analyse mathématique intéressant. Bien que l'idée de champ de potentiel semble facile, parfois la méthode devient moins d'efficacité à couse des problèmes des minima locaux et des mouvements oscillatoires.

Dans cette thèse, nous proposons un nouveau champ de potentiel pour le robot planification de trajectoire pour résoudre les problèmes de minima locaux et les problèmes d'oscillation. La nouvelle fonction est définie en tenant compte de l'orientation du robot et de la vitesse angulaire, en plus de la position et de la vitesse linéaire, et les critères de distance est ajouté pour réduire les oscillations. La fonction proposée permet au robot d'éviter les minima locaux et de diminuer les oscillations dans le mouvement du robot. Enfin, le contrôleur i-PID est utilisé pour la planification de mouvement du robot. Cette méthode a seulement besoin de la mesure de la vitesse du robot et de l'information d'obstacles.

Planification de trajectoire coopérative pour les robots mobiles non-holonomes sur la base de graphe de visibilité

Des systèmes des multi-robots sont actuellement un intérêt principal de la recherche dans le domaine de la robotique. La planification de trajectoire coopérative a la possibilité de résoudre les problèmes plus efficacement.

Les méthodes de planification de trajectoire peuvent être divisés en deux catégories: la planification globale et la planification local. Cependant, tous les deux méthodes ont leurs inconvénients. Les informations complets de l'environnement

est nécessaire pour la planification de trajectoire global, mains ce n'est pas possible, parceque l'environnement est détectée partiellement par les robots. Dans la planification de trjectoire local, la trajectoire prévue ne peut pas être optimisée globalement, car l'environnement n'est que partiellement connue. En conséquence, la planification de trajectoire coopérative est considéré à la base de l'information partagée de chaque robot, et les robots sont capables d'atteindre l'objectif par réaliser les objectifs intermédiaires générés par graphe de visibilité. L'atteinte des objectifs intermédiaires est assurée par le problème de contrôle optimal.

Conclusions et perspectives

Dans cette thèse, nous avons étudié l'identification de modèles cinématiques de robot, la planification de trajectoire et le contrôle de mouvement des robots mobiles non-holonomes.

Contributions

Tout d'abord, l'identification des différents types de systèmes des robots mobiles non-holonomes est discutée. Avec le modèle cinématique identifiés, trois algorithmes de planification de trajectoire dans l'environnement complexe sont proposées pour un seul robot et plusieurs robots. Enfin, un contrôleur i-PID, qui est robuste aux perturbations de mesure et est capable de stabiliser le robot à un point statique, est appliqué pour contrôler les robots mobiles non-holonomes.

Perspectives

Les concepts théoriques dans cette thèse peuvent produire des extensions et des applications futures. Tout d'abord, le *type (2.0)* robot est discuté dans la planification de trajectoire et le contrôle, et il est intéressant d'appliquer les algorithmes proposés pour les autres types des robots mobiles non-holonomes. De plus, le champ de potentiel proposé peut être étendu à contrôler les robots en gardant un motif géométrique souhaité. En outre, il est intéressant d'essayer d'appliquer les approches proposées pour d'autres types de robots tels que UAVs (Unmanned Aerial Vehicle).

Bibliography

Abbott, E. & Powell, D. (1999). Land-vehicle navigation using gps. *Proceedings of the IEEE*, **87**, 145–162. 15

Agrachev, A.A. & Liberzon, D. (2001). Lie-algebraic stability criteria for switched systems. *SIAM J. Control Optim.*, **40**, 253–269. 26

Aguilar, L., Hamel, T. & Soueres, P. (1997). Robust path following control for wheeled robots via sliding mode techniques. In *Intelligent Robots and Systems, IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on*, vol. 3, 1389 –1395 vol.3. 20

Aho, A.V. & Hopcroft, J.E. (1974). *The Design and Analysis of Computer Algorithms*. Addison-Wesley Longman Publishing Co., Inc., 1st edn. 18

Al Khawaldah, M., Al-Adwan, I. & Eid, K. (2012). New exploration strategy for cooperative mobile robots. In *Proceedings of the 11th WSEAS international conference on Electronics, Hardware, Wireless and Optical Communications*, EHAC'12/ISPRA/NANOTECHNOLOGY'12, 149–154, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA. 123

Alt, H. & Welzl, E. (1988). Visibility graphs and obstacle-avoiding shortest paths. *Mathematical Methods of Operations Research*, **32**, 145–164. 17

Ardiyanto, I. (2010). Task oriented behavior-based state-adaptive pid (proportional integral derivative) control for low-cost mobile robot. In *Computer Engineering and Applications (ICCEA), 2010 Second International Conference on*, vol. 1, 103 –107. 25

Barnes, L., Fields, M. & Valavanis, K. (2007). Unmanned ground vehicle swarm formation control using potential fields. In *Control & Automation, 2007. MED'07. Mediterranean Conference on*, 1–8, IEEE. 143

Barshan, B. & Durrant-Whyte, H.F. (1995). Inertial navigation systems for mobile robots. *Robotics and Automation, IEEE Transactions on*, **11**, 328–342. 15

Bicchi, A., Casalino, G. & Santilli, C. (1996). Planning shortest bounded-curvature paths for a class of nonholonomic vehicles among obstacles. *Journal of Intelligent and Robotic Systems*, **16**, 387–405. 16

Bloch, A.M. (2003). *Nonholonomic mechanics and control*, vol. 24. Springer. 14, 20

Bloch, A.M., Reyhanoglu, M. & McClamroch, N.H. (1992). Control and stabilization of nonholonomic dynamic systems. *Automatic Control, IEEE Transactions on*, **37**, 1746–1757. 14

Borenstein, J., Everett, H. & Feng, L. (1996). Where am i? sensors and methods for mobile robot positioning. *University of Michigan*, **119**, 120. 15

Bourdais, R., Fliess, M., Join, C. & Perruquetti, W. (2007). Towards a model-free output tracking of switched nonlinear systems. In *NOLCOS 2007 - 7th IFAC Symposium on Nonlinear Control Systems*, Pretoria, South Africa. 26

Brockett, R.W. *et al.* (1983). *Asymptotic Stability and Feedback Stabilization*. Defense Technical Information Center. 14

Brooks, R. (1986). A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*, **2**, 14–23. 15

Burgard, W., Moors, M., Stachniss, C. & Schneider, F. (2005). Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, **21**, 376–386. 11

Camacho, D., Fernĺćndez, F. & Rodelgo, M.A. (2006). Roboskeleton: An architecture for coordinating robot soccer agents. *Eng. Appl. of AI*, **19**, 179–188. 12, 123

Campion, G., D'Andréa-Novel, B. & Bastin, G. (1991). Modelling and state feedback control of nonholonomic mechanical systems. In *Decision and Control, 1991., Proceedings of the 30th IEEE Conference on*, 1184 –1189 vol.2. 14, 19

Campion, G., Bastin, G. & D'Andréa-Novel, B. (1996). Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *Robotics and Automation, IEEE Transactions on*, **12**, 47–62. 26, 27, 49

Cardoso, P., Molina, L., Freire, E. & Carvalho, E. (2012). A methodology to designing strategies for robot soccer based on discrete event systems formalism. In *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*, 143 –149. 12

Charifa, S. & Bikdash, M. (2011). Mobile robot navigation using a combined optimized potential field and a boundary following algorithm. *Journal of Control Engineering and Technology*, **1**. 93

Chiddarwar, S.S. & Babu, N.R. (2011). Conflict free coordinated path planning for multiple robots using a dynamic path modification sequence. *Robotics and Autonomous Systems*, **59**, 508 – 518. 124

Choset, H. & Nagatani, K. (2001). Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *Robotics and Automation, IEEE Transactions on*, **17**, 125 –137. 18

Chung, Y. & Harashima, F. (2001). A position control differential drive wheeled mobile robot. *IEEE Transactions on Industrial Electronics*, **48**, 853–863. 19

Chwa, D. (2004). Sliding-mode tracking control of nonholonomic wheeled mobile robots in polar coordinates. *Control Systems Technology, IEEE Transactions on*, **12**, 637–644. 69

Conn, R. & Kam, M. (1998). Robot motion planning on n-dimensional star worlds among moving obstacles. *Robotics and Automation, IEEE Transactions on*, **14**, 320–325. 93

Conte, G., Perdon, A. & Moog, C.H. (1999). *Nonlinear Control Systems: An Algebraic Setting*. Lecture Notes in Control and I 243, Springer. 30

Conticelli, F., Bicchi, A. & Balestrino, A. (2000). Observability and nonlinear observers for mobile robot localization. In *In IFAC Int. Symp. on Robot Control, SyRoCo 2000.[5*, Citeseer. 15

Coron, J.M. (1992). Global asymptotic stabilization for controllable systems without drift. *Mathematics of Control, Signals and Systems*, **5**, 295–312. 69

Cowling, I.D., Yakimenko, O.A., Whidborne, J.F. & Cooke, A.K. (2007). A prototype of an autonomous controller for a quadrotor uav. In *European Control Conference*, 1–8. 144

D'Andréa-Novel, B., Campion, G. & Bastin, G. (1995). Control of nonholonomic wheeled mobile robots by state feedback linearization. *Int. J. Rob. Res.*, **14**, 543–559. 20

Datta, A., Ho, M. & Bhattacharyya, S. (2000). *Structure and Synthesis of PID Controllers*. Advances in Industrial Control, Springer. 71

De Boor, C. (2001). *A Practical Guide to Splines*. No. 27 in Applied Mathematical Sciences, Springer. 50

De Luca, A. & Oriolo, G. (1994). Local incremental planning for nonholonomic mobile robots. In *Robotics and Automation. IEEE International Conference on*, 104–110 vol.1. 105

De Persis, C., De Santis, R. & Morse, A. (2002). Nonlinear switched systems with state dependent dwell-time. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 4, 4419 – 4424 vol.4. 26

De Wit, C. & Sordalen, O. (1992). Exponential stabilization of mobile robots with nonholonomic constraints. *Automatic Control, IEEE Transactions on*, **37**, 1791 –1797. 19

de Wit, C., Siciliano, B. & Bastin, G. (1996). *Theory of robot control*. Communications and control engineering, Springer. 26, 27

DE WIT, C., SICILIANO, B. & BASTIN, G. (2001). *Theory of Robot Control*. Springer, Corrected edition. 49

DEFOORT, M., FLOQUET, T., KÖKÖSY, A. & PERRUQUETTI, W. (2006). Integral sliding mode control for trajectory tracking of a unicycle type mobile robot. *Integrated Computer-Aided Engineering*, **13**, 277–288. 14

DEFOORT, M., FLOQUET, T., KÖKÖSY, A. & PERRUQUETTI, W. (2008). Sliding-mode formation control for cooperative autonomous mobile robots. *Industrial Electronics, IEEE Transactions on*, **55**, 3944 –3953. 25

DEFOORT, M., PALOS, J., KÖKÖSY, A., FLOQUET, T. & PERRUQUETTI, W. (2009). Performance-based reactive navigation for non-holonomic mobile robots. *Robotica*, **27**, 281–290. 18, 25, 45, 46, 48, 50

DELAHAYE, D., PEYRONNE, C., MONGEAU, M. & PUECHMOREL, S. (2010). Aircraft conflict resolution by genetic algorithm and b-spline approximation. In *Proceedings of ENRI Int. Workshop on ATM/CNS. Tokyo, Japan*, vol. 4, 71–77. 50

DI MARCO, M., GARULLI, A., GIANNITRAPANI, A. & VICINO, A. (2004). A set theoretic approach to dynamic robot localization and mapping. *Autonomous robots*, **16**, 23–47. 16

DIJKSTRA, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, **1**, 269–271. 16

DISSANAYAKE, G., PAXMAN, J., MIRO, J.V., THANE, O. & THI, H.T. (2006). Robotics for urban search and rescue. In *Industrial and Information Systems, First International Conference on*, 294 –298. 11

DISSANAYAKE, M.G., NEWMAN, P., CLARK, S., DURRANT-WHYTE, H.F. & CSORBA, M. (2001). A solution to the simultaneous localization and map building (slam) problem. *Robotics and Automation, IEEE Transactions on*, **17**, 229–241. 16

DUBINS, L.E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, **79**, 497–516. 14

Dudek, G. & Jenkin, M. (2010). *Computational Principles of Mobile Robotics*. Computational Principles of Mobile Robotics, Cambridge University Press. 16

Durrant-Whyte, H. & Bailey, T. (2006). Simultaneous localization and mapping: part i. *Robotics & Automation Magazine, IEEE*, **13**, 99–110. 16

Fliess, M. & Join, C. (2008). Commande sans modèle et commande à modèle restreint. *e-STA*, **5**, 1–23. 20

Fliess, M. & Join, C. (2009). Model-free control and intelligent PID controllers: towards a possible trivialization of nonlinear control? In *15th IFAC Symposium on System Identification (SYSID 2009)*, IFAC, Saint-Malo, France. 20, 70

Fliess, M. & Sira-Ramírez, H. (2003). An algebraic framework for linear identification. *ESAIM: Control, Optimisation and Calculus of Variations*, **9**, 151–168. 36

Fliess, M., Lévine, J. & Rouchon, P. (1995). Flatness and defect of nonlinear systems: Introductory theory and examples. *International Journal of Control*, **61**, 1327–1361. 14, 19, 49

Fliess, M., Join, C., Mboup, M. & Sira-Ramírez, H. (2004). Compression différentielle de transitoires bruités. *Comptes Rendus Mathematique*, **339**, 821–826. 36

Fliess, M., Join, C. & Perruquetti, W. (2008a). Real-time estimation for switched linear systems. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, 941 –946. 33

Fliess, M., Join, C. & Sira-Ramirez, H. (2008b). Non-linear estimation is easy. *Int. J. Modelling Identification and Control*, **4**, 12–27. 36

Fliess, M., Join, C. & Riachy, S. (2011). Revisiting some practical issues in the implementation of model-free control. In *18th IFAC World Congress, IFAC WC'2011*, CDROM, Milan, Italie. 20, 70

Ge, S. (2010). *Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications*. Control Engineering, Taylor & Francis. 70

Ge, S. & Cui, Y. (2000). New potential functions for mobile robot path planning. *Robotics and Automation, IEEE Transactions on*, **16**, 615–620. 93

Ge, S. & Cui, Y. (2002). Dynamic motion planning for mobile robots using potential field method. *Autonomous Robots*, **13**, 207–222. 93, 95, 97, 104, 105, 107

Gédouin, P.A., Join, C., Delaleau, E., Bourgeot, J.M., Arbab-Chirani, S. & Calloch, S. (2008). Model-free control of shape memory alloys antagonistic actuators. In *17th IFAC World Congress*, CDROM, Seoul, Korea, Republic Of. 20

Glavaški, D., Volf, M. & Bonkovic, M. (2009). Robot motion planning using exact cell decomposition and potential field methods. In *Proceedings of the 9th WSEAS international conference on Simulation, modelling and optimization*, SMO'09, 126–131. 16

Goto, Y., , Goto, Y. & Stentz, A. (1987). Mobile robot navigation: The cmu system. *IEEE Expert*, **2**, 44–54. 18

Gourley, C. & Trivedi, M. (1994). Sensor based obstacle avoidance and mapping for fast mobile robots. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, 1306–1311, IEEE. 15

Graham, R.L. (1972). An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, **1**, 132–133. 51

Guo, Y. & Tang, T. (2008). Optimal trajectory generation for nonholonomic robots in dynamic environments. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2552 –2557. 18, 25, 50

Han, B.O., Kim, Y.H., Cho, K. & Yang, H. (2010). Museum tour guide robot with augmented reality. In *Virtual Systems and Multimedia (VSMM), 2010 16th International Conference on*, 223 –229. 12

Harinarayan, K. & Lumelsky, V. (1994). Sensor-based motion planning for multiple mobile robots in an uncertain environment. In *Intelligent Robots and*

Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on, vol. 3, 1485 –1492 vol.3. 124

Hart, P., Nilsson, N. & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, **4**, 100–107. 16

Hespanha, J.P., Liberzon, D. & Morse, A.S. (1999). Logic-based switching control of a nonholonomic system with parametric modeling uncertainty. *Systems & Control Letters*, **38**, 167–177. 69

Huang, H.P. & Chung, S.Y. (2004). Dynamic visibility graph for path planning. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, 2813 – 2818. 16

Ismail, A.T., Sheta, A. & Al-Weshah, M. (2008). A mobile robot path planning using genetic algorithm in static environment. *Journal of Computer Science*, **4**, 341–344. 16

Jang, G., Kim, S., Kim, J. & Kweon, I. (2005). Metric localization using a single artificial landmark for indoor mobile robots. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, 2857–2862, IEEE. 15

Jiang, Z.P. & Nijmeijer, H. (1999). A recursive technique for tracking control of nonholonomic systems in chained form. *Automatic Control, IEEE Transactions on*, **44**, 265–279. 69

Join, C., Robert, G. & Fliess, M. (2010). Model-free based water level control for hydroelectric power plants. In *IFAC Conference on Control Methodologies and Tecnologies for Energy Efficiency, CMTEE*, CDROM, IFAC, Vilamoura, Portugal. 20

Kamon, I., Rimon, E. & Rivlin, E. (1996). A new range-sensor based globally convergent navigation algorithm for mobile robots. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 1, 429–435 vol.1. 19, 45

KHATIB, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2, 500 – 505. 18

KIM, D.H. & KIM, J.H. (2003). A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer. *Robotics and Autonomous Systems*, **42**, 17 – 30. 94

KIM, J. (2004). *Soccer Robotics*. Springer Tracts in Advanced Robotics, Springer. 12, 123

KIM, R.S., CHOI, H., KIM, E. & CHANG-WOO, P. (2011). Mobile robot exploration via pseudo range model. In *Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference on*, 108 –110. 123

KÖKÖSY, A., DEFAUX, F.O. & PERRUQUETTI, W. (2008). Autonomous navigation of a nonholonomic mobile robot in a complex environment. In *Safety, Security and Rescue Robotics, 2008. SSRR 2008. IEEE International Workshop on*, 102 –108. 19, 25, 45, 55

KOLMANOVSKY, I. & MCCLAMROCH, N. (1995). Developments in nonholonomic control problems. *Control Systems, IEEE*, **15**, 20 –36. 11, 18

KOREN, Y. & BORENSTEIN, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, vol. 2, 1398–1404. 18

LATOMBE, J.C. (1991). *ROBOT MOTION PLANNING.: Edition en anglais*. The Springer International Series in Engineering and Computer Science, Springer. 18, 25, 93

LAUE, T. & RÖFER, T. (2005). Robocup 2004. chap. A behavior architecture for autonomous mobile robots based on potential fields, 122–133, Springer-Verlag. 94

LAUMOND, J. (1998). *Robot motion planning and control*. Lecture notes in control and information sciences, Springer. 11, 18

Lawrence, C., Zhou, J. & Tits, A. (94). User's guide for CFSQP version 2.5: AC code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. *Institute for Systems Research TR*, **94**, 16r1. 50

Lemaire, T., Lacroix, S. & Sola, J. (2005). A practical 3d bearing-only slam algorithm. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, 2449–2454, IEEE. 15

Leroy, S., Laumond, J.P. & Simeon, T. (1999). Multiple path coordination for mobile robots: A geometric algorithm. In *In Proc. of the International Joint Conference on Artificial Intelligence (IJCAI*, 1118–1123. 124

Liu, D., Fei, S., Hou, Z., Zhang, H. & Sun, C. (2007). *Advances in neural networks–ISNN 2007*. No. 2 in Lecture Notes in Computer Science, Springer. 20

Liu, D.Y., Gibaru, O. & Perruquetti, W. (2011). Error analysis of Jacobi derivative estimators for noisy signals. *Numerical Algorithms*, **58**, 53–83. 36, 37

Lumelsky, V.J. & Stepanov, A.A. (1987). Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *ALGORITHMICA*. 18

Maron, O. & Lozano-Pérez, T. (1996). Visible decomposition: Real-time path planning in large planar environments. *AI Memo*, **1638**. 125

Martinelli, A. & Siegwart, R. (2005). Observability analysis for mobile robot localization. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, 1471–1476, IEEE. 15

Mayne, D. & Michalska, H. (1990). Receding horizon control of nonlinear systems. *Automatic Control, IEEE Transactions on*, **35**, 814–824. 48

Mboup, M., Join, C. & Fliess, M. (2009). Numerical differentiation with annihilators in noisy environment. *Numerical Algorithms*, **50**, 439–467. 36, 37

Mehrjerdi, H. & Saad, M. (2010). Dynamic tracking control of mobile robot using exponential sliding mode. In *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*, 1517 –1521. 25

Morin, P. & Samson, C. (2004). Trajectory tracking for non-holonomic vehicles: overview and case study. In *Robot Motion and Control, 2004. RoMoCo'04. Proceedings of the Fourth International Workshop on*, 139–153, IEEE. 14

Moulay, E., Bourdais, R. & Perruquetti, W. (2007). Stabilization of nonlinear switched systems using control lyapunov functions. *Nonlinear Analysis: Hybrid Systems*, **1**, 482 – 490, proceedings of the International Conference on Hybrid Systems and Applications, Lafayette, LA, USA, May 2006: Part I. 26

Murphy, R., Kravitz, J., Stover, S. & Shoureshi, R. (2009). Mobile robots in mine rescue and recovery. *Robotics Automation Magazine, IEEE*, **16**, 91 –103. 11, 123

Nagatani, K., Kiribayashi, S., Okada, Y., Tadokoro, S., Nishimura, T., Yoshida, T., Koyanagi, E. & Hada, Y. (2011). Redesign of rescue mobile robot quince. In *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, 13 –18. 11, 123

Nearchou, A.C. (1998). Path planning of a mobile robot using genetic heuristics. *Robotica*, **16**, 575–588. 16

Nilsson, N.J. (1969). A mobile automation: an application of artificial intelligence techniques. In *Proceedings of the 1st international joint conference on Artificial intelligence*, IJCAI'69, 509–520, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 11

Nilsson, N.J. (1980). *Principles of Artificial Intelligence*. Tioga Publishing Company. 18

Normey-Rico, J.E., Alcalĺć, I., Gĺőmez-Ortega, J. & Camacho, E.F. (2001). Mobile robot path tracking using a robust pid controller. *Control Engineering Practice*, **9**, 1209 – 1214. 14, 19

Ó'Dúnlaing, C. & Yap, C.K. (1985). A "retraction" method for planning the motion of a disc. *Journal of Algorithms*, **6**, 104–111. 16

OELEN, W. & VAN, J.A. (1994). Robust tracking control of two-degrees-of-freedom mobile robots. *Control Engineering Practice*, **2**, 333–340. 19

O'KANE, J.M. (2006). Global localization using odometry. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, 37–42, IEEE. 15

PARK, K., CHUNG, H. & LEE, J.G. (2000). Point stabilization of mobile robots via state-space exact feedback linearization. *Robotics and Computer-Integrated Manufacturing*, **16**, 353–363. 14

PEARS, N.E. (2001). Mobile robot tracking of pre-planned paths. *Advanced Robotics*, **15**, 97–107. 19

PEARSALL, J. (2001). *Concise Oxford Dictionary*. Oxford University Press. Tenth Edition, Rivised ed. 12

PERRUQUETTI, W. & BARBOT, J.P. (2002). *Sliding mode control in engineering*. CRC Press. 20

POMET, J.B. (1992). Explicit design of time-varying stabilizing control laws for a class of controllable systems without drift. *Systems & Control Letters*, **18**, 147–158. 69

QI, H. & MOORE, J.B. (2002). Direct kalman filtering approach for gps/ins integration. *Aerospace and Electronic Systems, IEEE Transactions on*, **38**, 687–693. 15

RASHID, M. & SIDEK, S. (2011). Dynamic modeling and verification of unicycle mobile robot system. In *Mechatronics (ICOM), 2011 4th International Conference On*, 1–5. 73

REDMILL, K.A., KITAJIMA, T. & OZGUNER, U. (2001). Dgps/ins integrated positioning for control of automated vehicle. In *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, 172–178, IEEE. 15

REEDS, J. & SHEPP, L. (1990). Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, **145**, 367–393. 14

Rezaee, H. & Abdollahi, F. (2012). Adaptive artificial potential field approach for obstacle avoidance of unmanned aircrafts. In *Advanced Intelligent Mechatronics, 2012 IEEE/ASME International Conference on*, 1–6. 93

Riachy, S., Fliess, M. & Join, C. (2011). High-order sliding modes and intelligent PID controllers: First steps toward a practical comparison. In *18th IFAC World Congress, IFAC WC'2011*, CDROM, Milan, Italy. 20, 70

Rooker, M.N. & Birk, A. (2007). Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, **15**, 435 – 445. 11

Ryu, J.C. & Agrawal, S.K. (2008). Differential flatness-based robust control of a two-wheeled mobile robot in the presence of slip. *ASME Conference Proceedings*, **2008**, 915–921. 19

Salichs, M. & Moreno, L. (2000). Navigation of mobile robots: open questions. *Robotica*, **18**, 227–234. 25

Samson, C. (1991). Velocity and torque feedback control of a nonholonomic cart. In *Advanced robot control*, 125–151, Springer. 69

Samson, C. (1995). Control of chained systems application to path following and time-varying point-stabilization of mobile robots. *Automatic Control, IEEE Transactions on*, **40**, 64–77. 69

Samson, C. & Ait-Abderrahim, K. (1991). Feedback control of a nonholonomic wheeled cart in cartesian space. In *Robotics and Automation, Proceedings., 1991 IEEE International Conference on*, 1136 –1141 vol.2. 14, 19

Schwartz, J.T. & Sharir, M. (1983). On the ąřpiano movers'ąś problem i. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, **36**, 345–398. 16

Sert, H., Kökösy, A. & Perruquetti, W. (2011). A single landmark based localization algorithm for non-holonomic mobile robots. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 293–298, IEEE. 15

Sfeir, J., Saad, M. & Saliah-Hassane, H. (2011). An improved artificial potential field approach to real-time mobile robot path planning in an unknown environment. In *Robotic and Sensors Environments, IEEE International Symposium on*, 208–213. 93

Sillito, R.R. & Fisher, R.B. (2009). Parametric trajectory representations for behaviour classification. In *British Machine Vision Conference, BMVC 2009, London, UK, September 7-10, 2009. Proceedings*, British Machine Vision Association. 50

Sira-Ramirez, H. & Fliess, M. (2006). An algebraic state estimation approach for the recovery of chaotically encrypted messages. *International Journal of Bifurcation and Chaos*, **16**, 295–309. 36

Sira-Ramírez, H., Cortés-Romero, J. & Luviano-Juárez, A. (2011). *Robust Linear Control of Nonlinear Flat Systems, Robust Control, Theory and Applications*. InTech. 74

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, 3310–3317. 18

Sun, Z., Ge, S.S. & Lee, T.H. (2002). Controllability and reachability criteria for switched linear systems. *Automatica*, 200–2. 26

Tayebi, A. & Rachid, A. (1996). Path following control law for an industrial mobile robot. In *Control Applications, 1996., Proceedings of the 1996 IEEE International Conference on*, 703 –707. 20

Tomatis, N., Philippsen, R., Jensen, B., Arras, K., Terrien, G., Piguet, R. & Siegwart, R. (2002). Building a Fully Autonomous Tour Guide Robot: Where Academic Research Meets Industry. 12

Vadakkepat, P., Lee, T.H. & Xin, L. (2001). Application of evolutionary artificial potential field in robot soccer system. In *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, 2781–2785, IEEE. 143

VIÉVILLE, T. & FAUGERAS, O.D. (1990). Cooperation of the inertial and visual systems. In *Traditional and non-traditional robotic sensors*, 339–350, Springer. 15

VILLAGRA, J. & BALAGUER, C. (2010). Robust motion control for humanoid robot flexible joints. In *Control Automation (MED), 2010 18th Mediterranean Conference on*, 963 –968. 20, 70

VU, L. & LIBERZON, D. (2005). Common lyapunov functions for families of commuting nonlinear systems. *Systems and Control Letters*, **54**, 405–416. 26

WAN, J. & CHEN, P. (2008). Analysis on nonlinear feedback controls for differential mobile robots and its application to multi-robot formation control - part one. In *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, 1224 –1229. 25

WARREN, C. (1990). Multiple robot path coordination using artificial potential fields. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, 500 –505 vol.1. 124

WEISBIN, C. & RODRIGUEZ, G. (2000). Nasa robotics research for planetary surface exploration. *Robotics Automation Magazine, IEEE*, **7**, 25 –34. 12

WILLIAMS, S.B. (2001). *Efficient solutions to autonomous mapping and navigation problems*. Ph.D. thesis, The University of Sydney. 16

WULF, O., ARRAS, K.O., CHRISTENSEN, H.I. & WAGNER, B. (2004). 2d mapping of cluttered indoor environments by means of 3d perception. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 4, 4204–4209, IEEE. 16

XIE, G., ZHENG, D. & WANG, L. (2002). Controllability of switched linear systems. *Automatic Control, IEEE Transactions on*, **47**, 1401 – 1405. 26

YANG, J.M. & KIM, J.H. (1999). Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots. *Robotics and Automation, IEEE Transactions on*, **15**, 578 –587. 20

Yang, X., He, K., Guo, M. & Zhang, B. (1998). An intelligent predictive control approach to path tracking problem of autonomous mobile robot. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, vol. 4, 3301 –3306 vol.4. 20

Youcef-Toumi, K. & Wu, S. (1991). Input/output linearization using time delay control. In *American Control Conference, 1991*, IEEE. 74

Zhang, B., Chen, W. & Fei, M. (2006). An optimized method for path planning based on artificial potential field. In *Intelligent Systems Design and Applications*, vol. 3, 35–39. 94

List of publications

Papers published

1. Ma, Y., Zheng, G., Perruquetti, W. and Qiu. Z: **Local path planning for mobile robots based on intermediate objectives**. To appear in *Robotica 2014*.

2. Ma, Y., Zheng, G., Perruquetti, W. and Qiu Z.: **Control of non-holonomic wheeled mobile robots via i-pid controller**. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 4413-4418.

3. Ma, Y., Zheng, G. and Perruquetti, W.: **Real-time identification of different types of non-holonomic mobile robots**. In 9th IFAC Symposium on Nonlinear Control Systems (NOLCOS), pp 791-796.

4. Ma, Y., Zheng, G. and Perruquetti, W.: **Real-time local path planning for mobile robots**. In 2013 9th Workshop on Robot Motion and Control (RoMoCo), pp 215-220.

5. Ma, Y., Zheng, G. and Perruquetti, W.: **Cooperative path planning for mobile robots based on visibility graph**. In 2013 32nd Chinese Chinese Control Conference (CCC), pp 4915-4920.

Papers submitted

1. Ma, Y., Zheng, G., Perruquetti, W. and Qiu Z.: **i-PID controller determination for non-holonomic mobile robots**. Submitted to *Robotica*.

2. Ma, Y., Zheng, G., Perruquetti, W. and Qiu Z.: **Non-holonomic mobile robot motion planning via a modified potential field function and the *i*-PID controller**. Submitted to *Journal of Intelligent and Robotic Systems*.

3. Ma, Y., Zheng, G., Perruquetti, W. and Qiu Z.: **Motion planning for mobile robots using potential field and the *i*-PID controller**. Submitted to *IROS 2014*.

Path planning and control of non-holonomic mobile robots

Abstract

This thesis is dedicated to the path planning and control strategy for non-holonomic mobile robots.

Firstly, the identification of different mobile robot kinematic models is discussed: robot kinematic models are formulated as a switched singular nonlinear system, and the problem becomes the real-time identification of the switching signal.

Secondly, based on the identified robot model, a local path planning algorithm is proposed, in which the irregular contour of obstacles is represented by line segments. The path planning problem is formulated as a constrained receding horizon planning problem in which the trajectory is obtained by solving an optimal control problem with constraints.

Thirdly, we apply an i-PID controller to control the non-holonomic mobile robot with measurement disturbance. A switching parameter α is proposed because of the particularity of the non-holonomic system.

In addition to our proposed path planning algorithm, another path planning approach using potential field is proposed. The modified potential field function, which takes into account orientation and angular velocity of the robot, is able to solve local minima problems and produce smooth forces to avoid oscillations.

Finally, a cooperative path planning approach between robots is proposed by using the shared local information of each robot. In this approach, an algorithm for expanding obstacles and merging intersecting obstacles is proposed. The visibility graph is used for each robot in order to generate a series of intermediate objectives which guide the robot to its final target.

Keywords: Non-holonomic mobile robot, Path planning, Motion control, Potential field function, Cooperative path planning

Planification de trajectoire et commande pour les robots mobiles non-holonomes

Résumé

Ce travail vise à proposer de nouvelles stratégies pour la planification et la commande des robots mobiles non-holonomes.

D'abord, la problématique d'identification des différents modèles cinématiques de robot mobiles est discutée. Le modèle cinématique du robot est formulé comme un système singulier non-linéaire et à commutation, donc l'identification de modèle est simplifiée à l'identification en temps réel du signal de commutation.

Ensuite, sur la base du modèle identifié du robot, un algorithme de planification locale est proposé, dans lequel le contour irrégulier de l' obstacle est représenté par des segments. La trajectoire est obtenue en résolvant un problème de commande optimale avec contraintes.

Puis nous appliquons un contrôleur i-PID pour contrôler le robot mobile non-holonome avec la perturbation dans les mesures. Un paramètre de commutation α est proposé en raison de la particularité du système non-holonome.

En plus de notre algorithme de planification proposé, une autre approche de planification en utilisant les champs de potentiels est proposée. La nouvelle fonction de champ de potentiel est en mesure de résoudre les problèmes de minima locaux et de produire des forces lisses pour éviter les oscillations.

Enfin, une approche de planification coopérative entre robots est proposée en utilisant les informations locales partagées par chaque robot. Un algorithme pour étendre les obstacles et fusionner les obstacles qui s'entrecroisent est proposé. Le graphe de visibilité est utilisé pour chaque robot afin de générer une série d'objectifs intermédiaires qui pourront le guider vers son objectif final.

Mots-clés: Robots mobiles non-holonomes, Planification, La commande du mouvement, Fonction potentiel, Planification coopérative.