Motion Planning and Robust Control for Nonholonomic Mobile Robots under Uncertainties

Amnart Kanarat

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Mechanical Engineering

May 10, 2004
Blacksburg, Virginia

Keywords: Motion Planning, Robust Control, Mobile Robot, Uncertainty, Optimization, CUF

Motion Planning and Robust Control for Nonholonomic Mobile Robots under Uncertainties

Amnart Kanarat

(ABSTRACT)

This dissertation addresses the problem of motion planning and control for nonholonomic mobile robots, particularly wheeled and tracked mobile robots, working in extreme environments, for example, desert, forest, and mine. In such environments, the mobile robots are highly subject to external disturbances (e.g., slippery terrain, dusty air, etc.), which essentially introduce uncertainties to the robot systems. The complexity of the motion planning problem is due to taking both nonholonomic and uncertainty constraints into account simultaneously. As a result, none of the conventional nonholonomic motion planning can be directly applied. The control problem is even more challenging since state constraints posed by obstacles in the environments must also be considered along with the nonholonomic and uncertainty constraints.

In this research, we systematically develop a new type of motion planning technique that determines an optimal path for a mobile robot in a given environment. This motion planning technique is based on the idea of a maximum allowable uncertainty, which is a number assigned to each free configuration in the environment. The optimal path is a path connecting given initial and goal configurations through a series of configurations respecting the nonholonomic constraint and possessing the highest maximum allowable uncertainty. Both linear and quadratic approximations of the maximum allowable uncertainty, including their corresponding motion planners, have been studied. Additionally, we develop the first real-time robust control algorithm for the mobile robot under uncertainty to follow given paths safely and accurately in cluttered environments. The control algorithm also utilizes the concept of the maximum allowable uncertainty as well as the robust control theory. The simulation results have shown the effectiveness and robustness of the control algorithm in steering the mobile robot along a given path amidst obstacles without collisions even when the level of robot uncertainty is high.

Dedication

To my late grandparents

Acknowledgements

I would like to express my gratitude and appreciation to my advisor, Dr. Robert Sturges, for his continuing support and guidance throughout my years at Virginia Tech. To me, he has been both a wonderful teacher and a dear friend. His remarkable mind and admirable qualities have always inspired me, and always remind me of how a great professor should be.

I would also like to thank the dissertation committee members, Dr. Mehdi Ahmadian, Dr. Charles Reinholtz, Dr. Donald Leo, and Dr. Craig Woolsey, for their constructive advices and efforts in reading and commenting on my dissertation. Special thanks to Dr. Eugene Cliff for reinforcing my background in optimal control theory and providing parts of source codes used in this work.

Thanks to all my office mates, including Roger, Munki, and Choi for being such good friends, I will miss you all. Thanks to all Thai students at Virginia Tech for making my time here full of fun, see you all back home. Further thanks go to my parents and family for their incessant love and care, I could not make it this far without them. Finally, I would like to thank my dearest, Natalie (Nok), who always be so patient and supportive through all the tough time.

Contents

List of Figures

List of Tables

Nomenclature

$\lVert \cdot \rVert$	Euclidean norm (2-norm) of vector or its induced matrix norm (2-norm)
$\lVert \cdot \rVert_\infty$	Infinity-norm
$\forall$	for all
$\exists$	there exists
$\in$	belongs to, or in the set
$\subset$	subset of
$\subseteq$	subset of or equal to
$\to$	tends to
$\Rightarrow$	implies that
$\sum$	summation
max	maximum
min	minimum
R^n	the *n*-dimensional Euclidean space
R^+	the set containing all nonnegative real numbers
A^c	the complement of set A in R^n
x^T	the transpose of vector x
$f : S_1 \to S_2$	a function f mapping a set S_1 into a set S_2

Chapter 1

Introduction

The chapter begins with the discussion of the lack of efficient motion planning and control methods for nonholonomic mobile robots operating in hostile environments. The pressing need for efficient motion planning methods is apparent in the present real-world applications. Literature reviews have evidently confirmed such statements, and the research objectives are established accordingly. The overview of the dissertation is given at the end to serve as a roadmap of this dissertation and a quick reference for readers.

1.1 Motivation

Nowadays, more and more autonomous mobile robots are being developed and deployed in many real-world applications, for example, factory automation, underground mining, military surveillance, and even space exploration. In those applications, the mobile robots often work in unknown and inhospitable environments. To survive, these robots must be able to constantly monitor and appropriately react to variation and uncertainty in their environments. A basic task that every autonomous mobile robot must perform is safely navigating itself from one point to another within its environment. Two key ingredients that help the mobile robot to accomplish this navigation task are motion planning and control. The motion planning deals with the ability of a mobile robot to plan its own motion in its working space, and the motion control concerns with the ability of a mobile robot to follow or track a planned motion. Clearly, as mobile robot applications become more complex, the needs for more efficient and reliable motion planning and control are indispensable.

On the basis of the application domain, mobile robots can be categorized into four classes: Terrestrial, Aquatic, Airborne, and Space robots [Dudek and Jenkin, 2000]. Out of these four classes, the terrestrial robots are the largest class, and can often be found in many industrial applications. Although there are various types of locomotion for the terrestrial robots, the most popular terrestrial robots are wheeled and tracked due to their practicality and simplicity. As a tradeoff, these wheeled and tracked robots—which are of our major interest—have been proven to be difficult to control because the ground contact imposes a kinematic constraint called "Nonholonomic Constraint". This constraint prevents the mobile robot from locally moves sideway along the direction of the wheel axis and, consequently, complicates the control design since the robot is globally controllable but not locally controllable.

As mentioned earlier, mobile robots work in a wide range of working conditions. The working conditions range from ideal conditions such as a factory floor to harsh conditions such as an underground mine floor. In harsh conditions, the robots are more susceptible to disturbances from the environments. Uneven terrain, muddy ground, dusty air, humidity, etc., can introduce disturbances into the sensing and control systems of the mobile robots. Besides, the robots also face disturbances from within the robot systems themselves such as friction and backlash in robot drive systems. These disturbances result in the presence of uncertainties in the mobile robots. To cope with these uncertainties effectively, one needs to identify and take into account the effect of the uncertainties in the control design for the mobile robots. The effect of the uncertainties to the success of the robot mission becomes even more evident when the mobile robot works in cluttered environment, where the size of the robot is comparable to the size of its free working space. For the robot to navigate safely through such tight space under uncertainties, the motion planning and control schemes must take advantage of the information about the environment geometry as well as the robot geometry to decrease the effect of uncertainties.

From the above paragraphs, it is apparent that better, or perhaps the best, motion planning and control schemes for nonholonomic mobile robots are needed to ensure safe navigation of the robots amidst cluttered environments. This is crucial especially when the uncertainties (both internal and external) are taken into account and when the geometry of the robots is also considered.

1.2 Literature Review

This section is divided into two parts. The first and the second parts present the literature review in robot motion planning and robot motion control, respectively. At the end of each part, we summarize the prior arts, and identify their shortcomings.

Robot Motion Planning. One important characteristic of autonomous robots is the ability to plan their own motions in working spaces without human intervention. This ability is referred to as "Motion Planning". The most classical problem in robot motion planning is often referred to as the "Piano Mover" problem. The problem is about moving a piano from one room to another. An environment or workspace is assumed to be known, cluttered, and static. The piano can be freely translated and rotated on a plane. This type of problem is easily solved by means of geometric planners when reduced to a configuration space model [Latombe, 1991], where the piano can be represented by just a point while each obstacle in the workspace is represented by configuration obstacles in the configuration space. In this way, the motion planning problem is reduced to finding a continuous path connecting between start and goal points in the collision-free portion of the configuration space.

The concept of configuration space is well known and widely accepted among roboticists. It becomes a basic mathematical tool in solving motion planning problems, and finds its place in many motion planning applications. Its applications, however, are mostly limited to holonomic systems such as robot manipulators, which represent more than half of the existing robotics systems. In a holonomic system, a set of independent generalized coordinates of the system can be found, and thus allows any arbitrary motion in the workspace subject to constraints due to clutter. A nonholonomic system, on the other hand, does not have a set of independent generalized coordinates; therefore, only motions satisfying nonholonomic constraints are feasible. This makes the problem of motion planning for a nonholonomic system is more difficult than for a holonomic system because an arbitrary collision-free path in the configuration space does not necessarily correspond to a feasible trajectory for the nonholonomic system [Kolmanovsky and McClamroch, 1995]. As a result, the configuration space concept alone (or similar geometric planners such as cell-decomposition and Voronoi diagram methods [Latombe, 1991]) cannot solve the problem of nonholonomic motion planning.

Motion planning for nonholonomic systems has been an area of active research for more than a decade. Among nonholonomic systems, terrain mobile robots receive the most attention from many researchers due to their wide range of applications. A large number of motion planning problems for nonholonomic systems in this class (e.g., wheeled mobile robots, car-like robots, and tractors with trailers) have been solved by various approaches. Optimal control approaches, based on Pontryagin's Maximum Principle, have been applied to find optimal controls that minimize some objective functions (e.g., time, distance, expended energy, and etc.) subject to some constraints such as actuator's limitation (e.g. [Fernandes et al., 1991] [Renaud and Fourquet, 1997] [Alexander et al., 1998] [Oriolo et al., 2000] [Cameron and Book, 1994] [Kondak and Hommel, 2001]). The results are minimum-time and shortest paths that connect between the initial state and the goal state while satisfying the constraints. However, the optimal control for a minimum-time problem mostly leads to bang-bang control, and the shortest path always forces a robot to move close to obstacles. These outcomes are undesirable in real applications where the lifetimes of robot actuators and the likelihood of collisions are of major concerns.

Nonlinear control techniques include differential-geometric and differential-algebraic techniques (e.g. [Sekhavat and Laumond, 1998] [Lafferriere and Sussmann, 1993] [Sussmann, 1992]), geometric phase method [Reyhanoglu et al., 1993], and control parameterization approach [Murray and Sastry, 1993]. All of these techniques mainly focus in determining open loop controls, which steer a nonholonomic system from an initial state to a final state over finite time. Only few of them take into account the obstacle-avoidance issue. Another approach is a canonical path (also known as "motion primitives") method. By choosing a family of paths and concatenating them together in specific order, desired motions can be produced. For example, we might consider paths for a car-like robot that cause a net rotation of any angle, or a translation in the direction that the robot is facing. We can then move the car to any configuration by reorienting the car, driving forward, and again reorienting the car [Murray et al., 1993]. The set of canonical paths used for a given problem is usually specific to that problem, and may be derived by variety of approaches such as optimal control and geometric nonlinear control.

As previously stated, most of literature discussed above concentrates on generating open loop controls to steer the states of the nonholonomic systems, in unobstructed workspace, to the desired states within a given period of time. Although these studies cannot solve the nonholonomic motion planning with obstacle-avoidance directly, they serve as a powerful mathematical foundation for later motion planning with obstacle-avoidance development. Gurvits and Li [Gurvits and Li, 1993], Laumond et al. [Laumond et al., 1994], and Laumond [Laumond, 1995] applied the results from optimal control and nonlinear control techniques in developing motion planners for nonholonomic systems in the presence of obstacles. The planners first compute a collision-free path connecting initial and final states without taking into account the nonholonomic constraints. After that, this path is approximated by a sequence of feasible paths such as shortest paths generated by previous steering methods and then smoothed.

One of the well-known approaches to nonholonomic motion planning in the presence of obstacles has been developed by Barraquand and Latombe [Barraquand and Latombe, 1991] and applied to the tractor-trailer system. It consists of a search in a discretized configuration space. At each iteration, the planner expands six successors corresponding to discretized set of controls starting from an initial state. The expansions terminate when a final state is reached. During the expansion, an A* (read "A-Star") algorithm is used to search the graph whose nodes are the expanded successors. The result is the path that minimizes a prescribed cost function. This approach has been adopted and modified by many researchers (e.g. [Podsedkowski, 1998] [Podsedkowski et al., 2001]).

Mirtich and Canny [Mirtich and Canny, 1992] have proposed a unique approach in planning a collision-free path for a car-like robot moving among obstacles. The planner is based on building one-dimensional maximal clearance skeletons through the configuration space of the robot and then computing a feasible path by moving the robot loosely along these skeletons through sequence of optimal paths from initial to final states. The boundary of each skeleton indicates the maximum distance that the robot can move before colliding with obstacles; in other words, the robot can move freely without collision as long as it stays within the skeleton boundary. Since the skeleton stays maximally clear from the obstacles, the resulting path tends to be of low complexity.

Many other approaches in solving the motion planning problem of nonholonomic systems with obstacle-avoidance have also been proposed, for example, curve fitting approaches (e.g. [Wang, 1996] [Xu et al., 1999] [Papadopoulos and Poulakakis, 2001]), which convert the motion planning problem into a curve fitting problem satisfying given end-point conditions, and path space approach [Divelbiss and Wen, 1997], which transforms the motion planning problem into a nonlinear least square problem. However, none of the approaches to nonholonomic motion planning mentioned in previous paragraphs take into account the effect of uncertainties, which more or less happen in real applications.

In the past decade, a group of researchers have incorporated the effect of uncertainties into the motion planning problem for both holonomic and nonholonomic systems. For holonomic systems, many approaches, for example, preimage backchaining method [Latombe, 1991], landmark-based approach [Lazanas and Latombe, 1995], sensory uncertainty field technique [Takeda et al., 1994], and path-space algorithm [Page and Sanderson, 1995], have been proposed and applied to holonomic motion planning. For nonholonomic systems, among the pioneer researches dealing with uncertainty in nonholonomic motion planning is Jacobs and Canny's work [Jacobs and Canny, 1993]. They developed the concept of δ-robustness as a means to measure the degree of robustness of a planned path. The number δ directly relates to the maximum error allowed in the orientation of a robot when it begins to travel along the planned path. However, their work was only limited to wheeled mobile robots that have minimum turning radius such as car-like robots.

Pruski and Rohmer [Pruski and Rohmer, 1997] proposed a motion planning method for a differentially driven-wheeled robot subject to control uncertainty. The planner runs in two steps. First, the motion planning as described in [Barraquand and Latombe, 1991] determines a collision-free path from given initial to final configurations without taking control uncertainty into account. To deal with the uncertainty, each segment of the path is then expanded to form tracks of the equiprobability ellipsoids. The collision verification is made throughout the expanded path, and the path is iteratively modified wherever the collision occurs. The outcome is the robust path that a robot with a given bounded control disturbance can traverse safely. Similar motion planner has been proposed in Hait, et al. [Hait et al., 1999], where the authors combined together the nonholonomic motion planning presented by Barraquand and Latombe [Barraquand and Latombe, 1991] and the idea of landmark-based approach by Lazanas and Latombe [Lazanas and Latombe, 1995].

Moon et al. [Moon et al., 1999] proposed a motion planner for a car-like mobile robot subject to both control and sensory uncertainties. Based on a given map, the planner first generates a path off-line from initial to goal configurations by the configuration space method. This path consists of series of straight lines and arcs, and serves as a targeted path that the robot will follow. While the robot is traveling along the path, an on-line viewpoint and the motion planner will determine the safe viewpoint and motion that the robot should execute to eliminate the likelihood of collision when the robot is subject to the control uncertainty. The control uncertainty model is introduced to the problem by means of positional uncertainty ellipsoid and directly used for determination of the viewpoint and motion for the robot. The sensory uncertainty model, on the other hand, is solely used for selecting the best landmark that the robot should sense in each location on the given map.

The application of probability theory was introduced when Timcenko and Allen [Timcenko and Allen, 1994] proposed a motion planning method, based on probability, for mobile robots in the presence of sensory and control uncertainties. A 2-D workspace is represented in the form of an occupancy grid where each grid contains a value of probability that a given grid is occupied by an obstacle (sensory uncertainty). The control uncertainty is modeled by a linear stochastic differential equation, which in turn is used to calculate a probability that the robot will be in free workspace at any given grid. An A* search algorithm is then employed to conduct the search from given start and goal grids for the best possible path.

Another interesting work by Fraichard and Mermond [Fraichard and Mermond, 1998] presents the first motion planner taking into account both nonholonomic and uncertainty constraints simultaneously for a car-like robot. The planner uses two-phase algorithmic structure, which is basically a collaboration of global and local path planners. The global path planner keeps generating a graph by selecting new node in configuration space, where each new node generated is connected to the existing graph by a path generated by the local path planner such as the shortest path for Reeds & Shepp's car [Reeds and Shepp, 1990] or Dubins' car [Dubins, 1957]. They propose a local path planner that takes into account configuration uncertainty due to error in sensing and propagation of the configuration uncertainty over the traversing path due to control uncertainty. The propagation of the configuration uncertainty is limited by the use of landmark concept. The planner guarantees no collision for given uncertainty bounds in sensing and control. However, the path obtained by this planner is conservative since the shape of configuration uncertainty is coarsely approximated by a circle.

Recently, inspiring works by Lambert, et al. [Lambert et al., 1998], Lambert and Le Fort-Piat [Lambert and Le Fort-Piat, 1999 and 2000], and Hamel et al. [Hamel et al., 2001] have shed some light for a new direction in developing a motion planning for mobile robots in the presence of bounded uncertainties. By introducing the concept of a security margin, the collision along a planned path can be predicted beforehand, and, as a result, can be prevented by modifying the existing path during the planning phase. The security margin can be thought of as a boundary of a series of perturbed planned paths. As long as the robot can traverse along this boundary without any collision, the planned path is considered safe and robust. This margin can be obtained by determining a bounded attractive domain containing the configuration error of the robot as the prescribed closed-loop control is performed around the planned path. However, the authors only used the concept of security margin as a means to check robustness of a pre-planned path generated by an A* search method but not to plan a path.

The literature review shows that only few of the previous works explicitly take into account both control and sensing uncertainties in nonholonomic motion planning. None of the previous works studies how the degree of uncertainties affect the shape of a planned path in a given environment. In particular, the previous works fail to provide answers to the following questions: how much uncertainty is allowed for a mobile robot to navigate safely through a given environment on a given path, what the best possible path is for a given environment, and what a safe fastest speed is for a mobile robot to follow a given path at given level of uncertainty. The answers to these questions are crucial, especially, when a mobile robot works in a tight cluttered space, where the room for error is extremely limited. Moreover, the previous efforts—in nonholonomic motion planning under uncertainties—determine only a possible solution, while the optimal solution has yet to be investigated.

To address the shortcoming of the existing art, a new motion planning strategy is needed, and the relationship between the uncertainties and the nonholonomic motion planning must be studied. The optimal path, if existent, must be identified. The new motion planner has to take into account both nonholonomic constraint and uncertainties, including robot geometry, and the planned path obtained from the motion planner must be optimal.

Robot Motion Control. The motion control problem for nonholonomic mobile robots has also recently drawn a lot of attention from many researchers. In general, the study of the motion control problem can be classified into two main problems: the stabilization problem and the tracking problem [Díaz del Río et al., 2002]. The stabilization problem concentrates on getting the robot to a fixed point in its workspace, while the tracking problem focuses on controlling the robot to follow a desired trajectory or path. Clearly, the tracking problem is of most interest to us. The tracking problem can be further divided into two groups. The first one is "trajectory tracking", where the robot must reaches the desired position at the specific time. The second one is "path following", where time dependence is not relevant because only the geometric displacement between the robot and the path is concerned [De Luca et al., 1998]. This time-independence characteristic makes the path following the most suitable candidate for our motion control scheme. In what follows, we gives the literature review of the existing works in path following control for nonholonomic mobile robots. The main attention will be focused on robust path following control for wheeled and tracked mobile robots.

The types of control for path following control can be largely categorized into two classes: linear and nonlinear controls. For linear control, Oelen and van Amerongen [Oelen and van Amerongen, 1994] proposed a linear controller that was robust to the perturbation in robot velocity control. The experiment with a mobile robot demonstrated good tracking performance, given that the initial condition of the robot was close to the desired path. Chung, et al. [Chung et al., 2001] also used a linear control method in controlling their experimental mobile robot platform. By using separated feedback loops for robot position and velocity, the controller was robust to control disturbance, and achieved good tracking performance on a straight path. Pears [Pears, 2001] presented a linear path following control for a mobile robot. The kinematic model of the mobile robot was linearized, and a simple proportional linear control was used. An intuitive velocity dependency concept was proposed, where the linear velocity of the robot varies depending on the curvature of the desired path. The controller exhibited a good tracking performance when the initial condition was close to the path. A popular PID controller was used by Normey-Rico et al. [Normey-Rico et al., 2001]. The proposed method used a simple linearized model of the mobile robot in the synthesis of a robust PID controller. Experimental results showed good tracking performance despite the existence of the model uncertainty.

Although the linear control has huge advantage due to its simplicity and advance in linear control theory, its performance and robustness are very limited when compare with the nonlinear control. For example, the initial state of the mobile robot must stay close to the reference to achieve stability. Therefore, the nonlinear control scheme—although harder to synthesize—seems to be a better alternative. Lyapunov-based design utilizing Lyapunov's direct method is among popular approaches used in nonlinear control design. Sørdalen and Canudas de Wit [Sørdalen and Canudas de Wit, 1993] proposed a nonlinear feedback control law that yielded exponential convergence to the desired path composed of straight lines and circular arcs. By using a coordinate transformation, the control law was derived, and its stability was proved by Lyapunov stability theory. Aicardi et al. [Aicardi et al., 1994] also used the coordinate transformation by representing the Cartesian position of the mobile robot in terms of its polar coordinates. Another interesting work based on the coordinate transformation can be found in [Aguilar et al., 1998]. Tayebi and Rachid [Tayebi and Rachid, 1996] presented a nonlinear control law based on partial state feedback linearization and Lyapunov's direct method. Although disturbance or uncertainty was not considered in the control design as the two previous works, the experimental results still showed good tracking performance on both rectilinear and circular paths.

Soft computing-based design is another well-known and widely accepted method for synthesizing the nonlinear controller for mobile robots. Ollero et al. [Ollero et al., 1994] combined fuzzy logic with the geometric pure-pursuit technique—the path tracking technique that follows the path by repeatedly fitting circular arcs to different goal points on the path as the mobile robot moves forward—and the generalized predictive control method to perform path following task. In the experiment, the proposed control showed significant improvement in tracking performance over the pure-pursuit technique alone. Yang et al. [Yang et al., 1998] also applied an artificial neuron network in their predictive control method. The network was used to predict future robot posture according to the current posture and control commands. The experiment results yielded satisfactory tracking performance, and proved the proposed controller reliable and robust. A work by Fierro and Lewis [Fierro and Lewis, 1998] was the application of the neuron network with backstepping control approach (one of the Lyapunov-based design methods) to develop a control law for path following, path tracking, and point stabilization. The proposed control could also cope with unmodeled bounded disturbances and unmodeled dynamics in the robot as shown in the simulation results. Homaifar et al. [Homaifar et al., 1999] proposed a controller utilizing a Genetic Algorithm (GA) and a fuzzy logic control. The GA was applied to learn fuzzy control rules, and used to adapt the membership functions and rule bases for the fuzzy logic controller. Through simulations, the controller showed good tracking performance, and its robustness to sensor noise was also demonstrated.

Other nonlinear control schemes were also often found in literature. Sarkar et al. [Sarkar et al., 1993] used input-output linearization and decoupling scheme to design a nonlinear feedback controller for mobile robots. The obtained controller not only tracks straight and circular paths accurately, but also was robust to modeling errors and sensor noise. The pure pursuit path-tracking algorithm, one of the most widely used strategies, was employed in Ollero and Heredia's work [Ollero and Heredia, 1995]. They proposed and proved the stability of the geometric pure pursuit for straight and circular path. The algorithm was implemented on a full-size robotic vehicle, and the experiment results showed satisfactory tracking performance.

The method of optimal control has recently been applied to the path following problem. The optimal control normally yields open-loop control and takes too much computation time, which is useless for real-time feedback control. However, in the case of car-like robots where the closed-form for the shortest path exists, the optimal feedback control can be synthesized. As a tradeoff, the optimal control often leads to bang-bang control when traveling time must be minimized. This type of control usually produces vibratory phenomena known as "chattering" when it is implemented on physical plant, as clearly shown in [Souéres et al., 2000].

Sliding mode control is also widely accepted by many researchers due to its capability to cope with uncertainty, and, therefore, is very popular for robust control design. Aguilar et al. [Aguilar et al., 1997] used sliding mode control to derive a path following feedback controller robust with respect to localization error. The experimental results showed good tracking performance, and agreed with their control robustness analysis.

Notice that all of the studies mentioned above are mainly concentrated on developing stabilizing controllers robust to modeling and/or sensing and/or control errors, assuming an obstacle-free workspace. Although the robot is guaranteed to follow the desired path, the collision-free condition during stabilization process is not guaranteed. To the best of the author's knowledge, only two works take into account the robot state constraint imposed by obstacles in a robot workspace while stabilizing the robot around the desired path. One was the work by Hamel et al. [Hamel et al., 2001]. Using variable structure control such as sliding mode control, the size of the attractive domain around the desired path could be determined. The information about the attractive domain could be used in selecting appropriate controller gains. However, the selecting procedure was done by trial and error and could not be achieved in real-time. The other was the work by Souères et al. [Souères et al., 1998], a path following controller based on a sliding mode technique is proposed. The controller can guide the robot around an unexpected obstacle encountered before the robot reaches the pre-planned paths by using the strategy similar to the wall-following strategy. However, the sizes of obstacle and workspace considered in the work are relatively large when compared to the size of the robot; as a result, this control scheme is not applicable to our application.

The literature review in robot motion control shows that none of the previous works explicitly consider the path following problem for nonholonomic mobile robots in tight cluttered workspaces, where the size of the free space are comparable to the size of robot. To address the shortcoming, a new motion control strategy—that takes into account the state constraints imposed by surrounding obstacles—is needed. Additionally, the control strategy must be robust, and must work in real-time.

1.3 Research Objectives

Let us assume that a partially known and static 2-D working environment, a mobile robot kinematic model, a robot dimension, bounded control and sensing uncertainties, initial and final configurations, and control constraints are given. The primary objectives of our research are:

1. To develop a motion planning algorithm for nonholonomic mobile robots under control and sensing uncertainties. The algorithm plans the most robust path that the robots must follow, so that the likelihood of collisions due to the presence of uncertainties is eliminated.

2. To develop a method for determining the allowable maximum degree of control and sensing uncertainties for the mobile robots to follow a given path safely.

3. To develop a method for determining the fastest speed for the mobile robot to follow a given path safely.

4. To develop a real-time path following control algorithm for nonholonomic mobile robots subject to state constraints and uncertainties.

1.4 Overview of the Dissertation

The dissertation is organized as follows.

Chapter 2: Mathematical Preliminaries

A fundamental theory and the detail of mathematical method used in this work are summarized here. In particular, the basic of dynamic optimization is given, and the detail of a direct search optimization method called "Nelder-Mead simplex method" is discussed. This optimization method is deployed in Chapter 3. A popular graph search algorithm called "A* algorithm" is also presented in detail, and it is later used in Chapter 4. In the last part of the chapter, an overview of nonlinear control theory for uncertain autonomous nonlinear systems is reviewed. Some of the theorems stated in this part are used to conclude the stability of the derived robust controller in Chapter 6. Some level of familiarity in the topics mentioned above is assumed.

Chapter 3: Linear Control Uncertainty Field-Based Motion Planning

This chapter begins with mobile robot and workspace models, which are used throughout this research. The types of uncertainties in robot systems and how they can be combined are also discussed. The concept of combined uncertainty leads to the introduction of "Linear Control Uncertainty (LCU)", and later has been developed into the "Liner Control Uncertainty Field (LCUF)". A new type of nonholonomic motion planning called the "LCUF-based motion planning" is proposed. A proof of the optimality of the path obtained from the path planning and the estimation of safe control uncertainty level for a mobile robot are also provided.

Chapter 4: Circular Control Uncertainty Field-Based Motion Planning

This chapter presents the extension to the concept of the LCU, which we call the "Circular Control Uncertainty (CCU)". The CCU removes the limitation of dealing with only a straight nominal path of the LCU. Therefore, it can determine the exact control uncertainty for any circular arc with arbitrary curvature. A field of CCU is utilized in a new motion planning called "CCUF-based motion planning". This motion planning based on a popular heuristic search method "A* algorithm" is guaranteed to return an optimal path—consisting of circular arcs and straight lines—if at least one feasible path exists. Conditions for collision-free motion of a mobile robot are also concluded at the end of the chapter.

Chapter 5: Motion Planning Performance Comparison and Improvement

The advantages and disadvantages of both LCUF-based and CCUF-based motion planners are identified and compared. The result reveals that the LCUF-based motion planner is more preferable in terms of planned path qualities. Due to the drawback of the LCUF-based motion planner that requires substantial optimization time, a solution is proposed by the introduction of the estimation method for the LCU, which is called the "estimated LCU". The proposed estimation method can accurately approximate the value of LCU, and reduces the optimization time by at least an order of magnitude. The global optimality property of a field of estimated LCU is also given at the end of the chapter. This property guarantees the optimality of the resultant path returned from all of the proposed motion plannings.

Chapter 6: Robust Path Following Control under State Constraints

This chapter presents a real-time robust path following control algorithm for nonholonomic mobile robots under uncertainty and state constraints. This control algorithm can navigate a convex polygon robot safely through a tight workspace, where other algorithms fail. The collision-free condition is also guaranteed during the navigation. The detail development of the control algorithm is described, and its algorithm is explicitly stated. By integrating the knowledge of the workspace through the LCUF into a robust path following control, we obtain the "LCUF-based robust path following control algorithm", which possesses all nice characteristics previously mentioned. The control algorithm demonstrates its remarkable performance through the simulation results given at the end of the chapter.

Chapter 7: Conclusions

This chapter is where all conclusions about this research are drawn. A summary of the whole dissertation is given at the beginning, including with the contributions we have made toward the field of mobile robot motion planning and control. Recommendations for future work are also given at the end to conclude this dissertation.

Chapter 2

Mathematical Preliminaries

The background of all mathematics, methods, and algorithms used in developing the work in this dissertation is given in this chapter. The chapter is organized into three sections. Each section is completely separated, and does not share common notation and symbols. The first section describes a dynamic optimization problem and a direct search optimization method called "Nelder-Mead simplex method", which is used in Chapter 3. Basic background in optimal control theory and optimization is assumed. The second section discusses a graph search method called "A* algorithm" (pronounced "A-star"), which is employed in Chapter 4. Most of the material in this section is summarized from the books by Latombe [Latombe, 1991] and Nilsson [Nilsson, 1980]. In the last section, a review of Lyapunov stability theory for autonomous nonlinear systems is given, along with the theorem of boundedness for uncertain system. These tools are necessary for concluding the stability of the robust control design presented in Chapter 6. Basic knowledge in real analysis and nonlinear control is assumed.

2.1 Dynamic Optimization

Dynamic optimization is the process of determining control and state histories for a dynamic system over a finite time period to minimize a given performance index, and it is commonly used to determine efficient maneuvers of aircraft, spacecraft, and robots [Bryson, 1999]. Although the main tools of dynamic optimization are the calculus of variations and dynamic programming, many of the dynamic optimization problems may be transformed into general optimization problems. Let us consider a discrete dynamic system, which is described by an *n*-dimensional state vector *x(i)* at step *i*. Choice of an *m*-dimensional control vector *u(i)* determines a transition of the system to state *x(i+1)* through the relations

$$x(i+1) = f[x(i), u(i), i], \tag{2.1}$$

where

$$x(0) = x_0 . \tag{2.2}$$

A general optimization problem for such system is to find the sequence of control vectors *u(i)* for *i* = 0,…, *N*-1 to minimize a performance index (or cost function), of the form

$$F = \phi[x(N)] + \sum_{i=0}^{N-1} L[x(i), u(i), i], \tag{2.3}$$

subject to (2.1) and (2.2) with *N*, x_0, and the function *f* specified. This is a standard form of a parameter optimization problem, so it can be solved using the existing optimization methods, by treating the control vector histories *u(i)* as the unknown parameters [Bryson, 1999]. Given an initial guess of *u(i)*, the state histories *x(i)* can be calculated from (2.1) to determine *F*, and optimization methods such as Newton-type methods can be used to modify *u(i)* to minimize *F*. To handle the constrained dynamic optimization problem subject to control and state constraints, the problem can be easily transformed to the unconstrained dynamic optimization problem by introducing penalty functions of constraints to (2.3). This optimization scheme can also be applied to the dynamic optimization problem of continuous dynamic systems by changing step *i* to time *t*, changing the difference equation in (2.1) to the differential equation and the summation sign in (2.3) to the integration sign, and treating the control vector *u(i)* as a piecewise-constant control vector.

After the dynamic optimization problem has been transformed into the parameter optimization problem, a suitable optimization method must be identified. In the case where the cost function *F* is discontinuous, especially where the gradient vector of *F* is discontinuous in the neighborhood of the solution, direct search methods (using only cost function values, no gradient calculations required) are more preferable than others [Gill et al., 1981]. One of the well-known direct search methods is the Nelder-Mead simplex method. It has become one of the most widely used methods for multidimensional nonlinear unconstrained optimization since its publication in 1965 [Lagarias et al., 1998]. At each step, the Nelder-Mead method maintains a nondegenerate simplex, which is a geometric figure in *n* dimensions of nonzero volume that is the convex hull of *n*+1 vertices. This means that if the problem is an *n*-dimensional optimization problem, one requires n+1 points in its state space to form a simplex. Four scalar parameters must be specified at the beginning of the optimization: coefficients of *reflection* (ρ), *expansion* (χ), *contraction* (γ), and *shrinkage* (σ). These parameters control the overall characteristics of the method, which reflect the rate of convergence of the method. The following detail algorithm of the Nelder-Mead method is extracted from the paper by Lagarias et al. [Lagarias et al., 1998].

In each iteration of the Nelder-Mead algorithm, the following computations are performed:

1. **Order**. Order the *n*+1 vertices, so they satisfy $F(x_1) \leq F(x_2) \leq \ldots \leq F(x_{n+1})$, where $F(x_i) = F_i$ is the cost function value at x_i.
2. **Reflect**. Compute the reflection point x_r from the following equation

$$x_r = \bar{x} + \rho(\bar{x} - x_{n+1}), \tag{2.4}$$

where $\bar{x} = \sum_{i=1}^{n} x_i / n$ is the centroid of the *n* best points, which are all vertices except for x_{n+1}. Then, evaluate $F_r = F(x_r)$. If $F_1 \leq F_r < F_n$, accept the reflected point x_r and terminate the iteration.

3. **Expand**. If $F_r < F_1$, calculate the expansion point x_e from

$$x_e = \bar{x} + \chi(x_r - \bar{x}), \tag{2.5}$$

and then evaluate $F_e = F(x_e)$. If $F_e < F_r$, accept x_e and terminate the iteration; otherwise, accept x_r and terminate the iteration.

4. **Contract**. If $F_r \geq F_n$, perform a contraction between $\bar{x}$ and the better of x_{n+1} and x_r.
    a. **Outside**. If $F_n \leq F_r < F_{n+1}$, perform an outside contraction by calculating

$$x_c = \bar{x} + \gamma(x_r - \bar{x}), \tag{2.6}$$

and evaluating $F_c = F(x_c)$. If $F_c \leq F_r$, accept x_c and terminate the iteration; otherwise, go to step 5.

    b. **Inside**. If $F_r \geq F_{n+1}$, perform an inside contraction by calculating

$$x_{cc} = \bar{x} - \gamma(\bar{x} - x_{n+1}) \tag{2.7}$$

and evaluating $F_{cc} = F(x_{cc})$. If $F_{cc} < F_{n+1}$, accept x_{cc} and terminate the iteration; otherwise, go to step 5.

5. **Perform a shrink step**. Evaluate F at the n points $v_i = x_1 + \sigma(x_i - x_1)$, $i = 2, \ldots, n+1$. The (unordered) vertices of the simplex at the next iteration consist of $x_1, v_2, \ldots, v_{n+1}$.

The algorithm keeps iterating until no vertex yielding lower cost function value is found, or the vertex x_1 is almost unchanged. Then, x_1 is declared as a solution, and the algorithm is terminated. For further detail discussion, see [Lagarias et al., 1998] and [Nelder and Mead, 1965].

2.2 Graph Search Methods

Some of the motion planning methods transform a "continuous" problem, such as finding a path in a manifold, into the "discrete" problem of searching a graph (e.g., the visibility graph, the Voronoi diagram, the connectivity graph) between an initial node and a goal node [Latombe, 1991]. Graph search methods can be classified into two main types: uniformed and heuristic graph search methods [Nilsson, 1980]. The uniformed graph search methods (e.g., depth-first search, breadth-first search) search a graph blindly without heuristic information from the problem domain. Thus, they are very inefficient and typically used in the problems that have no heuristic information available. The heuristic graph search methods (e.g., A algorithm, A* algorithm, bi-directional search, staged search), on the other hand, take advantage of task-independent information to help reduce search space, and, as a result, take substantially less time to search. This makes the heuristic search methods are far more favorable, when applicable, than the uniformed graph search methods. One of the most popular heuristic search methods is A* algorithm. The algorithm is guaranteed to return an optimal path (when the heuristic function used is admissible) from an initial node to a goal node whenever a path from an initial node to a goal node exists, and to return failure otherwise. Before we discuss the A* algorithm in more detail, a notation of graph is introduced.

Graph Notation. A *graph G* consists of a set of *nodes X*, which may or may not be finite. Certain pairs of nodes are connected by *arcs A*, and these arcs are *directed* from one member of the pair to the other. Such a graph is called a *directed graph*. If an arc is directed from node N_i to node N_j, then node N_j is said to be a *successor* of node N_i, and node N_i is said to be a *parent* of node N_j. A tree is a special case of graph in which each node has at most one parent [Nilsson, 1980]. A node N_i is adjacent to N if there is an arc of G connecting N to N_i [Latomb, 1991].

The followings explain the concept of the A* search algorithm, and a detail psudo-code of the algorithm—which can be applied directly for the implementation of a search program—is also provided. Given a directed graph G, the A* algorithm begins its search in the graph at an initial node N_{init}, and iteratively searches the graph by following paths originating from the initial node. In each iteration, there are some nodes that the algorithm has already visited, and there may be others that are still unvisited. For each visited node N, there is at least one path (generated in the previous iterations) connecting N_{init} to N; however, the algorithm only memorizes the path possessing the minimum cost. During the search, the algorithm also spans a tree T, which is formed by the set of all paths of the subset of G searched so far. T is represented by associating to each visited node N (except N_{init}) a pointer to its parent node in the current T. The A* algorithm assigns a cost function $f(N)$ to every node N in the current T. This function is an estimate of the cost of minimum-cost path in G connecting N_{init} to a goal node N_{goal} and constrained to go through N. It is computed as follows:

$$f(N) = g(N) + h(N) \tag{2.8}$$

where:

$g(N)$ is the cost of the path between N_{init} and N in the current T,

$h(N)$ is a heuristic estimate of the cost $h^*(N)$ of the minimum-cost path between N and N_{goal} in G.

The detail of the A* algorithm from the book by Latombe [Latombe, 1991] is given below.

Given *G*, N_{init}, N_{goal}, *k*, and *h*, where $k : X \times X \to R^+$ is the partial function specifying the cost of each arc in *G*. Before the algorithm starts, all the nodes of *G* are marked *unvisited*. During the search, the algorithm makes use of a list denoted by *OPEN* that contains nodes of *G* sorted by the values of the function *f*. The list *OPEN* supports the following operations:

- FIRST(*OPEN*): remove the node of *OPEN* with the smallest value of *f* and return it,
- INSERT(*N*,*OPEN*): insert node *N* in *OPEN*,
- DELETE(*N*,*OPEN*): remove node *N* from *OPEN*,
- MEMBER(*N*,*OPEN*): evaluate to `true` if *N* is in *OPEN* and to `false` otherwise,
- EMPTY(OPEN): evaluate to `true` if *OPEN* is empty and to `false` otherwise.

At each iteration, A* explores the nodes adjacent to the node returned by FIRST(*OPEN*). Initially, both the tree *T* and the list *OPEN* are empty. A psudo-code for the A* algorithm is as follows.

```
procedure A*($G$, $N_{init}$, $N_{goal}$, $k$, $h$);
  begin
    install $N_{init}$ into $T$;
    INSERT($N_{init}$, $OPEN$); mark $N_{init}$ visited;
    while not EMPTY($OPEN$) do
      begin
        $N \leftarrow$ FIRST($OPEN$);
        if $N = N_{goal}$ then exit while-loop;
        for every node $N_i$ adjacent to $N$ in $G$ do
          if $N_i$ is not visited then
            begin
              add $N_i$ to $T$ with a pointer toward $N$;
              INSERT($N_i$, $OPEN$); mark $N_i$ visited;
            end;
          else if $g(N_i) > g(N) + k(N, N_i)$ then
```

```
                begin
                   modify $T$ by redirecting the pointer of $N_i$ toward $N$;
                   if MEMBER($N_i$,$OPEN$) then DELETE($N_i$,$OPEN$);
                   INSERT($N_i$,$OPEN$);
                end;
         end; /*end of while-loop*/
      if not EMPTY($OPEN$) then
         return the constructed path by tracing the pointers in $T$ from
         $N_{goal}$ back to $N_{init}$;
      else return failure;
   end; /*end of procedure*/
```

From the above psudo-code, the main loop (while-loop) examines all of the nodes adjacent to a node N. This examination process is called the *expansion* of N, and it produces a visited node N_i. The inner loop (for-loop) makes sure that, during the expansion of N, if there is a new path to go from N_{init} to N_i that possess less cost, the algorithm then updates T by redirecting the pointer issued from N_i. If N_i is in *OPEN*, its position in the list must be updated according to the new value of $f(N_i)$. If N_i is not in *OPEN*, the algorithm inserts N_i in *OPEN*.

The heuristic function $h(\cdot)$ is said to be *admissible* if and only if it satisfies:

$$\forall N \in G \;:\; 0 \leq h(N) \leq h^*(N). \tag{2.9}$$

Where, $h^*(N)$, is the cost of the minimal cost path from N to N_{goal} [Latombe, 1991]. If $h(\cdot)$ is admissible, the A* algorithm is guaranteed to return a minimum-cost path between the initial and goal nodes whenever there exists a path connecting these two nodes in a directed graph G, and to return failure otherwise.

The proof of optimality of the A* search algorithm can be found in [Nilsson, 1980]. For more detail explanation of the A* algorithm and its psudo-code, the reader should see [Latombe, 1991].

2.3 Nonlinear System Analysis and Robust Control Design

In this section, brief reviews of the nonlinear system analysis and robust control for autonomous nonlinear systems are presented. Theorems and lemmas provided here can be found from the books by Khalil [Khalil, 1996] and Qu [Qu, 1998].

Existence and Uniqueness. Consider a mathematical model of a nonlinear system in the form:

$$\dot{x} = f(t,x), \tag{2.10}$$

where $x \in R^n$ is the state vector with initial condition $x(t_0) = x_0$. For the mathematical model to be useful, the solution of (2.10) must be exist and unique. The following theorem guarantees the local existence and uniqueness of the solution of (2.10).

Theorem 2.1 (Local Existence and Uniqueness) Let *f(t,x)* be piecewise continuous in *t* and satisfy the *Lipschitz condition*

$$\|f(t,x) - f(t,y)\| \le L\|x - y\| \tag{2.11}$$

$\forall x, y \in B = \{x \in R^n \mid \|x - x_0\| \le r\}$, $\forall t \in [t_0, t_1]$. Then, there exists some $\delta > 0$ such that the state equation (2.10), with $x(t_0) = x_0$ has a unique solution over $[t_0, t_0 + \delta]$. See [Khalil, 1996] for proof.

A function satisfying the Lipschitz condition (2.11) is said to be *Lipschitz* in x, and the positive constant *L* is called a *Lipschitz constant*. However, from Theorem 2.2, δ may be very small; hence, we cannot ensure existence and uniqueness over a given time interval $[t_0, t_1]$. A more restrictive, but conservative, theorem guarantees the existence of a unique solution over $[t_0, t_1]$ where t_1 may be arbitrary large.

Theorem 2.2 (Global Existence and Uniqueness) Suppose *f(t,x)* is piecewise continuous in *t* and satisfies

$$\begin{aligned} \|f(t,x)-f(t,y)\| &\le L\|x-y\| \\ \|f(t,x_0)\| &\le h \end{aligned} \tag{2.12}$$

$\forall x, y \in R^n, \forall t \in [t_0, t_1]$, and $h = \max_{t\in[t_0,t_1]} \|f(t,x_0)\|$. Then, the state equation (2.10), with $x(t_0) = x_0$ has a unique solution over $[t_0, t_1]$. See [Khalil, 1996] for proof.

The global Lipschitz property required in Theorem 2.2 is too restrictive, and models of many physical systems fail to satisfy it. The following theorem does not require the global Lipschitz condition, and only requires *f* to be locally Lipschitz. However, more knowledge about the solution of the system is required.

Theorem 2.3 Let *f(t,x)* be piecewise continuous in *t* and locally Lipschitz in *x* for all $t \ge t_0$ and all *x* in a domain $D \subset R^n$. Let *W* be a compact subset of *D*, $x_0 \in W$, and suppose it is known that every solution of the state equation (2.10), with $x(t_0) = x_0$ lies entirely in W. Then, there is a unique solution that is defined for all $t \ge t_0$. See [Khalil, 1996] for proof.

Stability Concepts. From the book by Qu [Qu, 1998], the standard definitions of stability of equilibrium points in the sense of Lyapunov are given below.

Definition 2.1 The equilibrium point $x = 0$ is said to be *Lyapunov stable* (or, in short, *stable*), at time t_0 if, for each $\varepsilon > 0$, there exists a constant $\delta(t_0, \varepsilon) > 0$ such that

$$\|x(t_0)\| < \delta(t_0, \varepsilon) \Rightarrow \|x(t)\| \le \varepsilon \qquad \forall t \ge t_0 .$$

It is said to be *uniformly Lyapunov stable* (or, *uniformly stable*) over $[t_0, \infty)$ if, for each $\varepsilon > 0$, the constant $\delta(t_0, \varepsilon) = \delta(\varepsilon) > 0$ is independent of initial time t_0.

Definition 2.2 The equilibrium point $x = 0$ is said to be *attractive* at time t_0 if, for some $\delta > 0$ and each $\varepsilon > 0$, there exists a finite time interval $T(t_0, \delta, \varepsilon)$ such that

$$\|x(t_0)\| < \delta \Rightarrow \|x(t)\| \le \varepsilon \qquad \forall t \ge t_0 + T(t_0, \delta, \varepsilon).$$

It is said to be *uniformly attractive* over $[t_0, \infty)$ if for all ε satisfying $0 < \varepsilon < \delta$, the finite time interval $T(t_0, \delta, \varepsilon) = T(\delta, \varepsilon)$ is independent of initial time t_0.

Definition 2.3 The equilibrium point $x = 0$ is *asymptotically stable* at time t_0 if it is Lyapunov stable at time t_0 and if it is attractive, or equivalently, there exists $\delta > 0$ such that

$$\|x(t_0)\| < \delta \Rightarrow \|x(t)\| \to 0 \qquad as \qquad t \to \infty$$

It is *uniformly asymptotically stable* over $[t_0, \infty)$ if it is uniformly Lyapunov stable over $[t_0, \infty)$, and if $x = 0$ is uniformly attractive.

Definition 2.4 The equilibrium point $x = 0$ at time t_0 is *exponentially attractive* if, for some $\delta > 0$, there exist constants $\alpha(\delta) > 0$ and $\beta > 0$ such that

$$\|x(t_0)\| < \delta \Rightarrow \|x(t)\| \le \alpha(\delta) \exp^{-\beta(t-t_0)}.$$

It is said to be *exponentially stable* if, for some $\delta > 0$, there exist constants $\alpha(\delta) > 0$ and $\beta > 0$ such that

$$\|x(t_0)\| < \delta \Rightarrow \|x(t)\| \le \alpha(\delta) \|x(t_0)\| \exp^{-\beta(t-t_0)}.$$

The above definitions are designed for systems whose equilibrium points are at origin, and can be easily extended to systems with a known but nonzero equilibrium state by means of coordinate transformation. However, for uncertain systems in which some of dynamics are unknown, it is sometimes impossible for one to determine the equilibrium state. As a result, the above stability definitions are not suitable when considering uncertain systems. The following are two stability definitions, which are more suitable for uncertain systems [Qu, 1998].

Definition 2.5 A solution $x : R^+ \rightarrow R^n$, $x(t_0) = x_0$, is said to be *uniformly bounded* if, for some $\delta > 0$, there is a positive constant $d(\delta) < \infty$, possibly dependent on δ (or x_0) but not on t_0, such that, for all $t \geq t_0$,

$$\|x(t_0)\| < \delta \Rightarrow \|x(t)\| \leq d(\delta).$$

Definition 2.6 A solution $x : R^+ \rightarrow R^n$, $x(t_0) = x_0$, is said to be *uniformly ultimately bounded* with respect to a set $W \subset R^n$ containing the origin if there is a nonnegative constant $T(x_0, W) < \infty$, possibly dependent on x_0 and W but not on t_0, such that $\|x(t_0)\| < \delta$ implies $x(t) \in W$ for all $t \geq t_0 + T(x_0, W)$.

The set W in Definition 2.6, called residue set, is usually characterized by a hyper-ball $W = B(0, \varepsilon)$ center at the origin and of radius ε. The uniformly ultimately bounded stability is often the best result achievable in controlling uncertain systems.

Lyapunov Function. The following definitions give a mathematically precise description of admissible energy-like functions [Qu, 1998].

Definition 2.7 A continuous function $\gamma : R^+ \rightarrow R^+$ is a class $\mathcal{K}$ function if $\gamma(0) = 0$ and if it is strictly increasing. It is said to belong to class $\mathcal{K}_\infty$ if $\gamma(p) \rightarrow \infty$ as $p \rightarrow \infty$.

Definition 2.8 A function $V : R^n \times R \rightarrow R^+$ is called locally positive definite if there exists a class $\mathcal{K}$ function $\gamma_1 : R^+ \rightarrow R^+$ such that, for some neighborhood of the origin $\Omega \subset R^n$,

$$\gamma_1(\|x(t)\|) \leq V(x(t), t), \qquad \forall (x, t) \in \Omega \times R^+.$$

Function V is said to be locally decrescent if there exists a class $\mathcal{K}$ function $\gamma_2 : R^+ \rightarrow R^+$ such that, for some neighborhood of the origin $\Omega \subset R^n$,

$$V(x(t),t) \leq \gamma_2(\|x(t)\|), \qquad \forall (x,t) \in \Omega \times R^+ .$$

The word "locally" is replaced by "global" if $\Omega = R^n$. Function V is radially unbounded if γ_1 is a class $\mathcal{K}_\infty$ function.

Definition 2.9 A function $V : R^n \times R \to R^+$ is a Lyapunov function candidate if it is continuously differentiable and if

(i) For concluding stability, *V(x,t)* is positive definite.

(ii) For concluding uniform asymptotic stability or exponential stability or uniform boundedness or uniform ultimate boundedness, *V(x,t)* is positive definite and decrescent.

(iii) For concluding global stability, *V(x,t)* is globally positive definite and radially unbounded.

(iv) For concluding global and uniform asymptotic stability or global exponential stability or global uniform boundedness or global uniform ultimate boundedness, *V(x,t)* is globally positive definite, globally decrescent, radially unbounded.

Basic Stability Theorem. The standard Lyapunov stability theorem is stated below [Qu, 1998].

Theorem 2.4 Consider ordinary differential equation (2.10) in which, for any bounded set $D \subset R^n$, function *f* maps $D \times R^+$ into bounded sets in R^n. Let V be a Lyapunov function candidate as defined in Definition 2.9 in some neighborhood of the origin denoted by $\Omega \subset R^n$. Suppose the time derivative of V along the solution to (2.10) has the property that, for all $(x,t) \in \Omega$,

$$\dot{V}(x(t),t) \leq -\gamma_3(\|x(t)\|),$$

where γ_3 is continuous and nonnegative with $\gamma_3(0) = 0$. Then, the system has the following stability property:

(i) either globally or locally uniformly Lyapunov stable if $\gamma_3(\|x(t)\|)$ is positive semidefinite,

(ii) either globally or locally uniformly asymptotically stable if $\gamma_3(\|x(t)\|)$ is positive definite,

(iii) either globally or locally exponentially stable if $\gamma_3(\|x(t)\|) \geq \lambda V(x,t)$ for some constant $\lambda > 0$ or if $\gamma_i(\|x(t)\|) = \lambda_i \|x\|^2$ for i = 1, 2, 3 and for positive constants λ_i,

(iv) either globally or locally exponentially stable with finite convergence time if $\gamma_3(\|x(t)\|) \geq \lambda V^p(x,t)$ for constants $\lambda > 0$ and $0 < p < 1$.

Proof: see [Qu, 1998] and [Khalil, 1996] for proof.

Theorem 2.4 can be used to conclude both local and global, in which stability results are global if the set $\Omega = R^n$ and if the conditions in Definition 2.9 are satisfied. The extension of the above Lyapunov stability theorem to develop stability results without requiring that *V* be positive definite or that $\dot{V}$ be negative definite is LaSalle's theorem, which is based on the concept of invariant set [Qu, 1998].

Definition 2.10 A set *M* is called *invariant set* for an autonomous system if every trajectory starting from a point in *M* will remain in the set for all future time.

Definition 2.11 A set *M* is said to be a *positively invariant set* if

$$x(0) \in M \Rightarrow x(t) \in M, \quad \forall t \geq 0.$$

Theorem 2.5 Let $\Omega \subset D$ be a compact set that is positively invariant with respect to $\dot{x} = f(x)$.

Let $V : D \rightarrow R$ be a continuously differentiable function such that $\dot{V}(x) \leq 0$ in Ω. Let *E* be the set of all points in Ω where $\dot{V}(x) = 0$. Let *M* be the largest invariant set in *E*. Then every solution starting in Ω approaches *M* as $t \rightarrow \infty$. See [Khalil, 1996] for proof.

Theorems of Boundedness. In the case of uncertain systems, the basic stability theorem mentioned above cannot be directly applied since the origin may not be the equilibrium state of the systems. When dealing with the uncertain systems, theorems of boundedness (an extension of the Lyapunov method) are more appropriate for concluding the stability of the systems. There are many stability theorems proposed for the uncertain systems. One of them is LaSalle's general theorem on uniform boundedness and uniform ultimate boundedness [Qu, 1998]. Before we state the theorem, consider the following definition:

Definition 2.12 Let *M* be a closed set in R^n. For any positive constant *ε*, M_ε is called the ε-neighborhood of set *M* if M_ε is the set of all points whose distance from *M* is less than *ε*, or mathematically,

$$M_\varepsilon = \{x \in R^n : \|x - y\| < \varepsilon, \quad \forall y \in M\}.$$

Using this definition, LaSalle's general theorem on boundedness is stated below [Qu, 1998].

Theorem 2.6 Suppose that, for any bounded set $D \subset R^n$, function *f* maps $D \times R^+$ into bounded sets in R^n. Let *V(x,t)* be a scalar function with continuous first-order partial derivatives and *M* be a closed set in R^n. Then,

(i) If $\dot{V}(x,t) \leq 0$ for all $x \in M^c$ and if $V(x_1, t_1) < V(x_2, t_2)$ for all $t_2 \geq t_1 \geq t_0$, all $x_1 \in M$ and all $x_2 \in M_\varepsilon^c$, then every trajectory of system (2.10) which at some time is in *M* can never thereafter leave M_ε.

(ii) If, in addition to the conditions in (i), $V(x,t) \geq 0$ for all $x \in R^n$ and $t \geq t_0$ and $\dot{V}(x,t) \leq -\varepsilon < 0$ for all $t \geq t_0$ and $x \in M_\varepsilon^c$, then every trajectory of system (2.10) will enter and then stay inside M_ε in a finite time.

(iii) If, in addition to the conditions in (ii), the set *M* is bounded and *V(x,t)* is radially unbounded, then every trajectory of system (2.10) is globally uniformly ultimately bounded with respect to the set M_ε.

As in LaSalle's invariant set theorem, part (i) of the above theorem does not require that function *V(x,t)* be positive definite. The proof of the theorem can be found in [LaSalle and Lefschetz, 1961].

Chapter 3
Linear Control Uncertainty Field-Based Motion Planning

In this chapter, mobile robot and workspace models used throughout this research are given in the first section. The types of uncertainties in robot systems and how they can be combined are also discussed before we introduce the definition of "Linear Control Uncertainty Field" leading to the discussion of a new type of nonholonomic motion planning (LCUF-based motion planning), which is the main result of this chapter. A proof of the optimality of the path obtained from the path planning and the estimation of safe control uncertainty level for a mobile robot are provided at the end of the chapter.

3.1 Robot and Workspace Models

Robot Model. A kinematic model for a differentially-driven wheeled mobile robot is selected to be the robot model for both wheeled and tracked mobile robots. Equation (3.1) shows the system of first-order ordinary differential equations representing the robot model.

$$\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.1}$$

$$x = [x_c, y_c, \theta]^T \text{ and } u = [v, \omega]^T$$

where x is the system state, and (x_c, y_c) are the coordinates of the origin of the robot-fixed coordinate frame *{xy}* with respect to the global coordinate frame *{XY}*, see Figure 3.1. The angle θ represents the robot heading angle with respect to the X-axis. The control u consists of the linear velocity v and the angular velocity ω.

From (3.1), the controls of the mobile robot are linear and angular velocities. However, the actual controls we provide to the mobile robot are right and left wheel velocities as shown in Figure 3.1. The following equations can be used to computed the linear and angular velocities when the values of right and left wheel velocities, v_r and v_l, are available.

$$\begin{aligned} v &= \frac{v_r + v_l}{2} \\ \omega &= \frac{v_r - v_l}{B} \end{aligned} \tag{3.2}$$

where B is the robot wheel base.

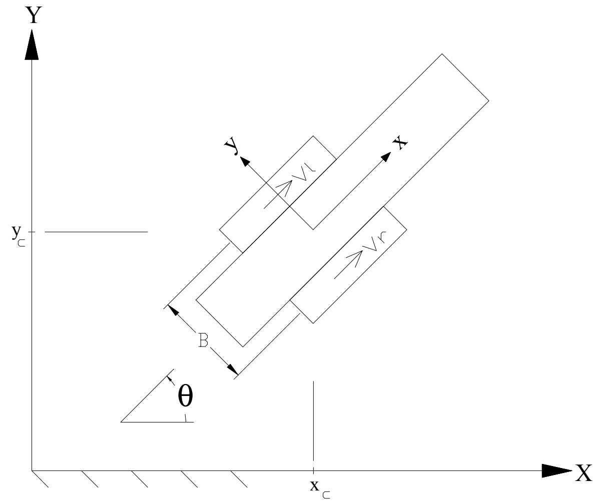


Figure 3.1 A mobile robot model.

Workspace Model. Knowledge about the robot workspace is assumed to be partially known. A two-dimensional map representing approximately the locations of obstacles and free space in the workspace is given a priori. Although this map does not necessarily match the real workspace exactly, the discrepancy between the map and the real workspace must be within some limit and cannot be too large. Otherwise, any knowledge we gain from off-line path planning will be useless when the robot plans the path itself on-line.

The two-dimensional map comprises segments of straight lines forming polygons and boundaries. The segments of straight lines are represented by a set of linear equations and their end points in the global coordinate frame *{XY}*, as shown in (3.3) below.

$$\begin{aligned}&Y = m_i X + c_i\\&s_i = (Xs_i, Ys_i)\\&e_i = (Xe_i, Ye_i)\\&i \in [1, N]\end{aligned} \tag{3.3}$$

where m_i = slope

c_i = Y-axis intercept

s_i = start point

e_i = end point

i = index number

N = the total number of lines

A line parallel to the Y-axis will be represented by (3.4) with its end points.

$$x = d_i \tag{3.4}$$

We choose to represent the workspace using these explicit algebraic forms of linear equations since they are practical and require minimal computer storage (need to store only their end points). The reason for using linear segments to approximate a robot workspace is that all of the range measuring sensors (e.g. Laser Rangefinder, Radar, Sonar, etc.) used in today mobile robots return only discrete range-data points. Moreover, the level of accuracy afforded by nonlinear segment models lies well-within the uncertainty of the measured data points. Therefore, it is reasonable to estimate range between a pair of data points with linear approximation to cope with the discretization in the sensors. Additionally, collision checking between the mobile robot and the workspace can be done effectively and quickly. An example of a two-dimension workspace map is shown in Figure 3.2.

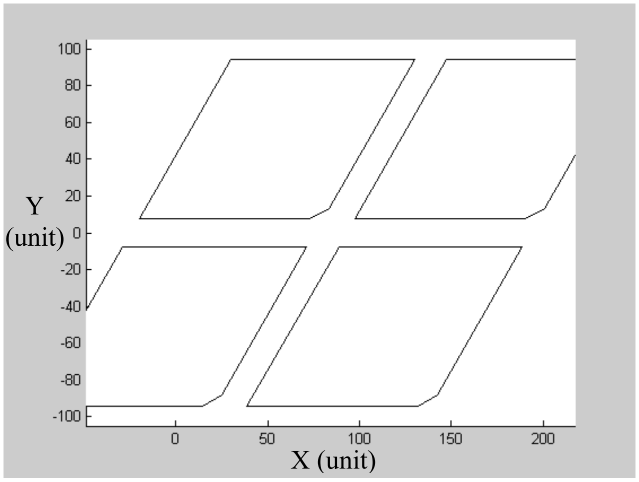


Figure 3.2 A two-dimensional workspace map.

3.2 Uncertainties in Robot Systems

The primary forms of uncertainties entering robot systems can be categorized into three types: model, sensory, and control uncertainties. Each of uncertainties comes from different sources. Model uncertainty appears due to un-modeled dynamics and inaccurate parameters of the robot system. One must admit that there is no mathematical model that can perfectly represent a real physical system. Thus, any discrepancy between the physical system and the mathematical model causes the model uncertainty. The error between the system parameters used in the mathematical model and the ones presented in the physical system also contributes to the model uncertainty since it is almost impossible to obtain all system parameters correctly. Besides, some parameters might even vary over time.

Sensory uncertainty happens from the fact that there is no perfect sensor, which can measure any physical quantities exactly. Therefore, sensory data obtained from sensors always have more or less built-in uncertainties, also commonly known as noises. Control uncertainty can also be clearly seen from the objective of control itself. The discrepancy between a desired output and a real output of a control system gives rise to the control uncertainty. The degree of these two uncertainties depends on the robot systems and their environment condition. In harsh working conditions, the robots are more susceptible to uncertainties from the environments. Uneven terrain, muddy ground floor, dusty air, etc., can introduce uncertainties into the robot systems. Moreover, the robots also face uncertainties from within the robot systems themselves. Friction, backlash, etc., can also introduce uncertainties into the robot systems.

In this research, the kinematic model (3.1) is used to describe the dynamics of the robot system. This kinematic model posits a constraint that the robot can only move in the direction tangent to its trajectory, but cannot move sideways. This constraint seems to hold at all time for every mobile robot moving and cornering at low speed. Therefore, the model uncertainty is assumed to be zero, and only uncertainties from sensing and control are considered in this study.

Degrees of sensory and control uncertainties vary from one robot system to another. The degrees vary widely and depend heavily on types of sensors, actuators, and control algorithms used in the robot systems. The degrees of sensory and control uncertainties for each robot system are treated as two separated numbers, where the sensory uncertainty may be represented in the form of uncertainty ellipsoid and the control uncertainty may be represented in the form of a percentage of steady-state error. Each number may vary for different configurations or remain constant for the entire configuration space; however, its value must be bounded. The degrees of uncertainties can be predetermined either by means of experiments or by means of simulations. To generalize and simplify the analysis, the degrees of uncertainties from both sensory and control are combined and represented by a single number. Although the sensory and control uncertainties may have different dimensions, the combining of the two uncertainties can be accomplished through the use of a feedback control law of a robot. Let assume that the feedback control *u* in (3.1) is in the following form (assuming that all states are available):

$$u = f(x_c, y_c, \theta). \tag{3.5}$$

Let the estimated state from the sensor measurement be $x_e = [x_c + \delta x \quad y_c + \delta y \quad \theta + \delta\theta]^T$, where δx, δy, and $\delta\theta$ are sensing errors in x_c, y_c, and θ due to sensory uncertainty. Thus, the feedback control (3.5) with the estimated state becomes:

$$u + \Delta u = f(x_c + \delta x, \quad y_c + \delta y, \quad \theta + \delta\theta) \tag{3.6}$$

where, Δu is the feedback control error due to sensory uncertainty.

In addition, the feedback control (3.6) is also corrupted by the control error Δc due to control uncertainty. This control error is a scalar value and generally expressed in terms of a fraction of steady-state error to the desired set point. Thus, the actual feedback control entering the system (3.1) is:

$$(1 + \Delta c) \bullet (u + \Delta u) = u + [\Delta c \bullet (u + \Delta u) + \Delta u] \tag{3.7}$$

Notice that the expression in the closed bracket in the right hand side of equation (3.7) is an overall error in feedback control due to both sensory and control uncertainties. Clearly, the overall error in feedback control is a function of the feedback control u itself—both explicitly shown as u and inherently embedded in Δu. This means that if the feedback control u and the uncertainties (δx, δy, $\delta\theta$, and Δc) are bounded, the overall error in feedback control is also bounded. Consequently, the actual feedback control (3.7) is also bounded. From equation (3.7), the overall error in feedback control (or the overall control error) can be rewritten in the form of (3.8) by factoring out u from Δu and summing all coefficients of u together.

$$u + [\Delta c \bullet (u + \Delta u) + \Delta u] = u + \Delta_{err} \bullet u \tag{3.8}$$

where, Δ_{err} is the fraction of the overall control error to the feedback control u.

In this form, the overall control error in equation (3.8) can be represented by the multiplication of the feedback control u with a single scalar value, Δ_{err}, which we call "Coefficient of Overall Control Uncertainty", or, in short, "Overall Control Uncertainty". This scalar value is a result of lumping together the consequences of both sensory and control uncertainties to feedback control. The overall control uncertainty conveniently and effectively represents the effect of both uncertainties as a whole to the closed-loop control of a mobile robot system. The value of the overall control uncertainty is zero when there are no uncertainties, and it has a finite positive real value when there are bounded uncertainties presented in the robot system. It is clear that the value of the overall control uncertainty also depends on the choice of feedback control (3.5); however, the discussion about choosing the most suitable (or optimal) feedback controller lies outside the scope of this dissertation and will be omitted.

3.3 Linear Control Uncertainty Field

In the context of robot motion planning, the concept of overall control uncertainty discussed in the previous section serves as a fundamental idea in the development of a novel motion planning algorithm for nonholonomic mobile robots under control and sensing uncertainties. We have learned that the degree of uncertainties can be represented through a single number, the overall control uncertainty, and also learned that this number depends on the choice of feedback control given the same degree of uncertainties. This may result in different paths planned for different controllers selected. However, an optimal path for a given robot and environment should be invariant regardless of the choice of a robot feedback control. Therefore, by slightly modifying the concept of overall control uncertainty, the notion of Linear Control Uncertainty (LCU) is developed. The main idea is to determine (at each configuration in the environment) the maximum allowable overall control uncertainty instead and to use this number in defining the optimal path. This way the optimal path stays unchanged with respect to different feedback control schemes since the value of LCU does not depend on types of control. The LCU indicates the degree of combined uncertainties—both sensory and control uncertainties—allowed at any particular configuration in the robot workspace, such that the mobile robot can move from the current configuration for one time period without bumping into obstacles in the environment. Before we give a formal definition of the LCU, some of the basic definitions used in robot motion planning will be introduced first [Latombe, 1991].

Definition 3.1 A **robot** $\mathcal{A}$ is a rigid object described as a compact (i.e. closed and bounded) subset of a Euclidean space $\mathcal{W}$, called **workspace**, represented as R^N, with N = 2 or 3. **Obstacles** $\mathcal{B}_i$, where i = 1, 2, …, N, are closed subsets of $\mathcal{W}$.

Definition 3.2 A **configuration** q of a robot $\mathcal{A}$ is a specification of the position and orientation of the robot-fixed coordinate frame *{xy}* with respect to the global coordinate frame *{XY}*. The **configuration space** of $\mathcal{A}$ is the space $\mathcal{C}$ of all the possible configurations of $\mathcal{A}$. The subset of $\mathcal{W}$ occupied by $\mathcal{A}$ at configuration q is denoted by $\mathcal{A}(q)$. A set of the interior points of set $\mathcal{S}$ is called the *interior* of $\mathcal{S}$, and denoted by $int(\mathcal{S})$.

Definition 3.3 The obstacle $\mathcal{B}_i$ in $\mathcal{W}$ maps in $\mathcal{C}$ to the region $\mathcal{CB}_i = \{ q\in \mathcal{C} \mid \mathcal{A}(q) \cap \mathcal{B}_i \neq 0 \}$. $\mathcal{CB}_i$ is called a **C-obstacle**.

Definition 3.4 **Free space** is the subset of $\mathcal{C}$ defined by $\mathcal{C}_{free} = \mathcal{C} - \bigcup_{i=1}^{N} \mathcal{CB}_i$.

Definition 3.5 A **contact space** is the subset of $\mathcal{C}$ made of configurations at which $\mathcal{A}$ touches one or several obstacles without overlapping any, or mathematically defined as:

$$\mathcal{C}_{contact} = \{q\in \mathcal{C} \mid \mathcal{A}(q)\cap \bigcup_{i=1}^{N} \mathcal{B}_i \neq 0 \text{ and } int(\mathcal{A}(q)) \cap \bigcup_{i=1}^{N} int(\mathcal{B}_i) = 0 \}.$$

Valid space is: $\mathcal{C}_{valid} = \mathcal{C}_{free} \cup \mathcal{C}_{contact}$.

We are now ready to state the definition of the Linear Control Uncertainty (LCU).

Definition 3.6 Let T be a constant time period, and $T > 0$. Given a robot $\mathcal{A}$ moving in a workspace $\mathcal{W}\in R^2$ containing obstacles $\mathcal{B}_i$, the **Linear Control Uncertainty** (LCU) of a configuration $q\in \mathcal{C}_{free}$ is a nonnegative real number that perturbs a constant nominal control V (where $V\in R$) for the actual controls v_r and v_l in (3.2), as defined by the following equations

$$\begin{aligned} v_r &= (1+LCU)\cdot V, \quad v_l = (1+LCU)\cdot V \\ v_r &= (1-LCU)\cdot V, \quad v_l = (1-LCU)\cdot V \\ v_r &= (1+LCU)\cdot V, \quad v_l = (1-LCU)\cdot V \\ v_r &= (1-LCU)\cdot V, \quad v_l = (1+LCU)\cdot V \end{aligned}, \tag{3.9}$$

such that the trajectory of system (3.1) starts from $x(0) = q_0 = q \in \mathcal{C}_{free}$ and ends at $x(T) = q_T \in \mathcal{C}_{free}$ with $x(T^+) = q_{T^+} \in \mathcal{C}_{contact}$, where $T^+ \in (T, \infty)$.

From the definition of LCU, there are two free variables. One is the constant nominal control *V*, and the other is the constant time period *T*. These two variables can affect the value of LCU of a free configuration in a given workspace. For example, if one decreases *V* and *T*, the value of LCU will increase, or vice versa. Although the selection of these two variables affects the value of LCU for all free configurations, the same effect applies to all free configurations. Therefore, the topology of the Linear Control Uncertainty Field, which will be discussed next, of a free configuration space are invariant to the choices of *V* and *T*. This means that the configuration possessing the higher LCU than its neighbors always possesses the higher LCU than its neighbors for different sets of *V* and *T*. In general, we choose *V* to be the maximum or minimum velocity of the robot wheel, and choose *T* to be a sampling period.

The reason to let *V* be the maximum or minimum velocity is to establish a lower bound for the value of LCU for any given *T*, so that the maximum allowable error in the closed-loop control system can be stated. The reason to let *T* be a sampling period is that for many practical control schemes, especially digital ones, the value of the control input to dynamic systems is held constant for one sampling period such as a zero-order hold in digital control [Franklin, et al., 1997]. Further notice that the nominal control *V* is the same in both equations of v_r and v_l in (3.9), this is because, for determining LCU, the nominal trajectory or path of system (3.1) is chosen to be a short straight line. Also notice that the LCU is assumed to be the same for both v_r and v_l , this makes sense because both v_r and v_l, at any instant, should experience the same magnitude of perturbation, but can be different in directions, shown as plus and minus signs in (3.9). Intuitively, the LCU, for an arbitrary configuration, with above suggested choices of *V* and

T can be thought of as the maximum allowable degree of perturbation to the nominal straight path, such that a robot can move from that arbitrary configuration along the perturbed path for one sampling period in a workspace without touching obstacles in the workspace.

The value of LCU at a particular configuration can be determined by two means: simulation and approximation (which will be later described in Chapter 5). Given a robot, a robot workspace, obstacles, *V*, and *T*, the simulation iterates for each particular configuration by increasing LCU (starting from zero), calculating perturbed controls from equation (3.9), integrating system equation (3.1) forward in time for *T* seconds, and checking collision between a robot and obstacles. If there is no collision, the LCU is increased and another iteration is carried out. The simulation terminates when collision occurs, and the LCU just before the collision is declared as the LCU of that particular configuration. A graphical illustration for the determination of the LCU of a point robot, represented as a point and an arrow to show orientation, at the configuration (0,0,0) is shown in Figure 3.3. Notice that the nominal path in Figure 3.3(a) spreads wider as the value of LCU increases. The simulation terminates at LCU = 0.3 as shown in Figure 3.3(d), and 0.2 is the value of LCU at configuration (0,0,0). In case of sized robots, a more involved collision detection algorithm is employed to detect the interference during the entire simulation.

Two types of collision detection algorithms have been implemented in this research. One is based on a direct implementation of linear algebra by determining all possible intersections between each pair of the edges of the robot and obstacles. The other is based on the "Separating Axis Theorem", of which detail is given in Appendix A. The linear algebra-based algorithm uses less computation time than the separating-axis-theorem-based algorithm does with a small collision detection problem—small numbers of obstacles in the problem. However, from our experiments, as the numbers of obstacles in the problem grows, the separating-axis-theorem-based algorithm becomes more computationally efficient than the linear algebra-based algorithm does. Therefore, we normally compare the run time for each collision detection algorithm to determine which algorithm will be used in a given problem.

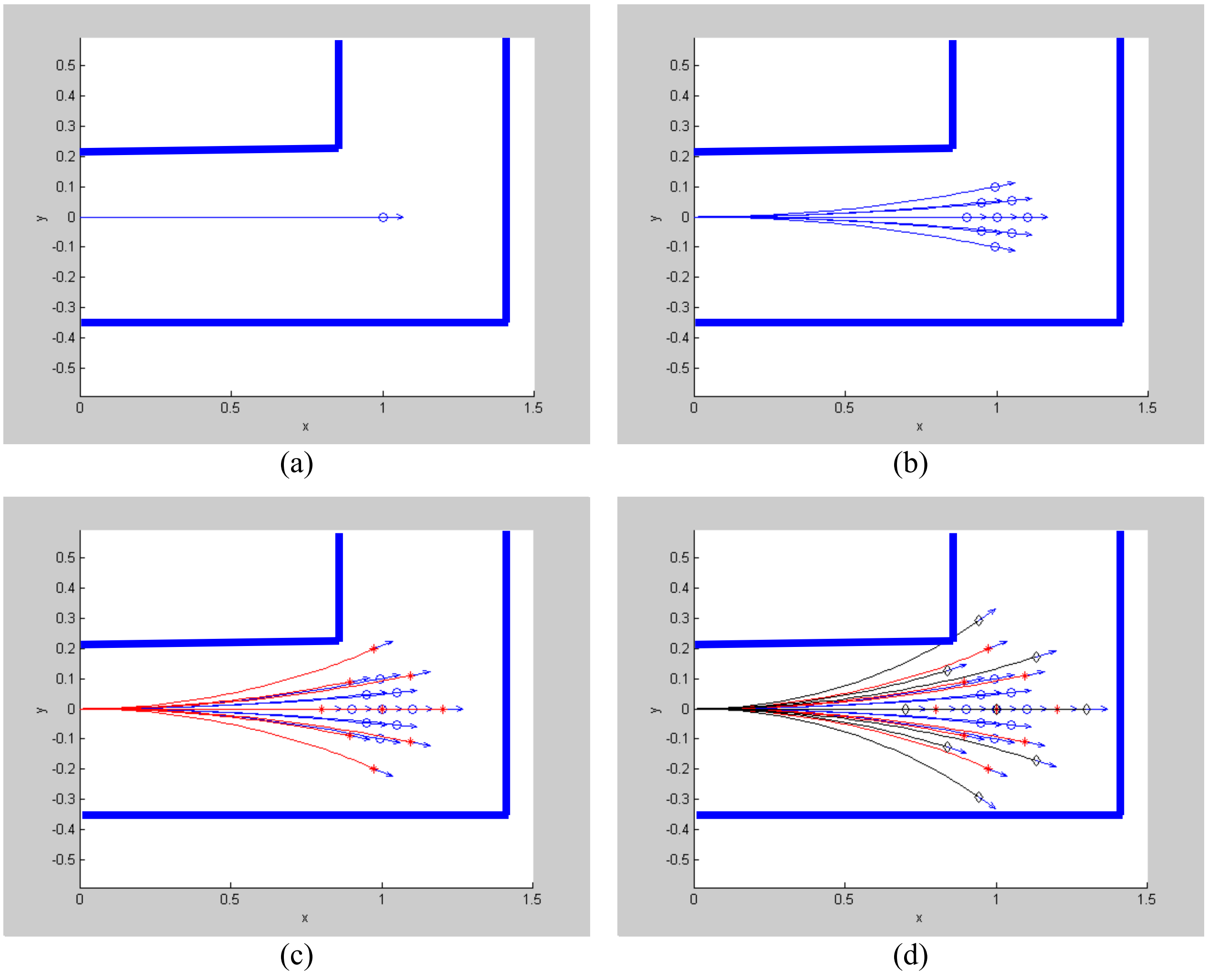


Figure 3.3 A sequence of the simulation for determining the LCU of configuration (0,0,0) of a point mobile robot, (a) a nominal path, (b) perturbed paths with LCU = 0.1, (c) perturbed paths with LCU = 0.2, and (d) perturbed paths with LCU = 0.3.

Next, the concept of LCU is directly applied to the motion planning problem of nonholonomic mobile robots. The idea is to find a path that connects between given initial and goal configurations, while all of the configurations on the path possess high values of LCUs. To be able to determine the LCUs along a path, the path must be approximated by a series of discrete configurations and their nominal straight paths as shown in Figure 3.4. Points (positions) and arrows (orientations and nominal paths) indicate the discrete robot configurations (x_c, y_c, θ) and their nominal paths along a continuous path.

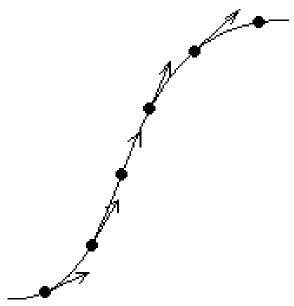


Figure 3.4 A path is approximated by a series of discrete configurations.

Before searching for an optimal path in a discrete three-dimensional configuration space, the LCUs at each discrete configuration q in a free space $\mathcal{C}_{free}$ must be computed first to create a field called "Linear Control Uncertainty Field (LCUF)", and the search can be conducted afterward. The definition of this field is as follows:

Definition 3.7 Given a robot $\mathcal{A}$ moving in a workspace $\mathcal{W} \in R^2$ containing obstacles $\mathcal{B}_i$, a **Linear Control Uncertainty Field** (LCUF) is a field of LCU for all $q \in \mathcal{C}_{free}$.

An example of a LCUF is presented in Figure 3.5. The figure depicts a graphical representation of a LCUF of a rectangular mobile robot (shown on the top of the figure) moving in a given workspace, a 90-degree turn passage. The magnitudes of the vectors in the figure show values of LCU at the points from which the vectors emanate. The directions of the vectors represent heading angles or orientations of the mobile robot at those points. One can notice that the value of LCU is highly sensitive to changes in the orientations of mobile robot as the magnitudes of the vectors at the same point vary greatly when there are small changes in orientations. This characteristic of the LCU poses a difficulty in searching for an optimal path through the LCUF since the values of a cost function of slightly different candidate paths can be considerably different, which gives rise to a highly nonlinear optimization problem. Nevertheless, the method used in the next section can cope with this difficulty.

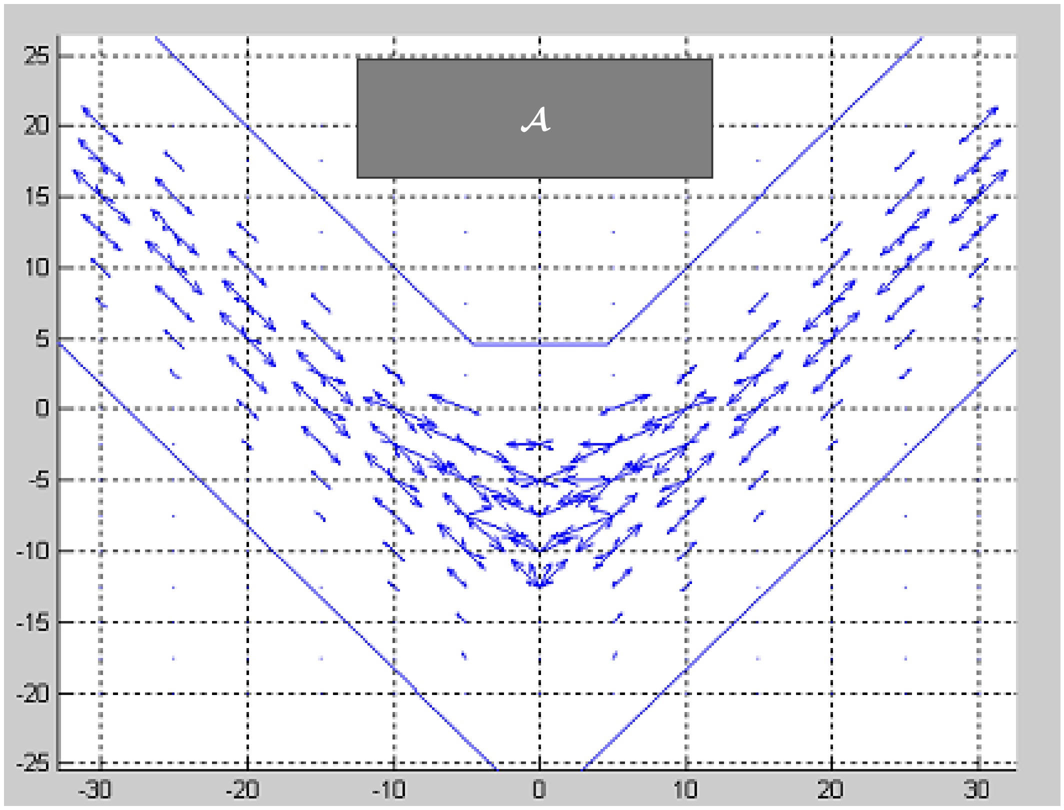


Figure 3.5 A Linear Control Uncertainty Field (LCUF).

In case of a holonomic mobile robot such as an omnidirectional robot, the search for a path in a given LCUF can be easily performed since every valid path in a valid space $\mathcal{C}_{valid}$ is a feasible path for a holonomic mobile robot. The difficulty arises when one deals with a nonholonomic mobile robot because not every valid path is a feasible one. By rearranging equation (3.1), we obtain the following nonholonomic constraint:

$$-\dot{x}_c \sin\theta + \dot{y}_c \cos\theta = 0 \tag{3.10}$$

The constraint restricts a feasible path to only the one that is tangent to the orientation of the mobile robot at that instant, and, therefore, makes a direct search for a path in a LCUF nearly intractable. For this reason, we propose a better alternative to the direct search in a LCUF by applying the method proposed in the next section to solve this search problem instead. To proceed carefully in determining an optimal path for nonholonomic mobile robots using the proposed method, more definitions of valid path, feasible path, and optimal path are needed.

Definition 3.8 A path is said to be **valid** if it is a path between an initial configuration q_{init} and a goal configuration q_{goal} in $\mathcal{C}_{valid}$ is a continuous map $\tau : [0,1] \rightarrow \mathcal{C}_{valid}$, with $\tau(0) = q_{init}$ and $\tau(1) = q_{goal}$.

Definition 3.9 A path is said to be **feasible** if it is valid, and respects the nonholonomic constraint (3.10).

Definition 3.10 A path is said to be **optimal** if it is feasible, and minimizes a performance index (cost function).

From above definitions, an optimal path must respect the nonholonomic constraint and at the same time minimizes a given cost function. If a cost function is specified, one can determine an optimal path using the method introduced in the next section.

3.4 Optimal Path Searching Method

For an optimal path, the system dynamic model (3.1) must be satisfied at all time. Additionally, the optimal path—connecting between an initial configuration and a goal configuration—of the system must minimize a given cost function. In this setting, the nonholonomic motion planning problem can be converted into an optimal control problem. When one determines an optimal control, v^* and ω^*, that steer the system (3.1) from a given initial state to a given terminal state such that a given cost function is minimized. This way it is guaranteed that the resultant path (or trajectory) always satisfy the nonholonomic constraint.

There are many techniques used in solving optimal control problems both analytically and numerically. However, most practical problems in optimal control are too complex to solve analytically; as a result, numerical methods are more applicable. One of the popular techniques used in optimal control problems is to transform an optimal control problem into a standard optimization problem, and then use well-established tools in optimization theory to solve the optimal control problem. This technique uses approximation-and-optimization approach, where it approximates the control with piecewise constant functions with a fixed or variable number of fixed time intervals. In each interval, the value of control can take on different constant values within limits since the control is bounded. With a given initial guess, the technique systematically optimizes the approximated control by altering the approximated control according to the information obtained from integrating the system model and evaluating the cost function. The optimization process keeps repeating until the cost function is not much further improved or the specified number of iterations is reached. The optimization result is a close approximation of the optimal control. This approximation-and-optimization approach is widely accepted, and also used in commercial codes such as POST (Program to Optimize Simulated Trajectories) written by Lockheed Martin Astronautics and NASA [Brauer, et al., 1977]. Figure 3.6 illustrates the concept of the approximation-and-optimization approach.

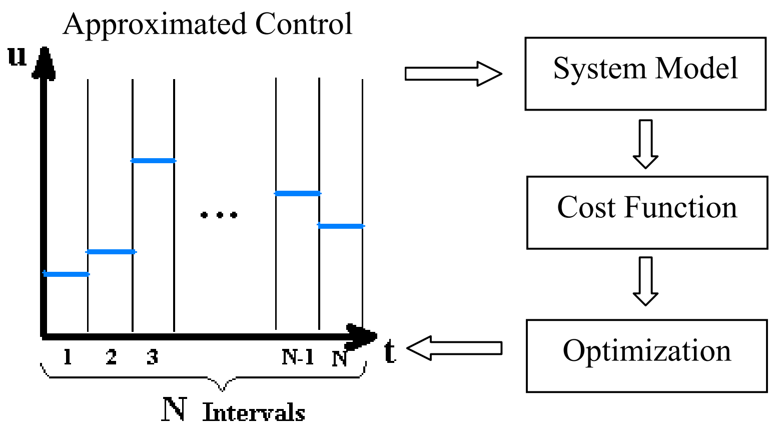


Figure 3.6 The approximation-and-optimization approach.

The success of the approximation-and-optimization approach depends heavily on the optimization method deployed in the algorithm. The type and characteristic of cost function, including the constraints on the variables, play major role in selecting a suitable optimization method. Thus, the cost function and constraints must be identified first.

A motion planning problem is a constrained optimization problem by nature due to constraints in both the state of the system (x_c, y_c, and θ) and the control of the system (v_r and v_l). The constraint of the state comes directly from obstacles in a workspace, while the constraint of the control arises from the fact that the control is bounded. To convert the constrained optimization problem to an unconstrained one, we utilize the concept of penalty functions, where we penalize the state and control constraint violation (represented by terms *col* and $\beta(u)$, respectively, in cost function (3.11)) by making the cost function a large positive real number compared to the non-violation cost function, which is usually a small positive real number. Thus, the cost function for the control that exceeds the control bound or leads the mobile robot to a collision will have much larger in magnitude than the one that stays inside the control bound or causes no collision. In addition, to make sure that the final configuration of the mobile robot terminates at the goal configuration, the cost function also includes a quadratic function of the discrepancy between the final and goal configurations, $(x_N - x_g)^2$. Ultimately, the most desired path should comprise configurations possessing the greatest LCUs. With all these quantities, the cost function can be explicitly written as follows:

$$F(u) = -\min LCU(x_i) + \{\ \alpha(x_N - x_g)^2 + \beta(u) + col\ \}, \qquad i \in [1,2,...,N] \qquad (3.11)$$

where, $LCU(x_i)$ = LCU at configuration.

α = constant positive weight.

x_N = final configuration of the robot.

x_g = goal configuration of the robot.

$\beta(u) = 0$ if no control constraint violation occurs.

$= 1000 \cdot (\|u\|_\infty)^2$ if control violation occurs

col = 0 if no collision occurs.

$= 1000 \cdot \min LCU(x_i)$ if collision occurs.

The cost function F is a function of control u, which can be easily transformed into a series of configurations x_i–state x_i and configuration q_i are equivalent and can be used interchangeably–by simulating the system (3.1) to impose the nonholonomic constraint. The

LCU of each x_i is varied along the path produced by the control u. The minimum one, *min LCU(x_i)*, is selected and used in the cost function as a minimum LCU of that path, *min LCU*. In this way, we can guarantee that the mobile robot with control uncertainty lower than this minimum value is able to traverse along the path safely so long as the mobile robot follows the path accurately—the proof of this statement will be provided in the next section. The weight, α, is used for controlling the characteristic of the end point x_N of an optimal path, for example, one can set α with a large positive number to find the optimal path which ends closer to the given goal configuration x_g. However, α must be carefully chosen such that the term $\alpha(x_N - x_g)^2$ will not dominate the term *min LCU(x_i)*.

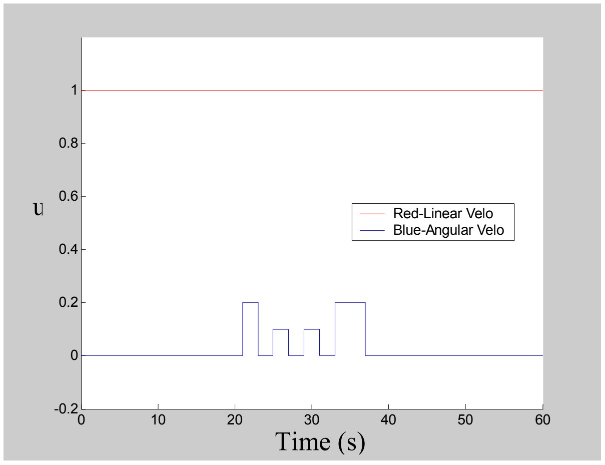
As all variable constraints in the constrained optimization problem are imbedded into the cost function in the form of penalty functions, the constrained optimization problem has been changed to the unconstrained one. However, using the penalty functions such as *col* and $\beta(u)$ in (3.11) introduces discontinuity to the cost function. The optimization schemes that can handle discontinuous or highly nonlinear cost functions efficiently are direct search methods. One of the well-known direct search methods is the Nelder-Mead simplex method, whose detail algorithm is provided in Chapter 2, and it is a popular direct search method for multidimensional unconstrained nonlinear minimization problem having non-smooth cost function. To this end, the Nelder-Mead simplex method is selected as the preferred optimization method.

The approximation-and-optimization approach above, including the cost function (3.11) and the Nelder-Mead simplex method, has been implemented in a popular engineering programming language, MATLAB®. The developed search program incorporates a built-in command in MATLAB® called "fminsearch", which is an optimization command based on Nelder-Mead simplex method. The main structure of the search program has been adopted from the existing optimization code written by Dr. Eugene Cliff. The search program, listed in Appendix B, is applied to search for an optimal path for the rectangle robot in the workspace as shown in Figure 3.5 with given initial and goal configurations. The result is given in the next section.

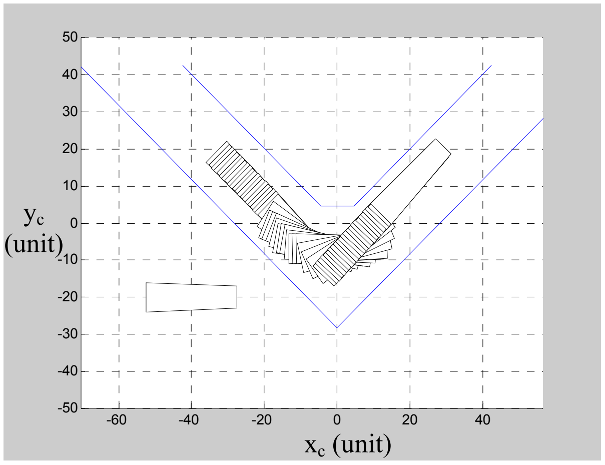
3.5 Results and Discussions

The rectangular robot and the workspace, as shown in Figure 3.5, are chosen to be a case study. The sampling period T and the maximum wheel velocity V of the robot for the LCU calculation are 0.5 second and 1 unit/second, respectively. The goal is to find an optimal path that minimizes the cost function in (3.11) with $\alpha = 0.01$. The controls, v and ω, are bounded within [-1,1] unit/second and [-1,1] rad/second, respectively. Both control are approximated by two piecewise constant functions, and each one is uniformly divided into 60 equal intervals, $N = 60$. Hence, we have a total of 120 variables to optimize. The final time is specified to be 60 seconds, $T_f = 60$. Given initial and goal configurations are $[-25,11,-\pi /4]^T$ and $[25,11,\pi/4]^T$, respectively. The initial guess for the control and the corresponding initial motion are presented in Figures 3.7(a) and 3.7(b), respectively.

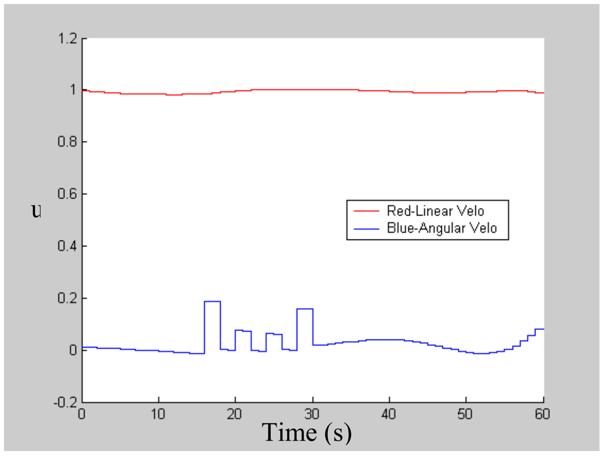
The search program ran on a personal computer with a 1.7 GHz processor, took almost 96 hours to search for the optimal path. Figures 3.7(c) and (d) present the search results: the optimal control and the corresponding optimal motion, respectively. From Figure 3.7(d), the mobile robot starts from the left side at the initial configuration, $[-25,11,-\pi /4]^T$, and stops close to the goal configuration at $[24.47, 10.36, \pi/4.08]^T$. In Figure 3.7(c), the optimal linear velocity (the top curve) stays at full forward, +1 unit/second, most of the time while the optimal angular velocity (the bottom curve) constantly changes within [-0.1,0.2] rad/second. The reason why the linear velocity stays almost constant at +1 is that the initial and goal configurations are set far apart from each other. This makes the mobile robot attempt to move from the initial configuration toward the goal configuration as fast as possible; as a result, the optimal linear velocity is mostly maintained at +1. Notice that the optimal angular velocity curve is not smooth, and, as a result, it yields the non-smooth optimal path as shown in Figure 3.8(a). This is because the discretization of time intervals is coarse—at 1 second/interval in this case study—to limit the search time to be within a reasonable period.



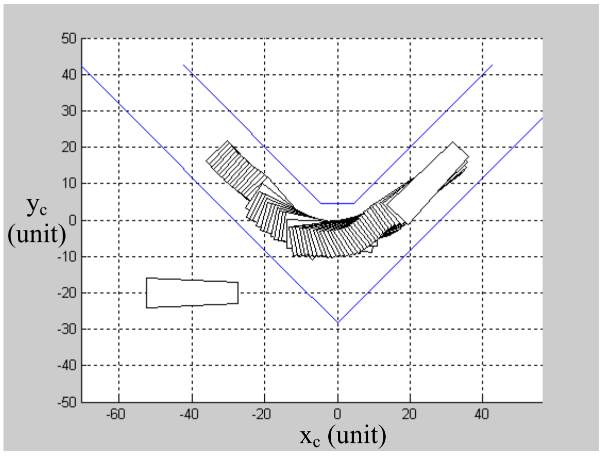
(a)



(b)



(c)

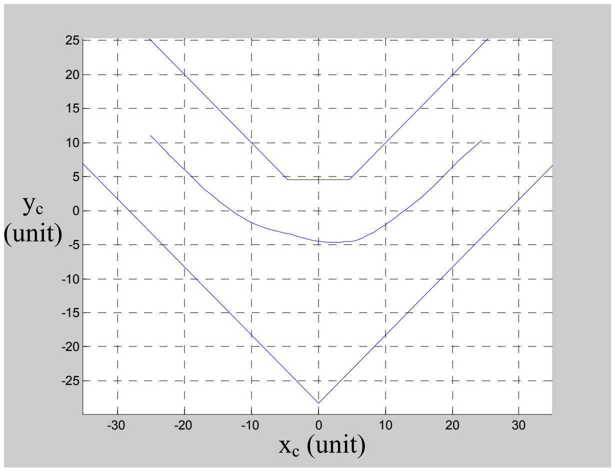


(d)

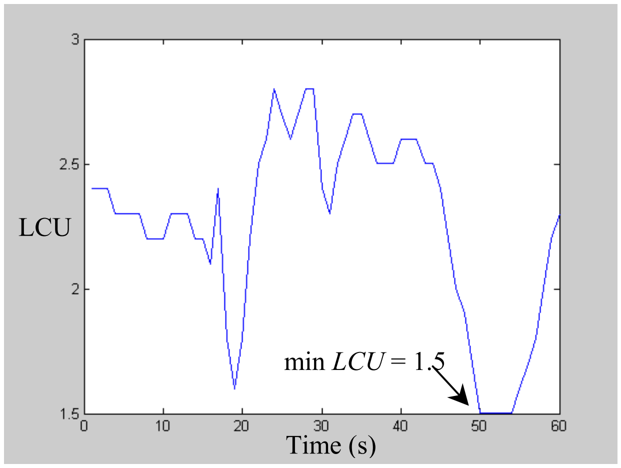
Figure 3.7 The initial guess and the search results, (a) initial control, (b) initial motion, (c) optimal control, and (d) optimal motion. Note: the trapezoidal robot is used for illustration propose only, to indicate the heading of the robot.

Also notice that the optimal path in Figure 3.8(a) is not anywhere near the medial-axis of the 90-degree passage suggested by the geometric planners. We see that hugging the corner makes sense since the robot is an elongated rectangle and that collisions are most likely to occur at the robot vertices. Figure 3.8(b) depicts the values of control uncertainties for every discrete configuration (x_i, where i = 1, 2, …, N) along the optimal path. The minimum LCU, *min LCU*, of the path is 1.5 or 150%, which means that the robot must have overall control uncertainty in its feedback control approximately less than or equal to 150% to traverse along the path safely with a linear velocity less than 1 unit/second, provided that the robot follows the path accurately.

In the next subsections, we will prove that the path obtained from the search method is indeed a global optimum, and also prove that, in some certain conditions, if the mobile robot stays on the optimal path, travel with a linear velocity less than specified limit, and has the overall control uncertainty in its feedback control below the minimum LCU of the optimal path, the robot is guaranteed not to collide with any obstacles in its workspace.



(a)



(b)

Figure 3.8 (a) Optimal path, (b) Linear control uncertainties along the optimal path.

Optimality of the path. We will prove that the path obtained from the search program as shown in Figure 3.8(a) is a global optimal path in the given workspace, the 90-degree passage. The sketch of the constructive proof for the optimality of the path is given below.

From the cost function (3.11), the search program tries to minimize the cost function by minimizing all the terms in the curly bracket (penalty functions) while maximizing the minimum LCU term. To keep the proof simple, we make all penalty functions go to zero by letting the final configuration be equal to the goal configuration (meaning that the path is now a free end-point path) and assuming that there is no state and control constraint violations (meaning that there is at least one feasible path in the workspace). Therefore, the cost function now depends only on the value of the minimum LCU, and any path that maximizes the minimum LCU is the optimal path. The cost function now is reduced to:

$$F(u) = -\min LCU(x_i), \qquad i \in [1, 2, ..., N] \tag{3.12}$$

Since the Nelder-Mead simplex method—in fact, most optimization methods—determines its search direction using local information on how the cost function changes, there is a possibility that the method will converge at local minima and never leaves. If we can prove that the LCUF of the passage has only one global maximal contour, then the path in Figure 3.8(a) is a global optimal path. This proof can be easily concluded from considering the contour and quiver plots of the LCUF of the passage as presented in Figure 3.9 and 3.10, respectively.

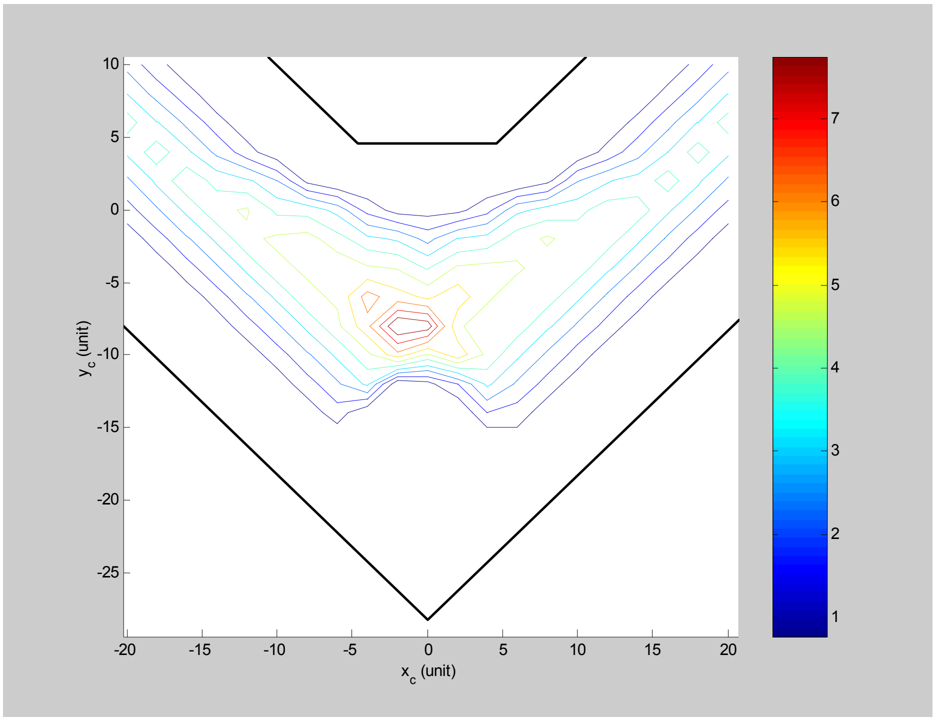


Figure 3.9 Contour plot of the LCUF.

The two-dimensional contour plot of a three-dimensional LCUF, *LCUF*(x_c, y_c, θ), is obtained by let $LCUF(x_c, y_c) = \max_{\theta \in (-\pi/2, \pi/2]} LCUF(x_c, y_c, \theta)$, which means that we consider only the maximum of LCUs at any (x_c, y_c) with the orientation θ between -π/2 to π/2, in order to restrain the mobile robot to start moving from left to right only. From the contour plot, it is clear that the LCUF has a valley-like shape, with its dip located near workspace boundary and its ridge near the middle of the passage. The ridge shows no sign of bifurcation (split), and, therefore, the optimal solution is unique and must lie on top of the LCUF ridge. This is also confirmed by the quiver plot in Figure 3.10, where the thick arrows indicate the maximum LCUs along the path, and clearly suggests where the optimal path should lie. Therefore, we can conclude that there is only one global maximal contour lying in the LCUF of the passage, and the path obtained from the search program in Figure 3.8(a) can be concluded to be global optimal. A more rigorous proof will be given in Chapter 5, where the proof is based on the characteristic of the LCUF.

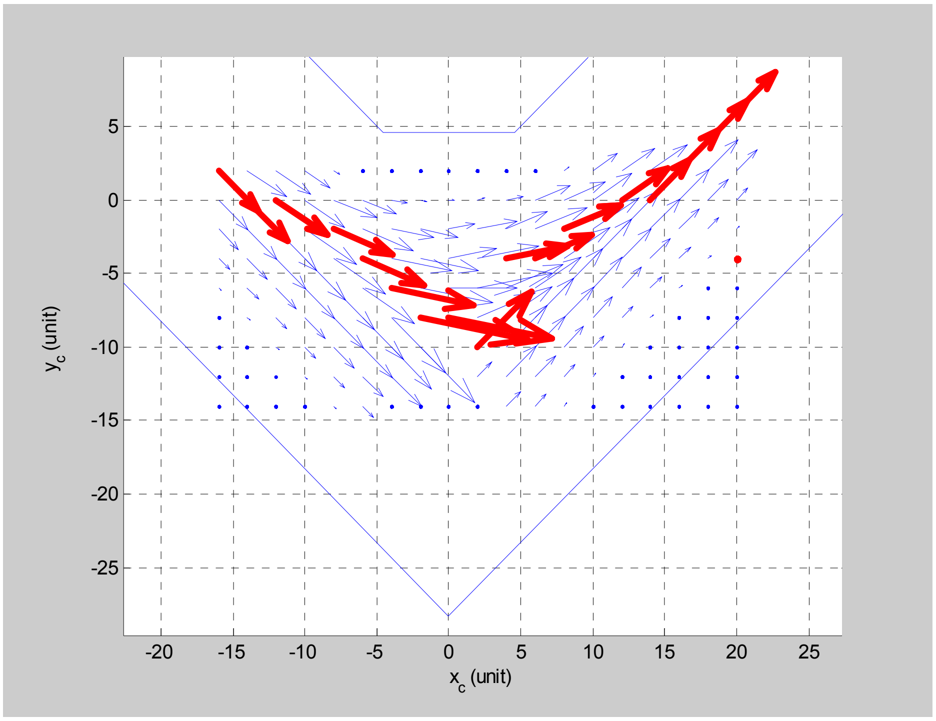


Figure 3.10 Quiver plot of the LCUF.

Comparing the path suggested by the quiver plot (by roughly connecting the thick arrows together) and the path obtained from the search program, one can clearly see that it is almost identical. This is because the path in Figure 3.8(a) is constrained to pass both initial and goal configurations, which is selected to be on the global maximal contour. Therefore, the cost function (3.11) of the path is essentially reduced to equation (3.12), which is the same. One might be able to get a sub-optimal path by tracing along the red arrows in Figure 3.10. However, a path forming by simply connecting each thick arrow together might not be a feasible path for the robot to traverse; as a result, the proposed search method is necessary in determining the optimal path. It is also worth noting that the contour plot is not symmetric, which means that the degrees of LCUs are different in the region where the robot approaches the turn and in the region where the robot leaves the turn. This happens because while approaching the turn the robot moves from narrower space to wider space, and, thus, the LCUs of the approaching region ($x_c < 0$) are higher than that of the leaving region ($x_c > 0$).

Estimation of safe control uncertainty. In this section, we will prove that the minimum LCU of the optimal path can be used to approximate maximum control uncertainty level that the mobile robot can have such that the likelihood of collision is eliminated when the robot traverses the path with a specified linear velocity limit. The proof for a straight optimal path will be discussed first, and the proof for a free-form optimal path will be discussed next.

Let us consider a straight path in Figure 3.11. The workspace comprises two different width straight passages forming a bottleneck. Intuitively, an optimal path is a medial-axis pass through both straight passages, shown as a dash line in Figure 3.11. For simple illustration, a point robot (with sampling period *T* second) is used, and is traveling with a linear velocity *V* unit/second in the workspace from left to right, points A to C. At each point, a fan-shaped geometry represents possible locations of the point robot–the geometry represents the swept-area instead if the robot is a sized one—corresponding to the value of LCU as defined in Definition 3.6.

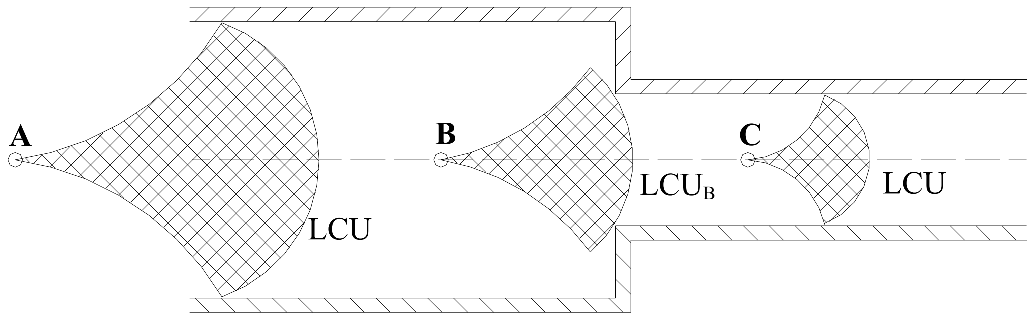


Figure 3.11 A straight optimal path in a bottleneck passage.

It is clear that the LCUs along the path stay constant in section AB (from point A to point B), strictly decrease in section BC (from point B to point C), and stay constant again in the last section (after point C on). At each point along the path, the robot cannot possess control error more than what is specified by the LCU at that point because, according to the definition of the LCU, any point has the maximum possible LCU. Otherwise, the collision will occur, which translates to the intersection between a larger fan-shaped geometry (a larger LCU) and the workspace.

In similar fashion, the robot cannot deviate from the optimal path if the robot has control error which exactly equals the LCU at that point because if one tries to move point A, B, or C up or down even slightly, the collision will occur. Moreover, since the values of LCUs vary along the path, to ensure safe path following for the entire path, the robot must possess the maximum control uncertainty in its system that is equal to or less than the minimum LCU of the path (which is LCU_C in this case).

From the proof above, a robot will traverse a straight optimal path safely at linear velocity less than V if and only if two necessary conditions in Proposition 3.1 are met.

Proposition 3.1 A nonholonomic mobile robot, with sampling period T, traverses a straight optimal path with a linear velocity V in a workspace without the likelihood of collision if and only if,
(i) the robot configuration is always on the optimal path, and
(ii) the maximum control uncertainty of the robot is equal to or less than the minimum LCU of the optimal path.
Proof: See the above paragraph.

In the case of a free-form optimal path such as the one in Figure 3.8, the LCU at each configuration along the optimal path is only an approximation of the control uncertainty at that configuration because the definition of the LCU according to Definition 3.6 is based on a straight nominal path. Therefore, at a given sampling period T and maximum wheel velocity V, the accuracy of the LCU approximation degrades as the curvature of the path increases. This is clearly shown in Figure 3.4, where both tangent vectors (indicate robot configurations and their nominal straight paths) at the beginning and the end of the s-shaped path point further away from the path than those in the middle of the path, and, therefore, both vectors do not reflect the tangents of the path after moving straight away from that point. To cope with this problem, one can decrease the sampling period to shorten the vectors and to approximate the path with more short vectors. However, the sampling period is limited by the robot hardware; for this reason, the method is impractical. Another alternative is to sample more points (configurations) on the free-form path, and use the same T for determining the LCU at each sampled points. The finer the sample is, the better the approximation gets. However, since the minimum LCU of the path obtained by this method, as shown in Figure 3.8(b), is just an approximation of the actual minimum LCU of the path, the same proposition in the case of a free-form optimal path cannot be stated. This minimum LCU, although not the exact one, can still be useful as a guideline to determine how much control error is allowed in a robot system.

The problem of the approximated control uncertainty has been solved by the introduction of the concept of Circular Control Uncertainty (CCU). This CCU gives the exact control uncertainty for a path consisting of straight lines and circular arcs, and will be discussed in the next chapter.

3.6 Conclusion

The notion of “Linear Control Uncertainty (LCU)” is proposed as a measurement, for each configuration in the configuration space, indicating how much uncertainty tolerance that the configuration has comparing to its neighbor. A field of LCUs in a workspace, which we called “Linear Control Uncertainty Field (LCUF)”, can be searched by the proposed search method for an optimal path lying within the workspace. This search method, called “LCUF-based motion planning”, is based on the approximation-and-optimization approach, which is often used in solving for optimal trajectories in optimal control problems. The search method yields a global optimal path with respect to the cost function (3.11). For any optimal straight path, collision-free condition can be guaranteed for a robot with control uncertainty equal to or less than the minimum LCU of the path if and only if the robot accurately follows the path with linear velocity less than or equal to the maximum wheel velocity V. For a free-form optimal path, the approximated minimum LCU of the path can be used as a rough guideline to determine how much control uncertainty is allowed in a robot system.

Chapter 4

Circular Control Uncertainty Field-Based Motion Planning

This chapter presents the concept of "Circular Control Uncertainty (CCU)", where the limitation of dealing with only a straight nominal path of the LCU is removed. The CCU can determine the exact CU for any circular arc with arbitrary curvature. Due to a large search space of the Circular Control Uncertainty Field (CCUF), a new kind of motion planning called "CCUF-based motion planning" is proposed. An optimal path—consisting of arcs and straight lines (zero curvature)—obtained from this motion planning is guaranteed to be global optimal. Conditions for collision-free motion of a mobile robot are also given at the end.

4.1 Definition of Circular Control Uncertainty

In Chapter 3, the linear control uncertainty or LCU is defined as a nonnegative real number that indicates the allowable maximum amount of perturbation to the controls, v_r and v_l, of a mobile robot, such that the likelihood of collision is eliminated. However, both wheel velocities in the definition of LCU are fixed at a constant nominal value V, which means that the value of LCU is exact only for a straight path and is only a close approximation for a free-form path. To determine the exact control uncertainty for any kind of paths, the current definition of the LCU must be modified. By introducing two additional velocity variables, V_r and V_l, for each wheel into Definition 3.6, we obtain a definition for what we called the "Circular Control Uncertainty (CCU)".

Definition 4.1 Let T be a constant time period, and $T > 0$. Given a robot $\mathcal{A}$ moving in a workspace $\mathcal{W} \in R^2$ containing obstacles $\mathcal{B}_i$, the **Circular Control Uncertainty** (CCU) of a configuration $q \in \mathcal{C}_{free}$ is a nonnegative real number that perturbs constant nominal controls V_r and V_l (where both V_r and $V_l \in R$) for the actual controls v_r and v_l in equation (3.2), as defined by the following equations

$$\begin{aligned} v_r &= (1+CCU)\cdot V_r, \quad v_l = (1+CCU)\cdot V_l \\ v_r &= (1-CCU)\cdot V_r, \quad v_l = (1-CCU)\cdot V_l \\ v_r &= (1+CCU)\cdot V_r, \quad v_l = (1-CCU)\cdot V_l \\ v_r &= (1-CCU)\cdot V_r, \quad v_l = (1+CCU)\cdot V_l \end{aligned}, \tag{4.1}$$

such that the trajectory of system (3.1) starts from $x(0) = q_0 = q \in \mathcal{C}_{free}$ and ends at $x(T) = q_T \in \mathcal{C}_{free}$ with $x(T^+) = q_{T^+} \in \mathcal{C}_{contact}$, where $T^+ \in (T, \infty)$.

According to this definition, the CCU represents the exact control uncertainty of a configuration on any nominal path realized by V_r and V_l, and the LCU in the previous chapter is a special case of the CCU when $V_r = V_l = V$. Because the values of V_r and V_l are held constant during each constant time period T, the nominal path for each time period is in the form of circular arc with different curvature corresponding to the values of V_r and V_l. As a result, if T is finite and large, a free-form path can be closely approximated by a series of discrete configurations and their arcs, as depicted in Figure 4.1. Points (positions) and curvy arrows (orientations and nominal paths) indicate the discrete robot configurations (x_c, y_c, θ) and their nominal paths along a continuous path.

If one reduces the time period T by letting T approaches zero from the right, $T \to 0^+$, the values of CCUs obtained along the path are the exact control uncertainty along the path. However, the difficulty arises when T is limited by the robot hardware. A more feasible approach is to approximate a path with a series of arcs and straight lines instead, and the exact control uncertainty can still be determined by the definition of the CCU at each period T.

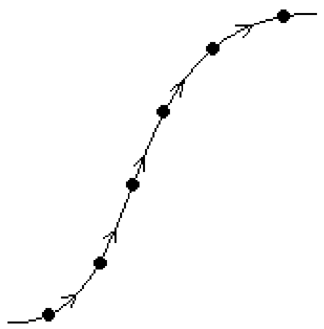
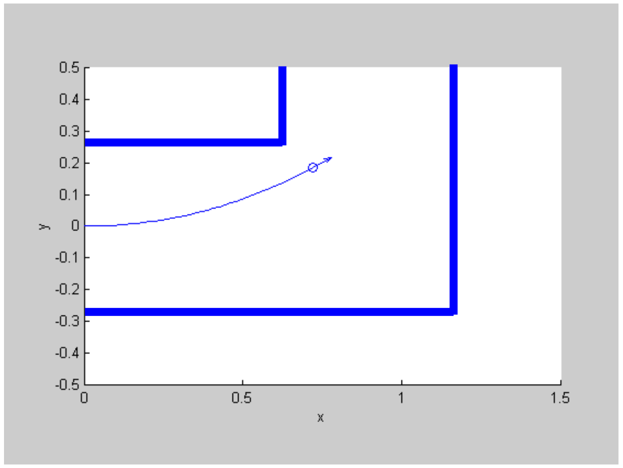
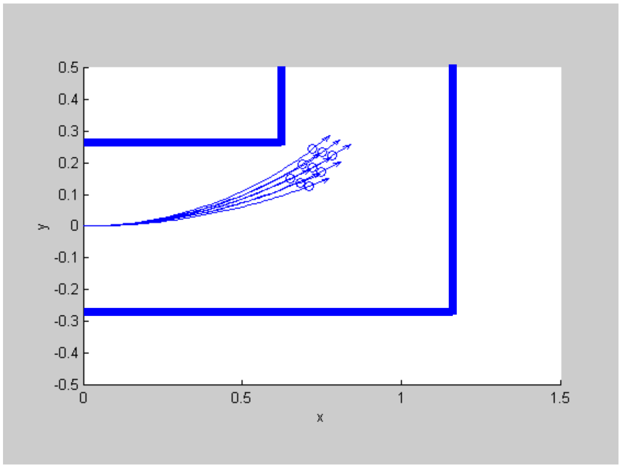


Figure 4.1 A path is approximated by a series of discrete configurations and their nominal paths.

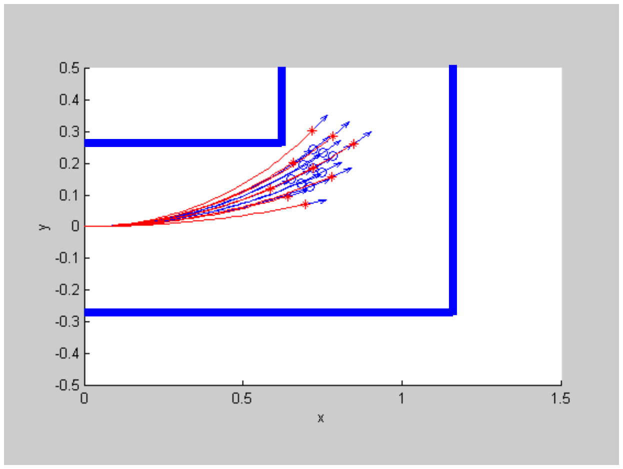
Similar to the LCU, the CCU can be thought as the maximum allowable perturbation applied to the nominal trajectory/path before the perturbed path causes the robot to collide with obstacles in its workspace. The process of computing the CCU for each configuration corresponding to velocities V_r and V_l is the same as in that of the LCU except that both nominal wheel velocities V_r and V_l can now take on any finite value. The value of CCU at a particular configuration with given V_r and V_l can be determined by means of simulation. Given a robot, a robot workspace, obstacles, and T, the simulation iterates for each particular configuration by increasing CCU (starting from zero), calculating perturbed controls from equation (4.1), integrating system equation (3.1) forward in time for T seconds, and checking collision between a robot and obstacles. If there is no collision, the CCU is increased and another iteration is carried out. The simulation terminates when collision occurs, and the CCU just before the collision is declared as the CCU of that particular configuration. A graphical illustration for the determination of the CCU of a point robot, represented as a point and an arrow to show orientation, at the configuration (0,0,0) with V_r = 1.0 unit/second and V_l = 0.75 unit/second is shown in Figure 4.2. Notice that the nominal path in Figure 4.2(a) spreads wider as the value of CCU increases. The simulation terminates at CCU = 0.3 as shown in Figure 4.2(d), and 0.2 is the value of CCU at configuration (0,0,0) with specific values of V_r and V_l. In case of sized robots, a more involved collision detection algorithm is implemented to detect the interference during the entire simulation.



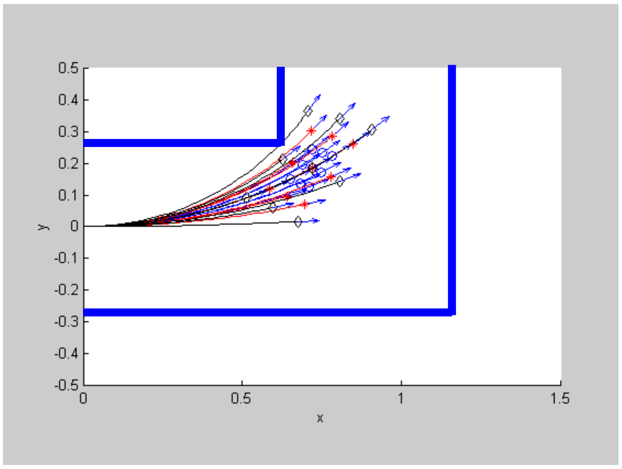
(a)



(b)



(c)



(d)

Figure 4.2 A sequence of the simulation for determining the CCU of configuration (0,0,0) of a point mobile robot with $V_r = 1.0$ unit/second and $V_l = 0.75$, (a) a nominal path, (b) perturbed paths with CCU = 0.1, (c) perturbed paths with CCU = 0.2, and (d) perturbed paths with CCU = 0.3.

The concept of the CCU can be directly applied to the motion planning problem of nonholonomic mobile robots the same way as that of the LCU. The idea is to find a path that connects between given initial and goal configurations, with all of the configurations on the path possess high values of CCUs, by searching through a field of CCU which we called “ Circular Control Uncertainty Field (CCUF)”. Its formal definition is given below.

Definition 4.2 Given a robot $\mathcal{A}$ moving in a workspace $\mathcal{W} \in R^2$ containing obstacles $\mathcal{B}_i$, a **Circular Control Uncertainty Field** (CCUF) is a field of CCU for all $q \in \mathcal{C}_{free}$.

At this point, we are ready to discuss a proposed approach for searching the CCUF for an optimal path. First, the direct search problem in the CCUF is transformed into a graph search problem, and then a standard search scheme can be utilized. The detail discussion of the proposed approach is presented in the next section.

4.2 Search Scheme

One problem we have discovered in the previous chapter is that the search program takes much time (up to 96 hours) to search for an optimal path in a LCUF. It is obvious that the same search program will take much more time in searching through a CCUF, which is also dependent with nominal control variables V_r and V_l. To cope with this problem, a new optimal path search strategy must be derived. One such strategy is limiting the search space for the search program.

By discretizing both state space (configuration space $\mathcal{C}$) and control space u of the system to have finite numbers of values, the search space is bounded and finite. The state of the system is divided into many cells and the control can take on any values within a prescribed set, as oppose to the LCUF-based motion planning method in which the sets of state and control are basically treated as being continuous. Therefore, the search time for the new search strategy is bounded and decreased. Although a result obtained from the search will be only an approximation of an optimal path, the accuracy of the result can approach the optimal path by refining the size of discretized state and control within the narrower search space obtained from the result from the previous search.

Since the number of discretized controls is fixed, a directed graph can be constructed by expanding successive nodes (or successors) from the initial node (at initial configuration, denoted as node **I** in Figure 4.3) toward the goal node (at the goal configuration, denoted as node **G**) with a finite number of expanded nodes, where each node connects to another by a straight line or a circular arc respecting the nonholonomic constraint to ensure that every node connected by feasible paths (denoted by arrows in Figure 4.3). This line or arc can be determined by integrating the system model (3.1) for given discretized controls. During node expansion, if the expanded nodes lie outside the free workspace, those nodes will be terminated and removed from the graph. Once the directed graph has been constructed, a heuristic graph-search method such as the A* algorithm—whose detail is provided in Chapter 2—can be deployed to search the graph. An important characteristic of the A* algorithm is that the search result is always guaranteed to be globally optimal, if the result exists. Additionally, it uses less search time than other graph-search methods such as breadth-first search.

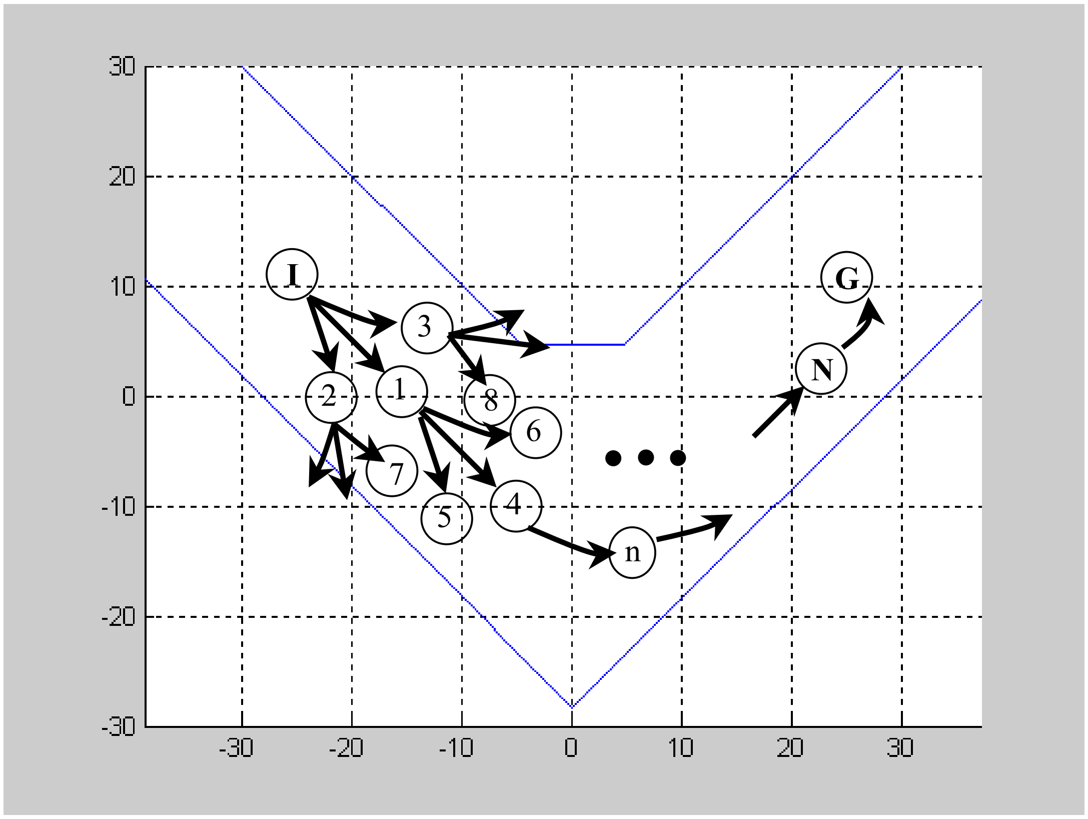


Figure 4.3 A directed graph in a workspace.

Figure 4.3 shows a directed graph constructed inside a workspace. In this illustration, the discretized controls assume three possible movements: forward straight, forward right, and forward left. The first expansion of the graph starts from node **I**, and its successors are node 1, 2, and 3. Then, the second expansion of the graph starts from the recently expanded nodes: node 1, 2, and 3. The successors of node 1 are node 4, 5, and 6. The successor of node 2 is node 7 only, since the other two nodes land outside the workspace. Similarly, the successor of node 3 is node 8 only. The expansion process repeats itself until the goal node **G** is attained, or the specified depth of the expansion is reached. The result is the directed graph that can be used for searching.

The objective in the search, in our case, is to find a path that connects initial and goal nodes and possesses the maximum value of the minimum CCU. The minimum CCU for each path is defined by the minimum value of CCUs of all nodes (configurations) in that path. The cost function for each node (assigned by the A* algorithm) in a directed graph is expressed in the following form.

$$f(n) = g(n) + h(n), \qquad n \in [1, N] \tag{4.2}$$

where

$$g(n) = -\min\{CCU(n-1), CCU(n)\}$$
$$h(n) = (x(n) - x_g)^2$$

n = node n, and node 0 is the initial node
$CCU(n)$ = circular control uncertainty at node n
$x(n)$ = configuration or state at node n

From (4.2), *f(n)* is the cost of a path from the initial node constrained to go through node *n*. This cost is assigned to each successor in every node expansion step inside the A* algorithm. The A* algorithm uses these cost values to determine which successor should be expanded (or explored) next and which search direction it should take. *g(n)* is the cost of the path so far from the initial node to node *n*, while *h(n)* is the heuristic function providing the lower bound of the cost of the path from node *n* to the goal node. In our application, *h(n)* is selected to be the Euclidean distance between node *n* and the goal node, and, therefore, serves as the lower bound of the traveling distance from node *n* to the goal node. This satisfies the condition (2.9) in which the optimality of the search result is guaranteed. If *h(n)* is set to be zero (the lowest value possible and a trivial admissible heuristic function), the A* search algorithm simply becomes a breadth-first search type algorithm (a graph-search with no heuristic information). In this case, the A* algorithm will explore every node in the graph (exhaustive search). Although the search result is optimal, the algorithm takes much more time to search.

The search strategy based on the A* algorithm described above has been implemented in an engineering programming language, MATLAB®. The search program, listed in Appendix B, is applied to search for an optimal path with respect to the cost function (4.2) for the same case study as discussed in Chapter 3 with slightly different initial configuration. The result is given in the next section.

4.3 Results and Discussions

We have applied the A* algorithm-based search program to solve the same motion planning problem as in the LCUF-based motion planning (section 3.5). Our primary goals are twofold. The first goal is to compare the result obtained from the CCUF-based motion planning to the result previously obtained from the LCUF-based motion planning. The second goal is to use the knowledge earned from the LCUF-based motion planning to help minimize the search time for the CCUF-based motion planning. Since we have prior knowledge from the previous study that an optimal path for a straight passage lies in the middle of the passage, to help reduce the search time, we move an initial configuration to $[-19,5.5,-\pi/4]^T$, which is in the middle of the passage and closer to the goal configuration at $[25,11,\pi/4]^T$. For the cost function, we use *h(n)* and *g(n)* as they are in (4.2). The depth of the expansion is limited at 50 steps. The sampling period T = 0.5 seconds.

The configuration space of the mobile robot is discretized into 0.25 unit increments in both x and y directions, and $\pi/64$ radian increments in the θ direction. The control space, $u = [v, \omega]^T$, is also discretized into 9 possible control sets as shown below.

$$u = \begin{bmatrix} 1.0 & 0 \\ 0.2 & 0.20 \\ 0.4 & 0.15 \\ 0.6 & 0.10 \\ 0.8 & 0.05 \\ 0.2 & -0.20 \\ 0.4 & -0.15 \\ 0.6 & -0.10 \\ 0.8 & -0.05 \end{bmatrix}^T.$$

The search program, running on a personal computer with a 1.7 GHz processor, took about 38 hours to arrive at the result. Figure 4.4 shows the optimal path obtained from the search program.

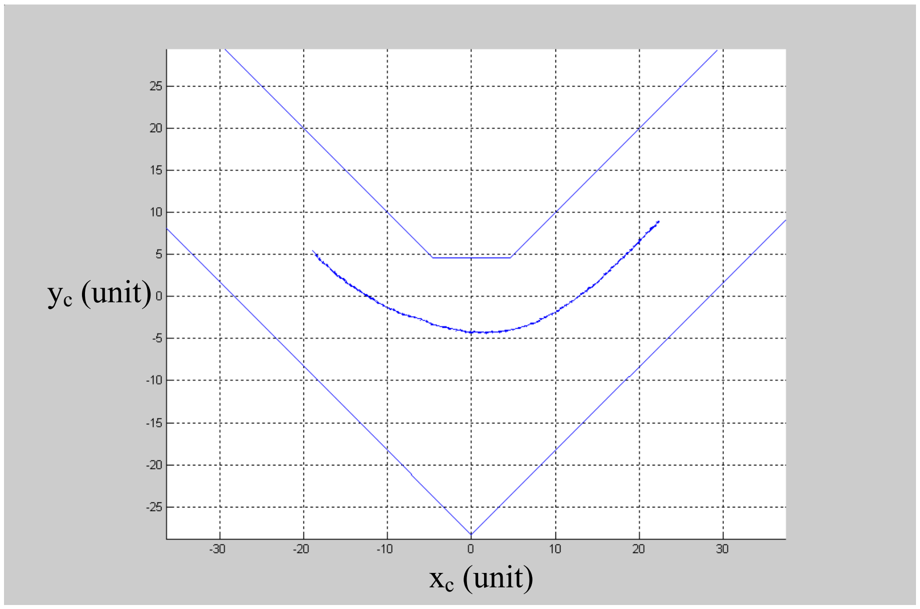


Figure 4.4 The optimal path from the A* algorithm-based search program.

From Figure 4.4, the optimal path starts from the left at the initial configuration and ends near the goal configuration, at configuration $[22.31, 8.79, \pi/3.94]^T$. When one compares the optimal path in Figure 4.4 and the one in Figure 3.8(a), it is clear that both of them are almost identical. This reassures that the optimal path obtained previously by the LCUF-based motion planning is a true global optimum, since the path obtained from the A* algorithm is always guaranteed optimal.

Figures 4.5 and 4.6 present the optimal control and the CCUs for every node along the optimal path. In Figure 4.5, the optimal linear velocity (the top curve) is fluctuated around 0.6 unit/second within the range from 0.4 to 0.8 unit/second, and the optimal angular velocity (the bottom curve) is fluctuated around 0 rad/second within the range from -0.15 to 0.15 rad/second. The reason for the apparently oscillation in the optimal control is due to coarse discretization in both sampling period T and discretized control to reduce the optimization time. This optimal control results in a small curvy path (comprising a series of arcs), which can be seen if one magnifies the optimal path shown in Figure 4.4. However, the optimal path turn outs to possess higher allowable control uncertainty than the one found in the previous LCUF-based motion planning, the minimum CCU of the optimal path is 3.2 or 320%, which is more than twice the value of the minimum control uncertainty obtained from the LCUF-based approach.

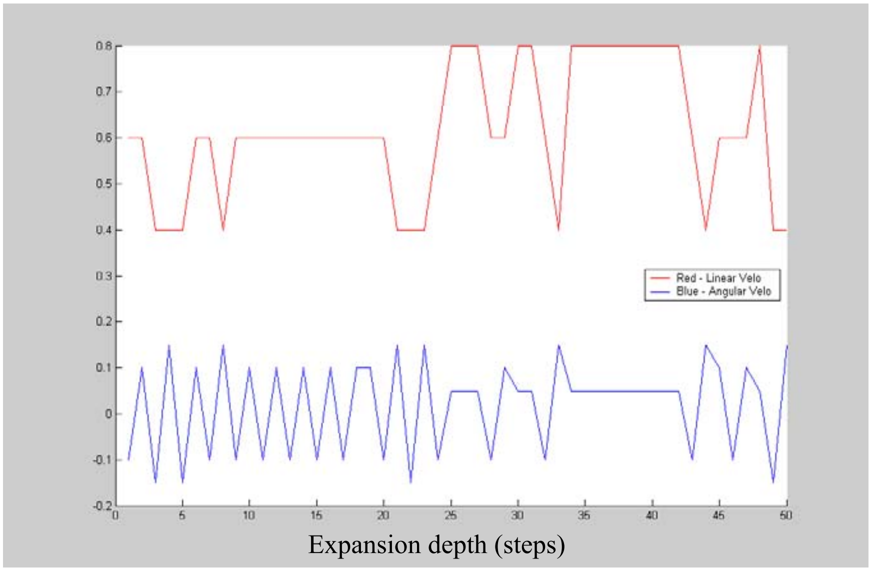


Figure 4.5 The optimal control from the A* algorithm-based search program.

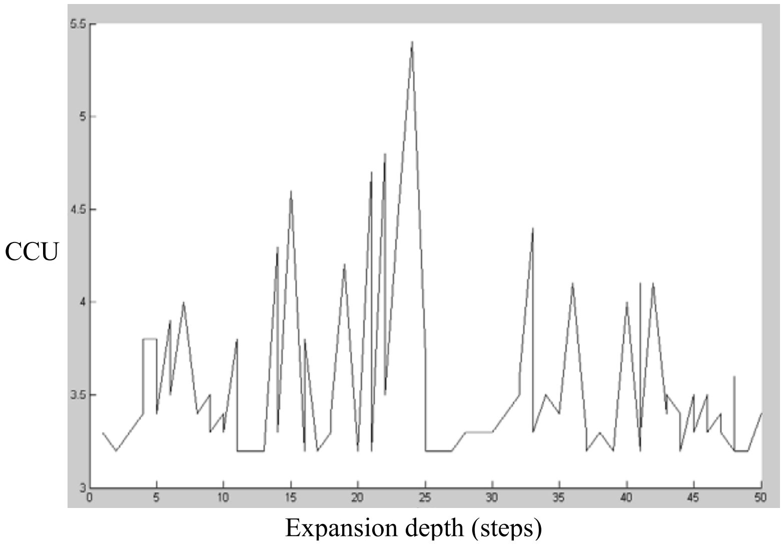


Figure 4.6 Circular control uncertainties (CCUs) along the optimal path.

One reason why such a curvy path possesses higher allowable control uncertainty is, in part, due to smaller average forward linear velocity along the path. As one can see from the LCUF-based approach in Figure 3.7(c) that the average linear velocity along the optimal path is close to 1 unit/s while it is close to 0.6 unit/s in this case. As we have learned from Chapter 3 that the value of LCU will increase if the value of the linear velocity decreases, this logic is also applied for the case of CCU. To discard the effect of the linear velocity, we have recalculated the value of minimum CCU of the optimal path using the linear velocity equals to 1 unit/s, and find that the minimum CCU drops to 2.0 or 200%. The other reason is that the control uncertainty computed by the LCUF-based approach is just an approximation of the exact control uncertainty computed by the CCUF-based approach; therefore, the value of the CCU is more reliable for a curved path. Regardless of the discrepancy in the minimum LCU and CCU of both optimal paths obtained from two different approaches, the contours of both paths are closely similar to each other.

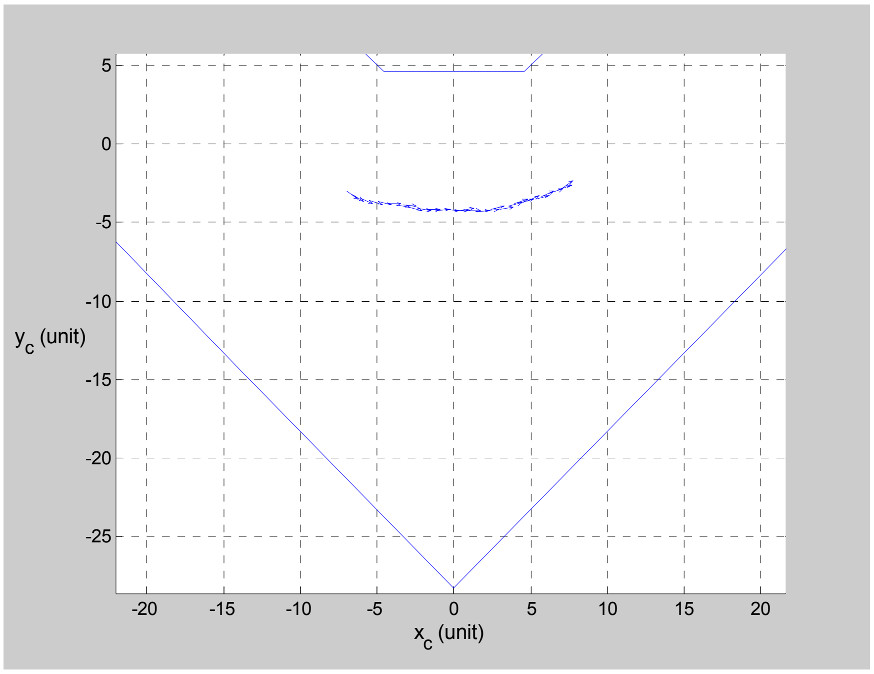
We have further investigated the effect of the discretization of the control space to the quality of the optimal path obtained from the A* algorithm-based search method. We have discovered that the quality of the optimal path (in term of smoothness) can be improved by using finer discretization for the control space, although the search time will be substantially increased (which is not our concern in this investigation). To show that the smoothness of the optimal path can be improved through the use of a finer discretized control space in the search, a simpler case study is used to limit the search time to be within a reasonable range (under 10 hours). Using the same motion planning problem as shown in Figure 4.4, the problem is made simpler by moving both initial and goal configuration closer to each other. Specifically, the new set of the initial and goal configurations is given to be $[-7, -3, -\pi/5]^T$ and $[7, -3, \pi/5]^T$, respectively.

The depth of the expansion is limited to no more than 35 steps. The configuration space of the mobile robot is now discretized into 0.1 unit increments in both x and y directions, and 0.1 radian increments in the θ direction. This discretized configuration space is fixed in all optimal path searches. Two search experiments are conducted and compared. The first search uses the same discretized control space discussed previously. The second search uses a finer discretized control space, $u = [v, \omega]^T$, which is given as follows.

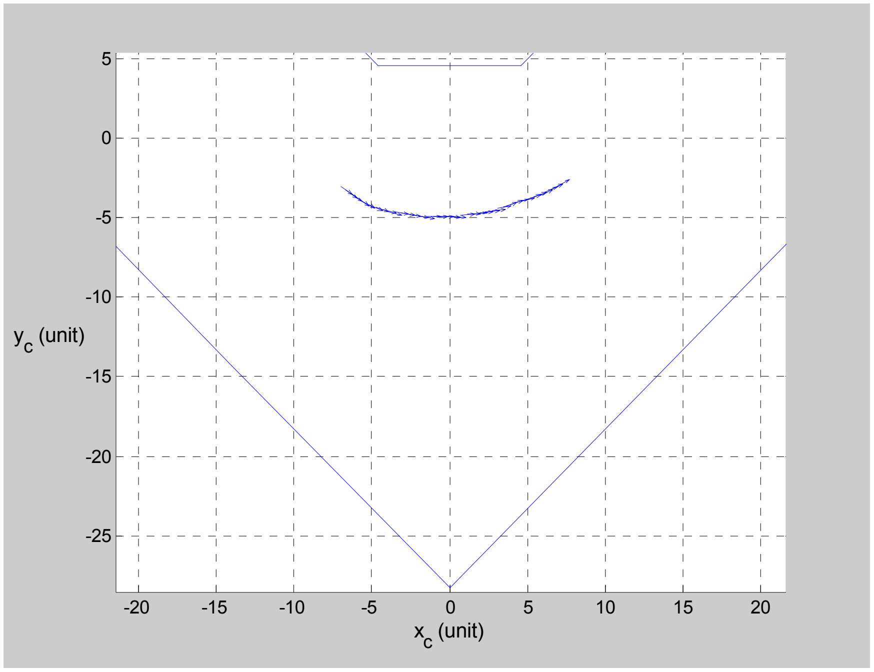
$$u = \begin{bmatrix} 1.0 & 0 \\ 0.4 & 0 \\ 0.4 & 0.025 \\ 0.4 & 0.050 \\ 0.4 & 0.075 \\ 0.4 & 0.100 \\ 0.4 & -0.025 \\ 0.4 & -0.050 \\ 0.4 & -0.075 \\ 0.4 & -0.100 \\ 0.5 & 0 \\ 0.5 & 0.025 \\ 0.5 & 0.050 \\ 0.5 & 0.075 \\ 0.5 & 0.100 \\ 0.5 & -0.025 \\ 0.5 & -0.050 \\ 0.5 & -0.075 \\ 0.5 & -0.100 \\ 0.6 & 0 \\ 0.6 & 0.025 \\ 0.6 & 0.050 \\ 0.6 & 0.075 \\ 0.6 & 0.100 \\ 0.6 & -0.025 \\ 0.6 & -0.050 \\ 0.6 & -0.075 \\ 0.6 & -0.100 \end{bmatrix}^T.$$

Obviously, the discretized control space of the second search allows more choices in angular velocities for a given linear velocity.

Two optimal paths obtained from two different sets of discretized control space are shown in Figure 4.7(a) and (b). The difference in smoothness of both paths can be clearly seen from both figures. The optimal path obtained from the first search appears curvier than the one obtained from the second search. This can be easily explained by considering their corresponding optimal controls. The optimal control history for each of the optimal path is presented in Figure 4.8(a) and (b). From these figures, it is obvious that the optimal control of the second search is smoother than that of the first search (less oscillatory). This is a direct consequence from using the finer discretized control space in the second search.

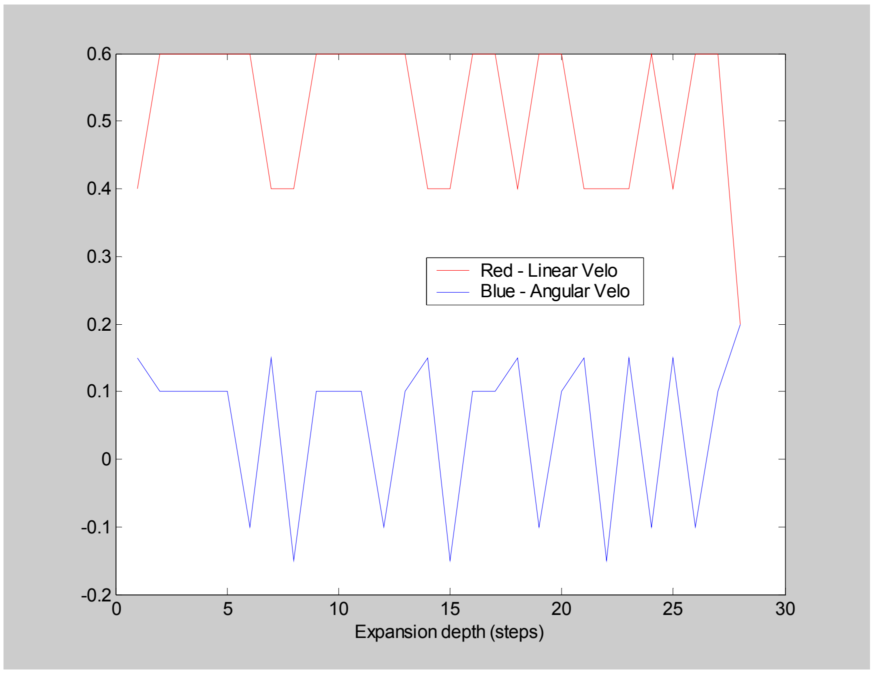


(a)

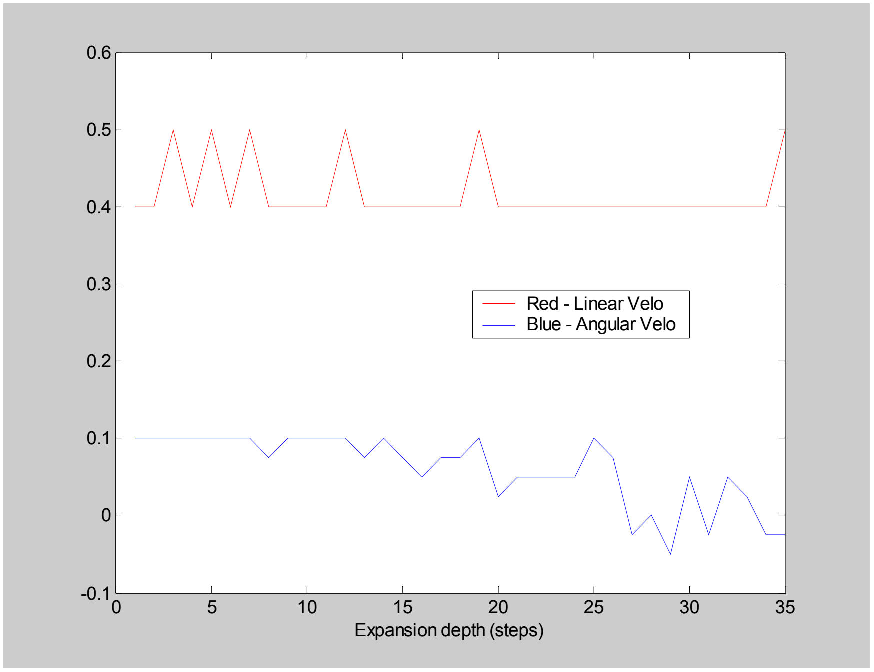


(b)

Figure 4.7 The optimal paths obtained from (a) the first search, and (b) the second search.



(a)



(b)

Figure 4.8 The optimal control obtained from (a) the first search, and (b) the second search.

From Figure 4.8, it is also worth noting that the first search takes 28 steps to move from the initial configuration to the goal configuration while the second search takes 35 steps. This is because the average linear velocity of the first search is more than that of the second search—approximately 0.55 unit/s for the first search and 0.45 for the second search. Therefore, the second search takes more steps in reaching the goal configuration.

Optimality of the path. Unlike the LCUF-based motion planning, an optimal path obtained from the CCUF-based motion planning with respect to the cost function (4.2) is guaranteed to be a global optimum. This is due to the fact that the A* search algorithm used in the search scheme always return the global optimal result, see Chapter 2 for detail. Therefore, the optimality of the resultant path from the CCUF-based motion planning can be stated without proof.

Estimation of safe velocity and control uncertainty. For a given robot and a workspace, the CCU is a function of configuration, sampling period, and nominal control (or path). The CCU, therefore, is in fact the exact control uncertainty of a given arc since the nominal control is held constant over a sampling period. If the sampling period is set to a constant, the CCU is only a function of both configuration and nominal control, which is basically a curvature of the arc. From Figures 4.5 and 4.6, the values of the CCUs stay above or equal to the minimum CCU of the optimal path as long as the optimal linear velocity is at the minimum value (0.4 unit/second) regardless of the value of the optimal angular velocity. This can be clearly seen in both Figures for steps 3 to 5, at step 8, for steps 21 to 23, at step 33, at step 44, and for steps 49 to 50. As a result, in this case, a mobile robot can traverse the optimal path with no collision if and only if the robot always stays on the optimal path, moves with linear velocity below 0.4 unit/second, and possesses uncertainty in its velocity control less than 3.2 or 320%.

In general case, once the optimal path, the optimal control, and the minimum CCU of the optimal path have been identified. The following proposition can be stated.

Proposition 4.1 A nonholonomic mobile robot, with sampling period *T*, traverses an optimal path in a workspace without the likelihood of collision if and only if,
(i) the robot configuration is always on the optimal path, and
(ii) the maximum control uncertainty of the robot is equal to or less than the minimum CCU of the optimal path.

Proof: The first condition is required to ensure that at any sampling period *T*, the robot always start moving from a point on the optimal path with the optimal control previously computed, which is essentially a nominal path for the robot to follow during a period of time equal to the sampling period. If the first condition is satisfied, the robot can have uncertainty in its control as much as the minimum CCU of the optimal path to ensure that after one sampling period has elapsed the robot configuration is still be in the free configuration space, $\mathcal{C}_{free}$. This statement is always true by the definition of the MCU. Therefore, the proof is concluded.

A more conservative estimation of safe velocity and control uncertainty can also be deduced from the knowledge of both the CCU and the LCU introduced in Chapter 3. Given any free-form optimal path, the CCU can be used to estimate the allowable control uncertainty along the path more closely than the LCU. Additionally, the optimal path possesses the highest allowable control uncertainty in any workspace, which means that the allowable control uncertainty drops immediately after a mobile robot deviates from the optimal path. The CCU has higher magnitude than the LCU since the nominal path defined for the CCU closely follows the contour of the optimal path. Therefore, if the mobile robot with its initial configuration on the optimal path begins to move along its straight nominal path as defined for the LCU, the robot instantly fall off the optimal path. As a result, the value of LCU is always lower than that of CCU. Hence, one can safely use the LCU as a lower bound of the CCU along the optimal path. We can now state that:

Proposition 4.2 A nonholonomic mobile robot, with sampling period *T*, traverses an optimal path with linear velocity *V* in a workspace without the likelihood of collision if and only if,

(i) the robot configuration is always on the optimal path, and

(ii) the maximum control uncertainty of the robot is equal to or less than the minimum LCU of the optimal path.

Proof: see above paragraph.

4.4 Conclusion

In this chapter, the notion of "Circular Control Uncertainty (CCU)" is proposed as an exact measure of the maximum allowable control uncertainty of a nonholonomic mobile robot traveling along an arbitrary nominal path (straight line or arc) from any configuration in a free configuration space with nominal wheel velocities, V_r and V_l, in one fixed period of time T. A field of CCUs, called "Circular Control Uncertainty Field (MCUF)", is used to search for an optimal path comprising arcs and straight lines. Instead of searching directly in the CCUF, a directed graph is proposed as an approximation of the representation of the CCUF. This helps reduce search effort considerably by way of discretizing both configuration and control of the robot.

A proposed motion planning based on the well-known A* search algorithm is employed to search the direct graph of the CCUF for an optimal path connecting given initial and goal configurations. We called this motion planning the "CCUF-based motion planning". The optimal path obtained from this motion planning scheme is global optimal. However, the optimal control is highly discontinuous, and so is the optimal path. The discontinuity in the optimal control, however, can be alleviated by using a finer discretized control in the search algorithm. Despite the fact that the optimal path is not smooth, collision-free condition is guaranteed for a robot that follows the optimal path exactly with the maximum allowable in its control system less than or equal to the minimum CCU of the path. Additionally, at each point along the optimal path, the LCU is the lower bound of the CCU, and a mobile robot can safely traverse the optimal path with linear velocity V if the maximum control uncertainty of the robot is equal to or less than the minimum LCU of the optimal path.

Chapter 5

Motion Planning Performance Comparison and Improvement

The chapter begins with the performance comparison of both motion planning techniques proposed in previous chapters. The LCUF-based motion planning is chosen to be better than the CCUF-based motion planning largely due to the quality of the returned path. However, the LCUF-based motion planning is plagued by its high computation time required to search for the optimal path. To cope with this problem, the idea of using the estimated LCU instead of the exact LCU is proposed. A simple but effective estimation scheme for approximating the LCU at each configuration is derived and proved to be very efficient in reducing the search time and improving search result. The global optimality property of the estimated LCUF is also given at the end of the chapter. This property guarantees the optimality of the resultant path returned from all of the proposed motion plannings.

5.1 Performance Comparison for Proposed Motion Planning Schemes

The performance of both path planning schemes proposed in the previous chapters—the LCUF-based motion planning and the CCUF-based motion planning—are compared to determine the advantages and disadvantages of each path planning schemes. The main criteria of the comparison are focused on five properties of both planners. The first criterion is the complexity of the planner, which can be directly measured by means of a total computation time used by each planner. Ultimately, we are interested whether the planner can be run in real-time. The second criterion deals with the optimality of the path returned from each planner. In this criterion, the path can be either globally or locally optimal. The third criterion relates to the geometrical attribute of the optimal path, where continuity and smoothness of the path are major concerns. The fourth criterion regards with the quality of the optimal control, which in turn generates the optimal path. The continuity and smoothness of the optimal control are considered in this case. The last criterion concerns with how accurate each planner can compute the value of control uncertainty (CU) for each configuration on the optimal path.

We used the data of the results obtained from applying two different path plannings to solve the same path planning problem in Chapters 3 and 4. Based on the criteria above, the collected data is compared, and the comparison result is given in Table 5.1. The advantages for each path planning are listed in italic typeface.

Table 5.1 The performance comparison between the LCUF-based motion planning and the CCUF-based motion planning. Italic typefaces denote advantages.

Criteria	**LCUF-Based Motion Planning**	**CCUF-Based Motion Planning**
Computation time	Not real-time (96 hours)	Not real-time (38 hours)
Optimality of the path	Global optimal	Global optimal
Quality of the optimal path	*Smoother*	Less smooth
Quality of the optimal control	*Piecewise continuous*	Coarse and oscillates
Accuracy of the CU	Exact (straight path only)	*Exact (straight path and arc)*

From Table 5.1, it is not obvious to conclude immediately which path planning is better than the other. At least, both motion planners yield a global optimal path. The LCUF-based planner produces more desirable results in terms of the qualities of both optimal path and control. But, the CCUF-based planner computes exact values of control uncertainty for both arcs and straight paths while the LCUF-based give only approximate values of the CU for arcs. Hence, it seems that both motion planners are equally good because they both have advantages and disadvantages over each other.

Nonetheless, the goal of developing a motion planning scheme is to produce a path. Therefore, the quality of the path generally is the primary concern. For this reason, we conclude that the LCUF-based motion planning scheme is more favorable than the CCUF-based motion planning scheme, and decide to pursue further in studying and improving the performance of the LCUF-based motion planning method.

5.2 Estimation of the LCU in a Workspace

To improve the performance of the LCUF-based motion planning method, we first identify the weaknesses of the motion planner. It is obvious from Table 5.1 that one of its major weaknesses is its computation time. After scrutinizing the structure of the planner and its computer program, we discovered that the planner spends most of its time determining the value of LCU for each configuration along a candidate path. As we mentioned in Chapter 3 that the value of LCU could be determined by means of simulation, the time takes to determine the LCU is, therefore, varied over the free configuration space. This means that the higher the value of the LCU gets, the longer time it takes to simulate. Moreover, the time increases when the problem deals with a more complex workspace (more line segments) since the collision detection algorithm employed in the simulation must require more computation time. To cope with this problem, a new technique for determining the value of LCU must be developed.

Reconsidering the definition of the LCU, we learn that the value of the LCU at each configuration depends only on the geometries of both mobile robot and workspace, provided that time period *T* and nominal control *V* are fixed constant. The value of the LCU is determined as a maximum perturbation that can be applied to a nominal trajectory of the robot without causing the collision; hence, the collision between the robot and its workspace is a key to determine the value of the LCU. Both robot and workspace in this research are represented by a convex polygon and line segments. When a collision occurs, a part of a robot polygon (either its vertices or line segments) intersects with a part of a workspace (either its vertices or line segments). Notice that, for any collision, the intersection occurs between a robot polygon and only one of the workspace line segments. Based on this observation, if we look at a workspace as a series of line segments joined together to form a continuous boundary, and each line segment contributes to the value of the LCU. One can determine the value of LCU by combining the effect of each line segment constituting the value of the LCU. This concept is the origin of the new technique developed for estimating the value of LCU. To this end, we need to determine two elements. One is the effect of each line segment of the workspace to the value of LCU. The other is a method to combine the effects from each line segment. These two elements are essential for the development of the LCU approximation technique, and will be discussed next.

LCUF template of a line. The effect of each line segment is represented by the determination of a field of LCUs around a given line segment. This LCUF around a line, which is later called a "LCUF template", can be determined by simulating a mobile robot in a workspace with obstacle comprising only one line. Given a convex polygon representing a mobile robot $\mathcal{A}$ and a line segment *LR* with length *l* (as shown in Figure 5.1), the robot configuration at the origin of the robot coordinate frame *{xy}* with respect to the line coordinate frame *{XY$_{line}$}*, whose origin is located at the middle of the line, is given by (x, y, θ). The LCUF template of a line is defined as follows:

Definition 5.1 Given a convex robot $\mathcal{A}$ moving in a workspace $\mathcal{W} \in R^2$ containing only one obstacle $\mathcal{B}$, which is a single line segment, a **Linear Control Uncertainty Field Template of a Line** (LCUF template) is a field of LCUs for all $(x, y, \theta) \in [x_{\min}, x_{\max}] \times (0, y_{\max}] \times (-\pi, \pi]$.

From the definition, the LCUF template is a field of LCUs over the upper half plane of the *{XY$_{line}$}* frame bounded by $x \in [x_{\min}, x_{\max}]$, $y \in (0, y_{\max}]$, and $\theta \in (-\pi, \pi]$, as shown in Figure 5.2. A LCUF template also consists of three sub templates corresponding to three regions of a line: Left End, Line, and Right End regions. Each region can overlap one another as presented in Figure 5.2. The usage of sub templates is necessary since in the case where a line is very long (imagine an infinite line) the LCUs of a robot is affected by the effect of a line region only. On the other hand, if a line was very short, the effects of both end regions overwhelm the effect of the line region. To make a LCUF template invariant to the length of a line, the concept of separately determine a sub template corresponding to each region is essential. However, the method of combining these three sub templates to obtain a LCUF template must also be derived.

The Line region represents the effect of an infinite length line to a convex robot. As a result, the value of LCU does not depend on the x-coordinate. This means that the value of LCU at any (y, θ) is invariant over $x \in (-\infty, \infty)$ for an infinite length line, and can be assumed to be invariant over $x \in [-l/2, l/2]$ for a finite length line. However, in both Left End and Right End regions, the value of LCU is still the function of (x, y, θ). The definition of each region is provided below.

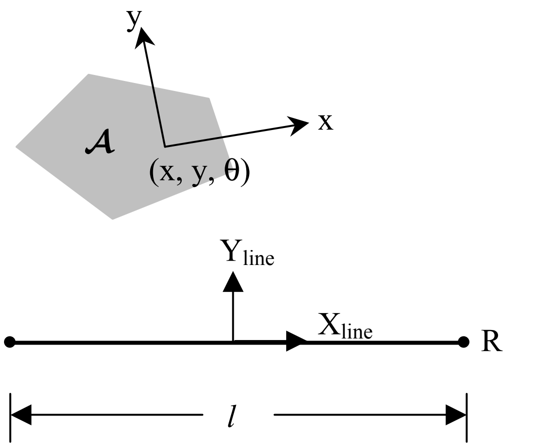


Figure 5.1 A robot coordinate frame *{xy}* and a line coordinate frame *{XY$_{line}$}*.

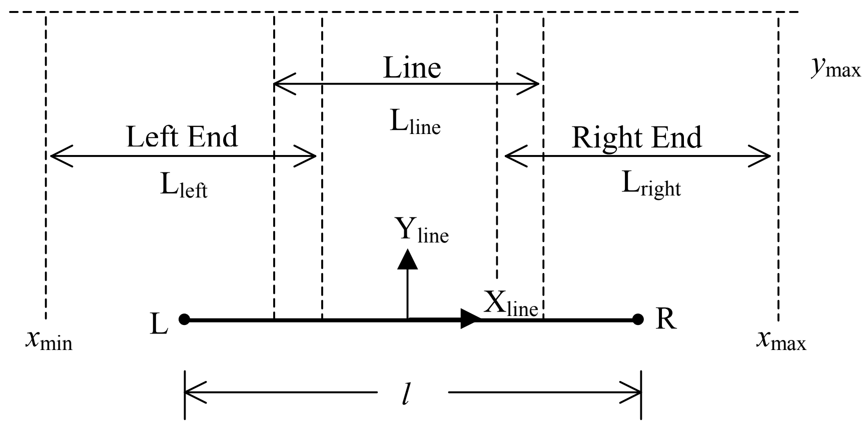


Figure 5.2 Three regions of a LCUF template.

Definition 5.2 The three regions are defined by:

Left End = $\{\ q=(x,y,\theta)\,|\,(x,y,\theta)\in[\frac{-(l+L_{left})}{2},\frac{(L_{left}-l)}{2}]\times(0,y_{\max}]\times(-\pi,\pi]\ \}$,

Line = $\{\ q=(x,y,\theta)\,|\,(x,y,\theta)\in[-\frac{l}{2},\frac{l}{2}]\times(0,y_{\max}]\times(-\pi,\pi]\ \}$, and

Right End = $\{\ q=(x,y,\theta)\,|\,(x,y,\theta)\in[\frac{(l-L_{right})}{2},\frac{(l+L_{right})}{2}]\times(0,y_{\max}]\times(-\pi,\pi]\ \}$.

From the definition, the width for each region, L_{left} or L_{right}, generally is selected to be larger than the maximum length in any direction of the robot polygon, and $y_{\max}$ is chosen to be the maximum width of a passage in the workspace. Once the boundary of each region is identified, the region is discretized along *x*, *y*, and *θ*-axis to obtain a finite number of configurations (called grid points) whose LCUs must be determined. After all the LCUs of every grid points have been determined by means of simulation discussed in Chapter 3, the LCUF of the grid can be used as a sub template for each region under consideration. Each sub template can now be used to estimate the value of LCU at any configuration *q* inside the region. This estimation is achieved by using linear interpolation between neighbor grid points of that configuration. Obviously, the accuracy of the LCU estimation depends highly on the size of discretization. However, we will show later that the high accuracy estimation can be obtained even though the size of discretization is coarse.

As mentioned earlier, the method of combining all three sub templates to form a single LCUF template must be derived. Based on observations from many simulations, we discover that the value of the LCU at any point in the overlap region (even overlapped by all three sub templates) always equal to the highest value of the three LCUs obtained from the three sub templates. For this reason, the equation used to combine all three sub templates together can be written in the following form.

$$LCUFtemplate = \underset{i\in[1,2,3]}{Max}\ SubTemplate_i\,, \tag{5.1}$$

where, $SubTemplate_1$ = Left End sub template.

$SubTemplate_2$ = Line sub template.

$SubTemplate_3$ = Right End sub template.

$LCUFtemplate$ = LCUF template of a line.

A contour plot of a LCUF template of a line for a general convex polygon robot is presented in Figure 5.3. The value of LCU at each position (x, y) in the plot is the maximum value of LCU at that configuration (x, y, θ) for all $\theta \in (-\pi, \pi]$. Notice that the LCUF template is not symmetric due to an asymmetric shape of the robot polygon, and the value of LCUs increase as the robot position move further away from the line. It is also clear that the LCUs stay constant in the middle region above the line. This is due to the effect of the Line sub template, which is mentioned previously.

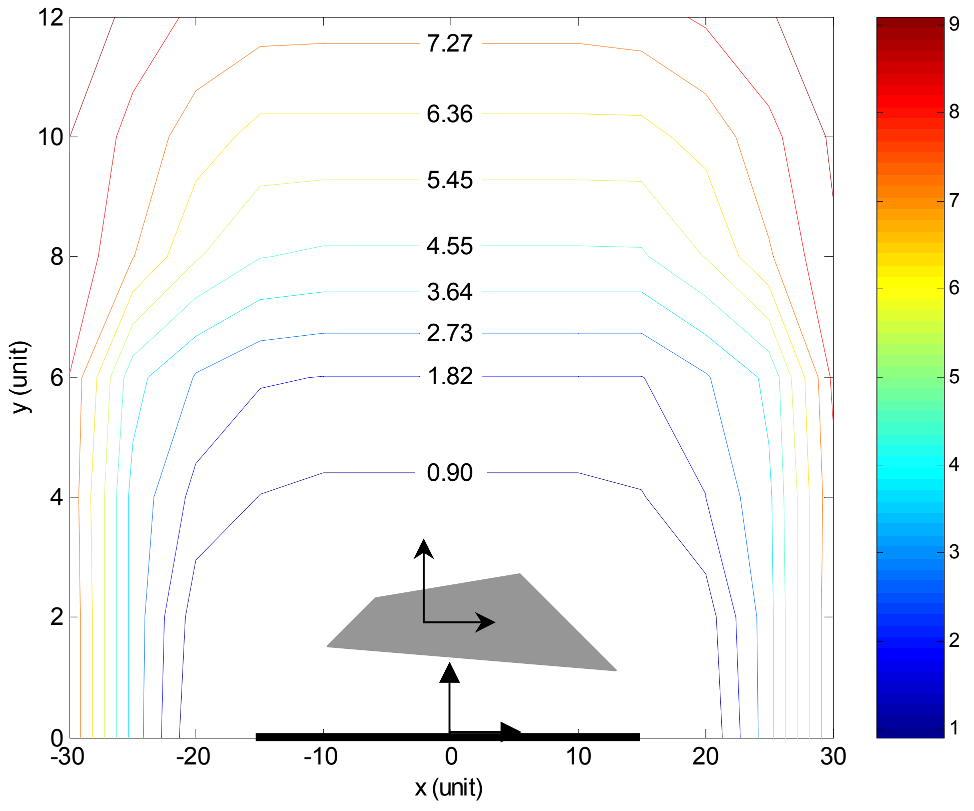


Figure 5.3 An example of a LCUF template of a convex robot.

The *LCUFtemplate* obtained in (5.1) can be used in determining the value of LCU at any configuration of a mobile robot by the following linear interpolation:

$$LCUline(q) = Interpolation[LCUFtemplate(q)] \tag{5.2}$$

Although the linear interpolation is simple, it produces satisfactory result in a very short period of time—the approximation accuracy will be given later.

Method to combine LCUs from many LCUF templates. Once we have obtained a LCUF template of a line, which can be used to determine the value of LCU of a robot configuration with respect to each line. All LCUs obtained from every line in a workspace can be combined with the following equation to give an approximation of the LCU at that particular configuration.

$$EstLCU(q) = \underset{k\in[1,N]}{Min}\ LCUline_k(q)\,, \tag{5.3}$$

where, $EstLCU(q)$ = an estimated LCU at configuration q.

$LCUline_k(q)$ = an interpolated LCU from line k at configuration q.

N = a total number of lines in a workspace.

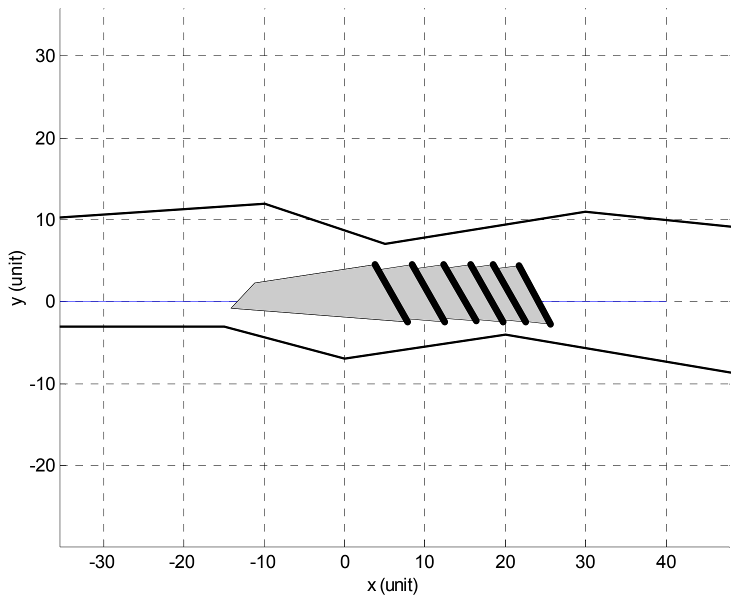
From (5.3), an estimated LCU at any configuration in a workspace is equal to the minimum value of interpolated LCUs obtained from every line in the workspace. It is worth noting that the equations (5.2) and (5.3) can be applied with a general convex polygon only. However, the LCU estimation method can be applied to a general concave polygon by approximating the concave polygon by a convex one. This helps expanding the applicability of the estimation method to cover every type of polygon. Although we lose any advantage obtained by the concavity, we do not wish to solve the related “Piano movers’ problem” in this work.

The proposed LCU estimation method has been employed to estimate values of LCUs in a simulation of a convex robot moving along a path in a general workspace, as shown in Figure 5.4(a). The robot moves in a straight line fashion from right to left. During the simulation, the exact value of LCUs for every configuration along the path are determined and compared to the estimated value of LCUs. Figure 5.4(b) presents the comparison between the exact LCU (solid curve) and the estimated LCU (dash-dot curve) over the simulation time (150 seconds). Clearly, the exact LCU curve is piecewise constant due to the discretization of the exact LCU in the simulation (which is set to 0.05). The simulation reveals very satisfactory result, where the estimated LCU curve closely tracks the exact LCU curve. The average approximation error—which is computed by averaging the percentage of error of the estimated LCU to the exact LCU of every configuration during the simulation—is 7 to 8 percent, which is impressive considering that we use coarse discretization in constructing the LCUF template (1 unit in x and y axes, $\pi/32$ in θ-axis, and 0.05 increment step for the exact LCU) and that a simple linear interpolation between grid points is used.

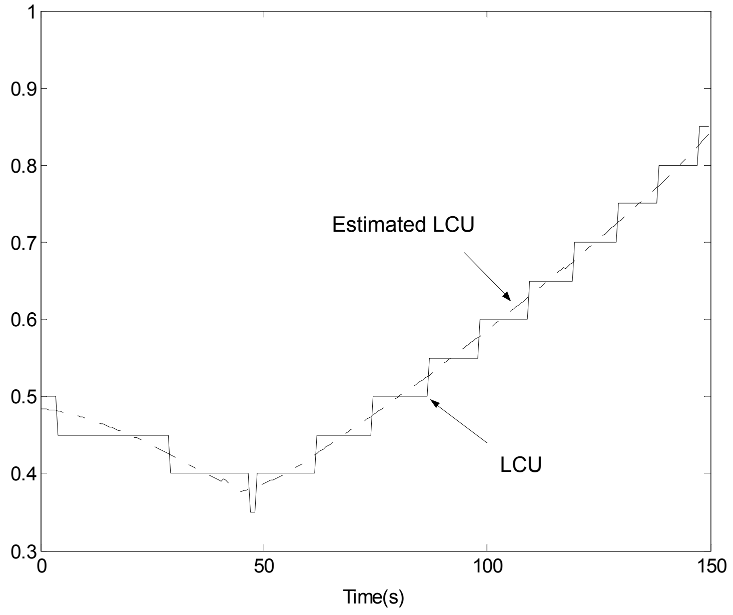
Moreover, the LCU estimation method requires much less computation time than the conventional method (presented in Chapter 3) does. This fact will be emphasized in the next section by showing through the overall improvement of the improved LCUF-based motion planning over the conventional one. In addition, the LCU computation time is now invariant to the configuration under consideration, and depends only on the complexity of workspace. The complexity in this case is defined by the number of line segments presented in a workspace. If a workspace consists of N line segments, the proposed LCU estimation method will require a constant $\mathcal{O}(N)$ time to compute.

It is also worth noting that, in general, two different *LCUFtemplate*s for a line is required for estimating LCUs of a nonsymmetrical polygon robot. One template is for the case of forward motion of the robot (using V and T). The other template is for the case of backward motion of the robot (using $-V$ and T).

In the next section, we apply the LCU estimation method to the existing LCUF-based motion planning proposed in Chapter 3. The goal is to reduce the optimization time required by the motion planner, and help improving its performance.



(a)



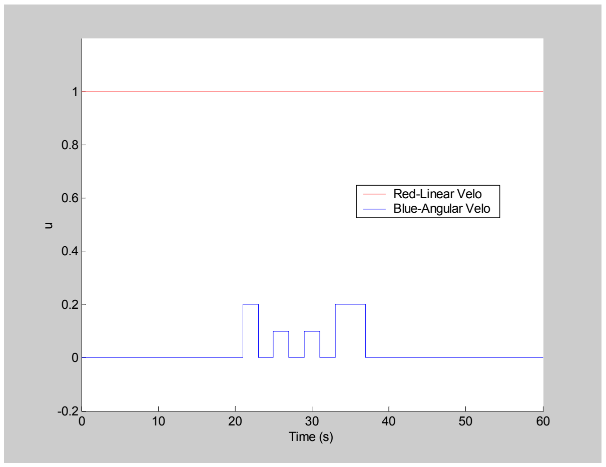
(b)

Figure 5.4 The estimation of LCUs along a path (a) a general convex polygon robot moving along a path in a workspace, (b) a plot of exact LCUs and estimated LCUs versus time.

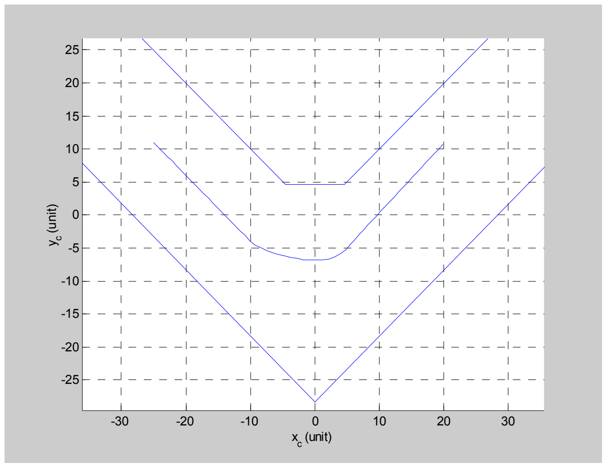
5.3 Improving LCUF-Based Motion Planning via Estimated LCUF

Proposed in the previous section, the LCU estimation method is applied to estimate the value of the LCU for the LCUF-based motion planning proposed in Chapter 3. The result from this modification is called the "Estimated LCUF-based motion planning". The purpose is to improve the performance of the motion planner by decreasing its search time. The LCU estimation method has been incorporated into the existing search program in Chapter 3, and the resultant search program, listed in Appendix B, has been applied to solve the same motion planning problem—an optimal path through a 90-degree passage. The problem setup is exactly the same except that the computation of LCU is now performed by the LCU estimation method. To show how the search evolves, the search program is set to report the results twice. One is at an hour after the search started, and the other is at the end of the search. Snap shots of evolving optimal path and control are shown in Figure 5.5.

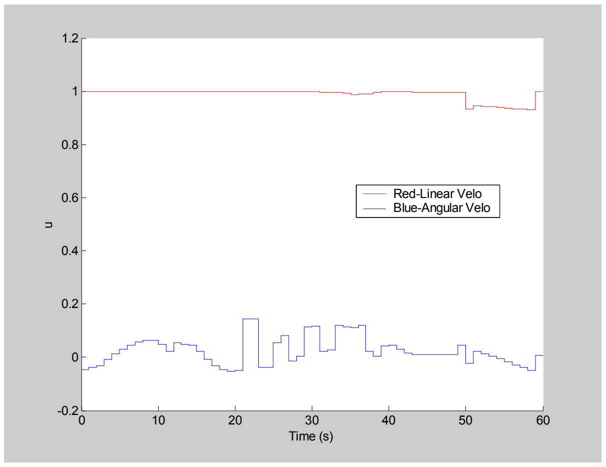
The search process starts from given initial condition for the control *u* as depicted in Figure 5.5(a) and its corresponding path in Figure 5.5(b). After one hour elapsed, the control and its corresponding path evolve into Figure 5.5(c) and 5.5(d), respectively. Notice that the algorithm optimize the path by move the right portion of the path toward the center of the straight passage and try to cut closer to the corner (apparently, the cut is too close, and it needs to be further optimized). At the end of the search, the optimal control and the optimal path—returned from the estimated LCUF-based motion planning—are presented in Figure 5.5(e) and 5.5(f), respectively. Although the optimal control obtained looks much different from the one obtained previously in Chapter 3 (Figure 3.7(c)), the optimal path looks similar to the one in Chapter 3 (Figure 3.8(a)). In fact, the optimal path in Figure 5.5(f) appears more symmetrical and smoother. The optimal path starts from the initial configuration $[-25, 11, -\pi/4]^T$ and ends at $[24.01, 10.34, \pi/3.97]^T$ near the goal configuration $[25, 11, \pi/4]^T$. Note that the optimal paths for both instances guide the robot closer to the corner, but do so with improved control uncertainty.



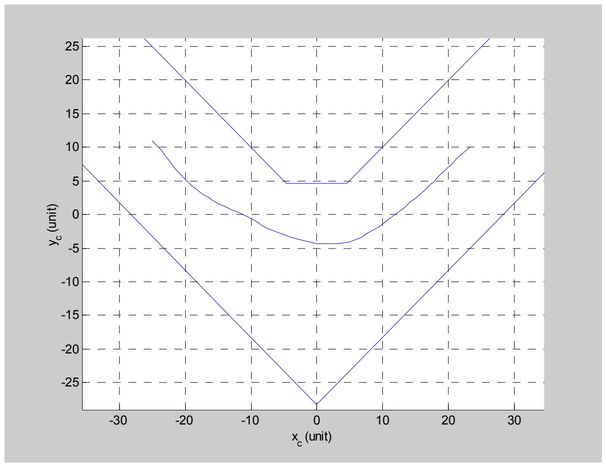
(a)



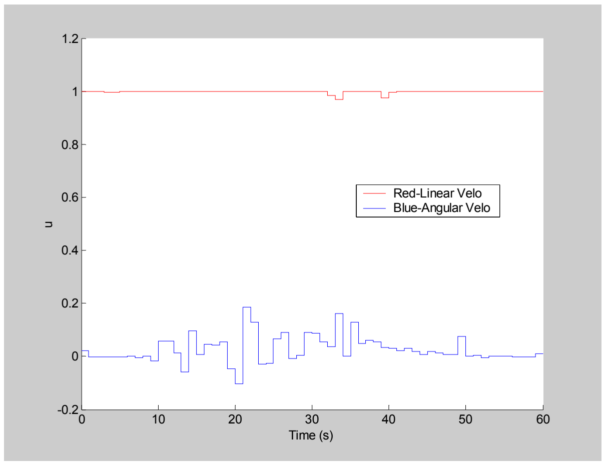
(b)



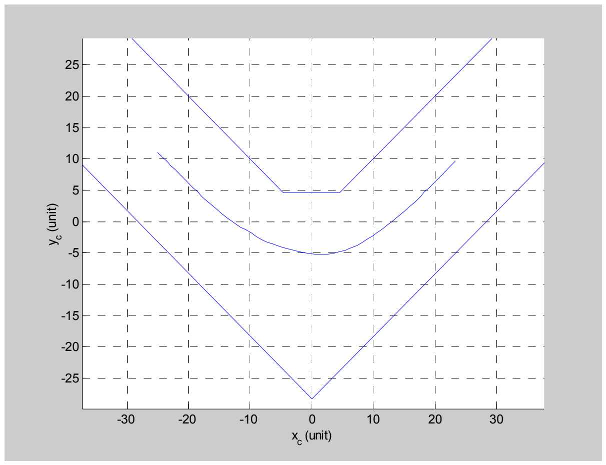
(c)



(d)



(e)



(f)

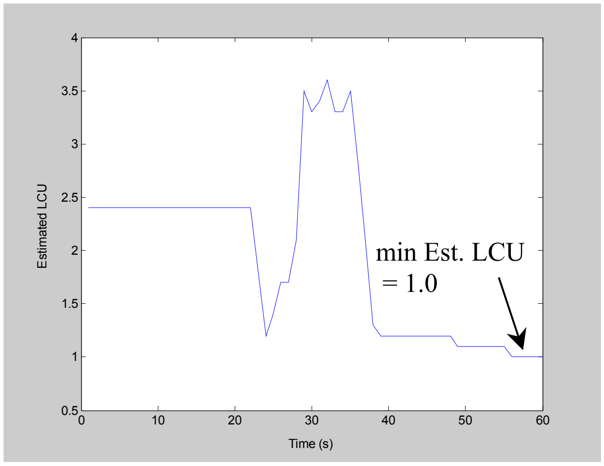
Figure 5.5 Snap shots of evolving controls and paths (a) control at t = 0, (b) corresponding path

at t = 0, (c) control at t = 1 hour, (d) corresponding path at t = 1 hour, (e) control at t = 6.44 hours , and (f) corresponding path at t = 6.44 hours.

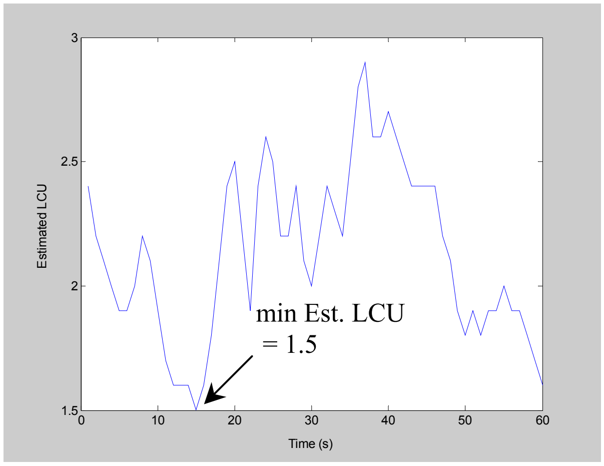
The estimated LCUF-based path planning demonstrates an impressive performance by taking only 6.44 hours of run time on a 1-GHz personal computer as oppose to 96 hours on a 1.7-GHz personal computer for the case of the LCUF-based path planning. If we conservatively assume that the same search program runs 1.25 times faster on the 1.7-GHz personal computer, it will take only 4.83 hours to complete one run. This equates to 19.87 times improvement in computation time. The improvement in computation time completely removes the weakness of the LCUF-based path planning in term of run time, and makes the LCUF-based path planning even more preferable over the CCUF-based path planning method.

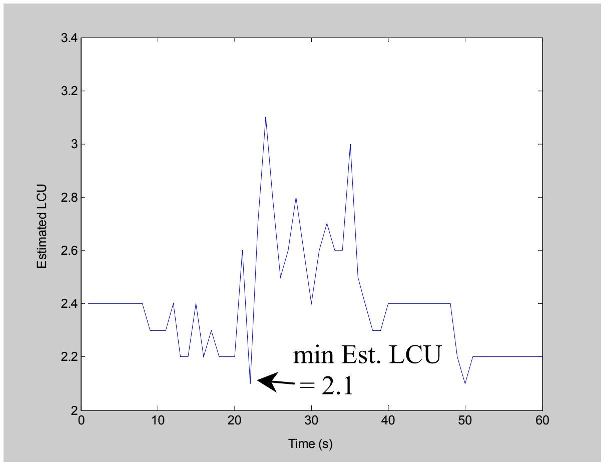
Figure 5.6 shows snap shots of the corresponding estimated LCUs along the evolving path in Figure 5.5. It is clear that the values of the estimated LCUs are optimized over the entire period of the search process. At time t = 0, the initial guess path yields the minimum estimated LCU of the path equal to 1.0. At time t = 1 hour, the minimum estimated LCU of the path becomes 1.5, which is equal to the minimum LCU of the optimal path attained in Chapter 3, see Figure 3.8(b). If one compares the paths in Figures 5.5(d) and 3.8(a) together, one will find that they both share a lot of similarity. This also reflects on the value of minimum LCU of both paths.

As the search continues to run, the path continues to evolve until the optimal path is attained at time t = 6.44 hours. The minimum LCU of the optimal path equals 2.1, which is an improvement over the previous optimal path obtained in Chapter 3. This is an indication of the sensitivity of the optimization method to the change of its cost function—which occurs due to using the estimated LCU in place of the exact LCU. As we can see from Figure 5.4(b), the estimated LCU curve is much less discontinuous than the exact LCU curve (which is discretized at 0.05 increment step). This discretization in the cost function causes the LCUF-based path planning to terminate prematurely since the value of the cost function has a dead-band of 0.05. On the other hand, due to less discontinuity of the estimated LCU curve (and so does the estimated LCUF), the estimated LCUF-based motion planning takes advantage of the well-behaved cost function, and is able to attain a better solution in much less time. One might suggest decreasing the discretization of the exact LCU to prevent the LCUF-base path planning from premature termination; however, the search time will be substantially increased. Clearly, the estimated LCUF-path planning not only runs faster, but also yields a better result.



(a)





(b) (c)

Figure 5.6 Snap shots of the estimated LCUs along the evolving path (a) at t = 0, (b) at t = 1 hour, and (c) at t = 6.44 hours.

The estimated LCUF has some interesting properties such as a global optimality property and a gradient property. In the next subsection, we study the global optimality characteristic of the estimated LCUF. The gradient property will be discussed later in Chapter 6.

Global optimality property of an estimated LCUF. We will prove that an estimated LCU from a LCUF template of a line is a radially unbounded function of x and y, where the value of the estimated LCU (or *LCUline*(x, y)) approaches infinity as either value of x or y approaches infinity. The result from this proof then is used to prove that there exists a unique location, in a two-dimensional workspace, in which a tube (in a three-dimensional configuration) around the location contains an optimal path.

Consider a convex polygon (representing a mobile robot) in a workspace containing only one line as shown in Figure 5.7(a). The vector indicates the robot position (x, y) is defined by r, and its magnitude is $\|r\|$. Further, let us fix the orientation of the robot to any value θ and translate the robot position, along vector r, toward or away from the origin of the line coordinate frame $\{XY_{line}\}$. It is clear that the value of the estimated LCU increase as $\|r\|$ increases, and this also applies to all vectors (in the upper plane) emanate from the origin of the line coordinate frame. As a result, the value of the estimated LCU is strictly increasing and approaches infinity as $\|r\|$ approaches infinity. This proves that the estimated LCU is a radially unbounded function of (x, y) with any constant θ.

At any position (x, y), there is at least one orientation θ_{m} that yields the maximum value of the estimated LCU. We know from the previous proof that the estimated LCU is a radially unbounded function of (x, y) with respect to any constant θ. As a result, one can always find a larger value of the estimated LCU for a larger value of $\|r\|$ regardless of the orientation θ_{m} of the configuration with smaller $\|r\|$. Hence, it can be concluded that the maximum estimated LCU of configuration (x, y, θ) for all $\theta \in (-\pi, \pi]$ is also a radially unbounded function, as clearly shown by the contour plot in Figure 5.3.

Next, consider an example of a simple contour plot of the maximum estimated LCUF of a line in Figure 5.7(b). This plot depicts a general representation of the maximum estimated LCUF. Each contour around the line represents an Isocontour passing through each point (x, y) possessing the same value of the maximum estimated LCUs in the maximum estimated LCUF. The next part of the proof is to prove that there is a unique location where the solution (the optimal path) will lie in its neighborhood.

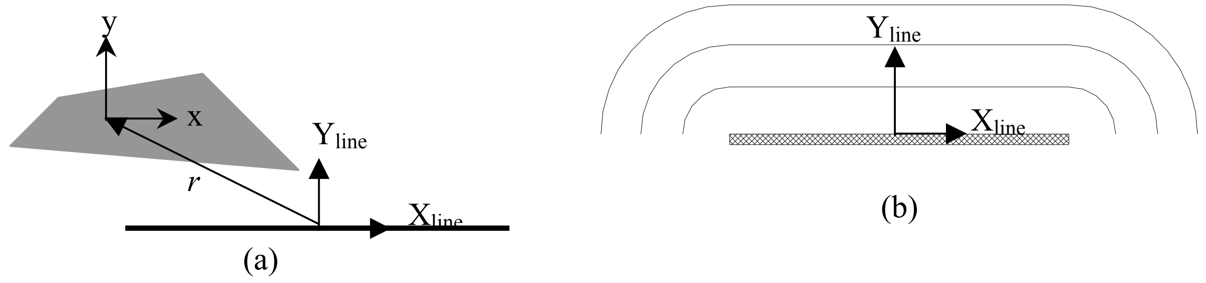


Figure 5.7 (a) Position vector, *r*. (b) Contour plot of the maximum estimated LCUF around a line.

Consider two connecting lines with their maximum estimated LCUFs shown in Figure 5.8(a). After using the rule (5.3) to combine the overlap region, the result obtained is presented in Figure 5.8(b). Intuitively, an optimal path in the case of Figure 5.8(b) is the one that furthest away from both lines. In a general workspace comprising *N* lines, the same rule applies to all of the lines in the workspace. Figure 5.8(c) shows an example of the resultant maximum estimated LCUF for a general workspace with four lines. The closed area in the middle of the four lines indicates the peak of a valley located between those lines. The ridge along the valley indicates a unique location where an optimal path should lies, represented by a dash spline going through the maximum estimated LCUF, as shown in Figure 5.8(d).

Recall that the dashed spline is obtained from the maximum estimated LCUF (in which the orientation θ is temporary removed). Once the orientation must also be considered, a candidate path should still stay close to the dashed spline to be an optimal path. Due to the nonholonomic constraint, a feasible candidate path might not be exactly the same as a path suggested by the dash spline; however, it should be in the vicinity of the dash spline. Therefore, to take care of the orientation, we expand (inflate) the dash spline with a radius of ε equally around the spline—this expanded spline will be called an ε-tube—and use this ε-tube to indicate a unique three-dimensional (x, y, θ) region containing the optimal path, see Figure 5.8(e). Since there is only one such compact region that contains the optimal path, we conclude that an estimated LCUF of any workspace contains a unique global optimal path. The same conclusion can be drawn for the case of the exact LCUF introduced in Chapter 3. The proof confirms the optimality of the optimal path discovered in section 3.5.

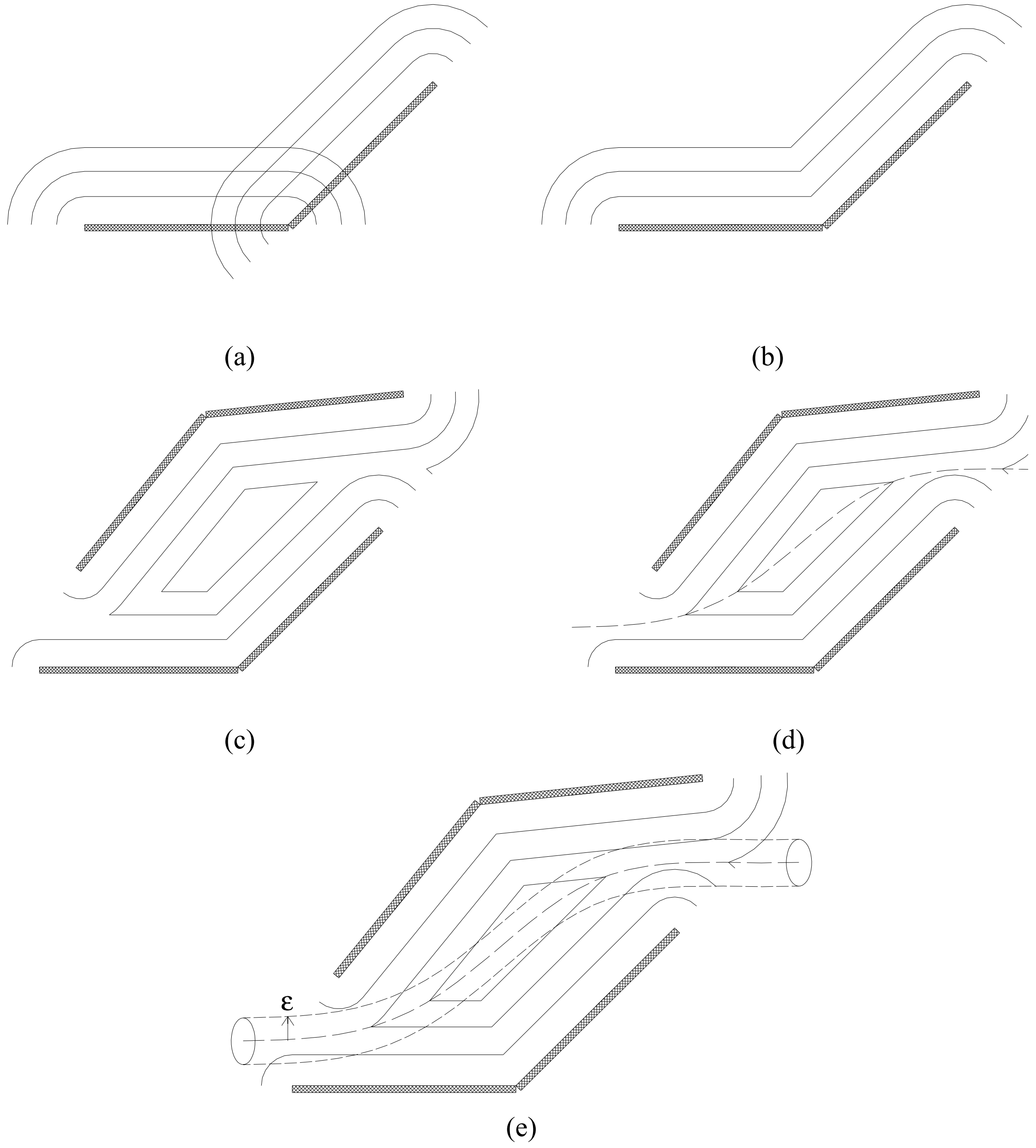


Figure 5.8 (a) Two maximum estimated LCUFs before combining, (b) after combining, (c) maximum estimated LCUF for a general workspace, (d) possible location for an optimal path, and (e) an ε-tube of the possible location.

An example of the maximum estimated LCUF in a general case (a general convex polygon robot in a general workspace as shown in Figure 5.4(a)) is presented in Figure 5.9. It is obvious that the contour plot of the maximum estimated LCUF of the workspace in this case is not as orderly as in the ideal situation as depicted in Figure 5.8(e). For this reason, with only the contour plot, the shape of the ε-tube in this case can no longer be determined intuitively. An alternative is to use the quiver plot which shows a field of vectors indicating the configuration as well as its maximum estimated LCU value (represented by the length of the vector), see Figure 5.10. The thick arrows indicate the maximum estimated LCU along y-axis (when *x*-coordinate is fixed constant and limit $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$), and they clearly suggest the location of the ε-tube. The size of ε depends on the shape of the robot in the workspace, and cannot exceed the width of the passage.

5.4 Conclusion

The comparison of the performance of both motion planning techniques proposed in previous chapters are considered. The LCUF-based motion planning is selected as a better one, but its weakness is determined to be high search time. The weakness is completely eliminated by the introduction of an approximation scheme used to estimate the value of LCU. This approximation scheme can accurately estimate the value of LCU (with error less than 7 to 8 percent), and helps reducing search time by more than an order of magnitude of the computation time required by the conventional LCUF-based motion planning. Moreover, the new motion planning (estimated LCUF-based motion planning) gives better results due to less discontinuity in the estimated LCUF. The estimated LCUF, as well as the exact LCUF, is proved to have unique global optimal region, which contains the optimal path. This region is referred to as “the ε-tube”, which can be approximately located by the plot of the maximum estimated LCUF. This proof guarantees the optimality of a resultant path returned from the motion planner.

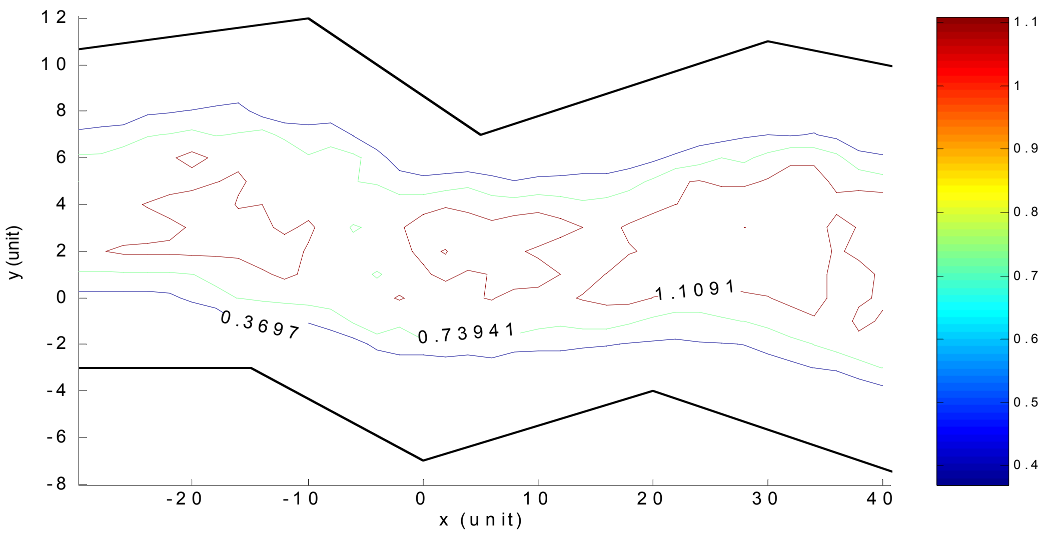


Figure 5.9 The maximum estimated LCUF of the robot and the workspace in Figure 5.4(a).

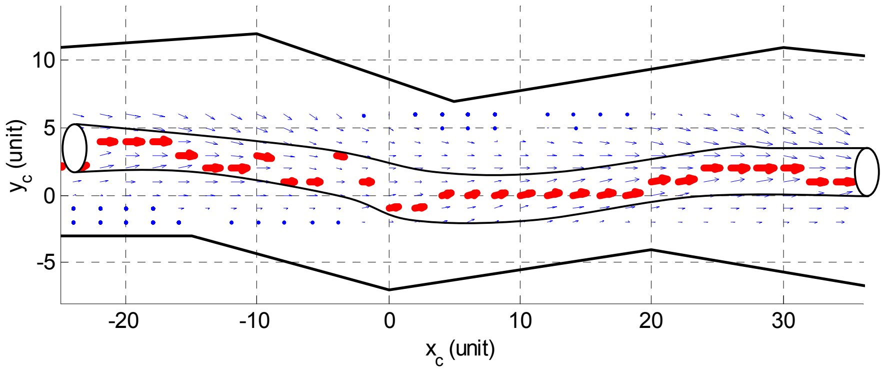


Figure 5.10 The ε-tube for the maximum estimated LCUF in Figure 5.9.

Chapter 6

Robust Path Following Control under State Constraints

As shown in the literature review, there is no path following control for a nonholonomic mobile robot that takes into account both uncertainties and the state constraints posed by obstacles in a workspace. In this section, we develop the first real-time path following control algorithm that guarantees collision-free operation during path following. The development begins with the design of a robust path following control, which is robust to sensory uncertainty. At this design stage, the state constraints due to obstacles in a workspace are totally neglected. Then, we integrate the state constraints to the obtained robust path following control by the usage of the LCUF. This LCUF helps the robust controller in determining safe linear velocity throughout the entire motion of the robot. It also prevents the robust controller from steering the robot toward obstacles while the robust controller makes sure that the robot follows the path. The resultant control is called the "LCUF-based robust path following control". The simulation results show the effectiveness of this path following control.

6.1 Robust Path Following Control Design

Once an optimal path has been planned, a mobile robot must track or follow the optimal path as accurate as possible to ensure safe navigation in a workspace. Moreover, many collision-free conditions can be concluded by Propositions given in Chapters 3 and 4 if the mobile robot tracks or follows the optimal path exactly. As we mentioned earlier in the literature review that there are two main tracking schemes for mobile robots (trajectory tracking and path following), we must first choose which scheme is the more desirable one. The objective of trajectory tracking is to steer the robot from its current configuration to a target configuration at a specified time; therefore, a target configuration or a reference configuration changes over time. Thus, there is a chance that the error between the initial and the target configurations can grow up to some value that can cause instability in the feedback control. For example, the robot might get trapped and cannot move, while the trajectory generator keeps generating new target configurations (which is further away). The error is built up over time, and eventually causes the instability in the control system.

The objective of path following, on the other hand, is to bring the robot from its current configuration to a target configuration specified by the arc length along the path. This means that the target configuration will not change over time, but will change only when the robot moves (when initial configuration changes). Thus, the path following does not suffer from the built-up error problem. For this reason, we have chosen the path following scheme over the path tracking scheme.

Robust Path Following Control. There are many nonlinear feedback control laws, both static and dynamic ones, proposed in literature to stabilize system (3.1) around a given path. Each of them is slightly different in terms of stabilization type, convergence rate, robustness, and complexity. Inspired by Aguilar's work [Aguilar, et al., 1998], a new feedback control law for path following control is investigated. Using the model given in [Samson, 1992] to represent the system (see Figure 6.1), the origin of a moving coordinate frame *{xy},* point *R*, is located at the orthogonal projection of the current robot configuration, point *C*, on the reference path defined in the global coordinate frame *{XY}*. The orientation of frame *{xy}* is tangential to the path at point *R*. Point *R* can be taught as a reference configuration, toward which the current configuration must be steered. With respect to frame *{xy}*, the configuration error $P_e = P_c\text{-}P_R$ is expressed as:

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_R(x_c - x_R) + \sin\theta_R(y_c - y_R) \\ -\sin\theta_R(x_c - x_R) + \cos\theta_R(y_c - y_R) \\ \theta - \theta_R \end{bmatrix} \tag{6.1}$$

Taking the time derivative and rearranging terms, we have:

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} \omega_R y_e - v_R + v\cos\theta_e \\ -\omega_R x_e + v\sin\theta_e \\ \omega - \omega_R \end{bmatrix} \tag{6.2}$$

Due to the property of the orthogonal projection, both x_e and $\dot{x}_e$ always equal zero as the robot moves. Thus, from the first row of equation (6.2) we obtain:

$$v_R = y_e\omega_R + v\cos\theta_e \tag{6.3}$$

Substituting $\omega_R = \kappa_R v_R$ into equation (6.3), where κ_R is the reference path curvature, we get:

$$v_R = \left(\frac{\cos\theta_e}{1-\kappa_R y_e}\right)v \tag{6.4}$$

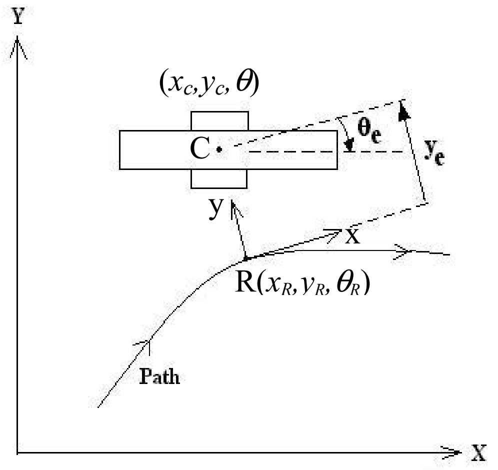


Figure 6.1 Path following parameters.

Equation (6.4)holds as long as $1-\kappa_R y_e > 0$ or, equivalently, $y_e < r_R$, where r_R is a reciprocal of κ_R (i.e., a radius of the reference path). This constraint means that the shortest distance to the path must be smaller than the path radius, so that the moving frame *{xy}* can be uniquely defined. From equation (6.2), only the last two rows are considered, as the first row is always zero. Therefore, the system equation is reduced to:

$$\begin{bmatrix} \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} v\sin\theta_e \\ \omega_e \end{bmatrix}, \tag{6.5}$$

where $\omega_e = \omega - \omega_R$.

Given the linear velocity, v, the path following control has been turned into the problem of determining a feedback control, ω_e, to stabilize the system (6.5) to the origin (i.e., (y_e,θ_e) = (0,0)).

In determining a suitable feedback control law, we began by studying a control law proposed by Aguilar [Aguilar, et al., 1998]. We found that Aguilar's control law has a major drawback in that the stability of the closed-loop system cannot be guaranteed if the system error is large. Thus, we improved the stability of the system by modifying Aguilar's control law, and the obtained control law is as follows.

$$\omega_e = -v\cdot[\frac{k_1}{\ell}\tan^{-1}(\frac{y_e}{\ell})] - k_2\left|\frac{v}{\ell}\right|\theta_e \tag{6.6}$$

where, k_1 and k_2 = positive controller gains.

ℓ = characteristic length.

The above control law is proposed to stabilize the system (6.5) in the domain of $y_e \in \mathbb{R}$ and $\theta_e \in (-\pi,\pi)$. An arc tangent function is introduced in (6.6) to prevent the magnitude of the feedback control from being too large due to y_e; hence, the closed-loop stability is always guaranteed. The characteristic length in the control law (6.6) is the length measure from the center point on the wheel axis of the robot to the farthest point on the perimeter of the robot. Next, we prove that the control law (6.6) stabilizes the system (6.5) using Lyapunov's direct method. Let a Lyapunov function be:

$$V(y_e,\theta_e)=\frac{k_1}{\ell}[y_e\tan^{-1}(\frac{y_e}{\ell})-\frac{\ell}{2}\ln(1+(\frac{y_e}{\ell})^2)]+(1-\cos\theta_e) \tag{6.7}$$

This Lyapunov function is continuously differentiable over the domain $y_e \in \mathbb{R}$ and $\theta_e \in$ (-π,π), and it is locally positive definite and locally decrescent (according to Definition 2.8). The first term of the Lyapunov function is derived from taking indefinite integral of the first term (in the closed bracket) of equation (6.6), so that we will get the arc tangent term back when we take the time derivative of equation (6.7). The cosine in the second term of equation (6.7) is introduced in the Lyapunov function to help canceling common terms, which are shown up later in equation (6.10). For now, taking the time derivative of equation (6.7), we get:

$$\dot{V}(y_e,\theta_e)=\frac{k_1}{\ell}\tan^{-1}(\frac{y_e}{\ell})\cdot\dot{y}_e+\sin\theta_e\cdot\dot{\theta}_e \tag{6.8}$$

Substituting equation (6.5) into equation (6.8), we have:

$$\dot{V}(y_e,\theta_e)=\frac{k_1}{\ell}\tan^{-1}(\frac{y_e}{\ell})\cdot v\sin\theta_e+\sin\theta_e\cdot\omega_e \tag{6.9}$$

Substituting equation (6.6) into the equation above, we obtain:

$$\dot{V}(y_e,\theta_e)=\frac{k_1}{\ell}\tan^{-1}(\frac{y_e}{\ell})\cdot v\sin\theta_e+\sin\theta_e\cdot[-\frac{v}{\ell}(k_1\tan^{-1}(\frac{y_e}{\ell}))-k_2\left|\frac{v}{\ell}\right|\theta_e] \tag{6.10}$$

Canceling common terms, we arrive at:

$$\dot{V}(y_e, \theta_e) = -k_2 \left| \frac{v}{\ell} \right| \theta_e \sin \theta_e \tag{6.11}$$

The right hand side of the above equation is negative definite for all $y_e \in \mathbb{R}$ and $\theta_e \in (-\pi,\pi)$. From Lyapunov's stability theorem (Theorem 2.4), we can conclude that the closed-loop system is locally uniformly asymptotically stable in the domain $y_e \in \mathbb{R}$ and $\theta_e \in (-\pi,\pi)$ around the origin. From the control law (6.6), k_1 and k_2 are positive controller gains that can be chosen to control the shape of the system trajectory from any initial configuration toward the origin, and v reflects the speed that the robot travels along the trajectory.

However, the design of the control law (6.6) above does not take into account the effect of sensory uncertainty. We still need to prove that the control law is still be able to stabilize the system (6.5) even if the feedback state entering the control law is corrupted by some noise. Let $(\hat{y}_e, \hat{\theta}_e)$ be the estimated state from the sensor measurement corrupted by sensor noise, and is defined by the following equation.

$$\begin{bmatrix} \hat{y}_e \\ \hat{\theta}_e \end{bmatrix} = \begin{bmatrix} y_e + \delta y_e \\ \theta_e + \delta\theta_e \end{bmatrix}, \quad \begin{matrix} \delta y_e \in [-\Delta y_e, \Delta y_e] \\ \delta\theta_e \in [-\Delta\theta_e, \Delta\theta_e] \end{matrix} \tag{6.12}$$

where, δy_e and $\delta\theta_e$ are sensing errors in y_e and θ_e due to sensory uncertainty. The positive constants Δy_e and $\Delta\theta_e$ are bound of sensing error δy_e and $\delta\theta_e$, respectively. Replacing (y_e, θ_e) in the control law (6.6) by $(\hat{y}_e, \hat{\theta}_e)$, we obtain:

$$\hat{\omega}_e = -v \cdot [\frac{k_1}{\ell} \tan^{-1}(\frac{\hat{y}_e}{\ell})] - k_2 \left| \frac{v}{\ell} \right| \hat{\theta}_e, \tag{6.13}$$

which is the control law with the estimated state. To prove that the system (6.5) is still stable through the control (6.13), we must prove that $\dot{V}(y_e, \theta_e)$ is still negative definite on some domain to conclude stability over that domain. From equation (6.9), replacing ω_e with $\hat{\omega}_e$, we get:

$$\dot{V}(y_e,\theta_e) = \frac{k_1}{\ell}\tan^{-1}(\frac{y_e}{\ell})\cdot v\sin\theta_e + \sin\theta_e \cdot \left\{ -\frac{v}{\ell}(k_1\tan^{-1}(\frac{\hat{y}_e}{\ell})) - k_2\left|\frac{v}{\ell}\right|\hat{\theta}_e \right\} \tag{6.14}$$

Rearranging equation (6.14), we obtain:

$$\dot{V}(y_e,\theta_e) = \frac{k_1}{\ell}v\sin\theta_e\left[\tan^{-1}(\frac{y_e}{\ell}) - \tan^{-1}(\frac{\hat{y}_e}{\ell})\right] - k_2\left|\frac{v}{\ell}\right|\hat{\theta}_e\sin\theta_e \tag{6.15}$$

Notice that in the case of the perfect measurement, the first terms would cancel each other out, and we would get the same result as in (6.11). To make the right hand side of the equation (6.15) be negative definite, k_2 must be chosen to satisfy the following condition.

$$k_2 > k_1\left|\frac{\tan^{-1}(\frac{y_e}{\ell}) - \tan^{-1}(\frac{\hat{y}_e}{\ell})}{\hat{\theta}_e}\right| \quad \text{for} \quad |\theta_e| > \Delta\theta_e \tag{6.16}$$

This means that the second term in equation (6.15), which is always negative when $|\theta_e| > \Delta\theta_e$, must dominate the first term to make the right hand side be negative definite. However, the same guarantee cannot be concluded in the domain where $|\theta_e| < \Delta\theta_e$ because the sign of $\dot{V}(y_e,\theta_e)$ is unpredictable due to the uncertainty. Although the system under uncertainty can no longer be said to be locally uniformly asymptotically stable, by invoking Theorem 2.6, we can at least conclude that the system (3.1) and the feedback control law (6.6) with the estimated measurement (6.12) in the domain $y_e \in \mathbb{R}$ and $\theta_e \in (-\pi,\pi)$ is uniformly bounded with respect to the bounded set $y_e = 0$ and $|\theta_e| < |\delta\theta|$. This proves the robustness of the proposed control law (6.6) to the sensory uncertainty.

According to (6.16), k_2 is measurement dependent. However, in general, one can choose k_2 to be a constant by using (6.16) to compute the value of k_2. Given uncertainty bounds Δy_e and

$\Delta\theta_e$, one can determine the value of k_2 by evaluating a largest possible value of the term $\tan^{-1}(\frac{y_e}{\ell}) - \tan^{-1}(\frac{\hat{y}_e}{\ell})$ in the absolute value sign in (6.16) according to the given uncertainty bound Δy_e. A possible value of $\hat{\theta}_e$ can be calculated from (6.12) by selecting a value of θ_e satisfying $|\theta_e| > \Delta\theta_e$, which means that $\hat{\theta}_e$ can be small but never be zero.

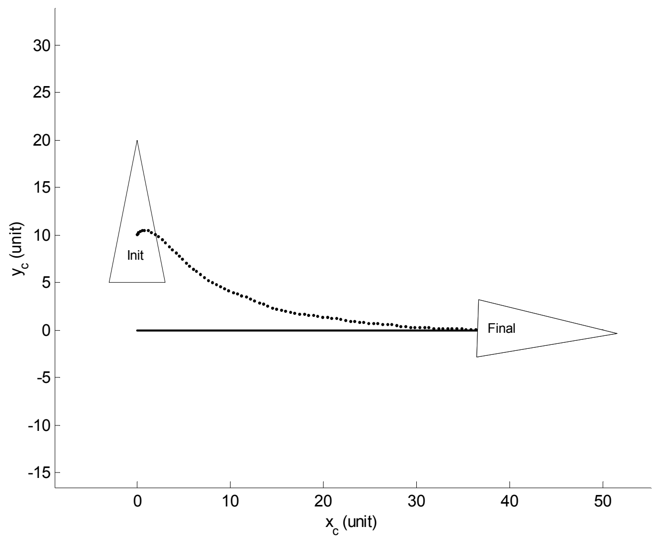
One important characteristic of the control law (6.6) is that the shape of the trajectory of the system (3.1) is only a function of the controller gains, k_1 and k_2, and the sign direction of the linear velocity. This can be clearly seen by rearranging equation (6.6), we obtain:

$$\frac{\omega_e}{v} = \kappa = -\frac{1}{\ell}(k_1 \tan^{-1}(\frac{y_e}{\ell})) - k_2 \frac{sign(v)}{\ell}\theta_e . \tag{6.17}$$

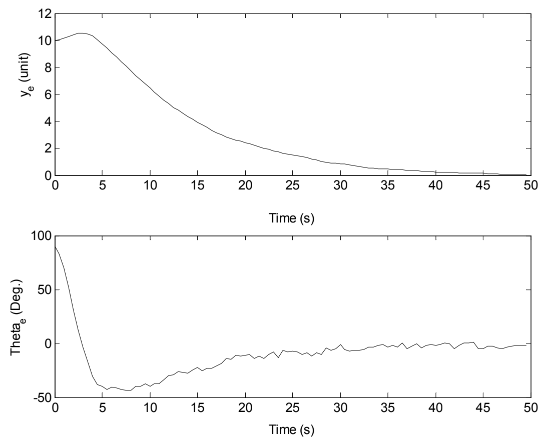
where, $\kappa =$ trajectory curvature.

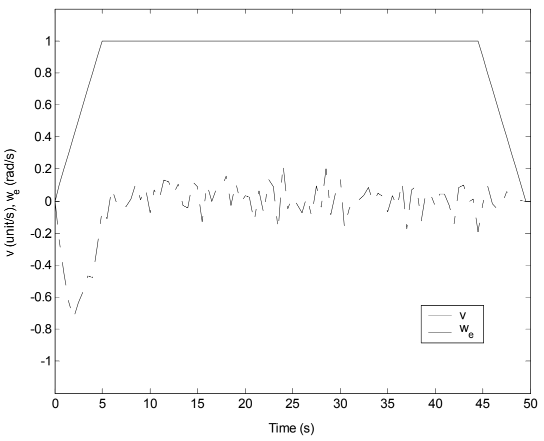
The above equation shows that the curvature of the system at any (y_e,θ_e) depends only on the values of the controller gains and the sign direction of the linear velocity. The magnitude of the linear velocity does not affect the shape of the system trajectory at all. However, both controls v and ω_e are coupled through equation (6.17). This means that at any instant the controls must be within given bounds, and also respect the equation (6.17).

Simulations. The synthesis and analysis of the proposed robust path following control law are confirmed through simulations, the MATLAB® program listed in Appendix B. The first simulation is a straight path following, where the robot initial condition is set far apart from its straight reference path at (y_e,θ_e) = (10, π/2). Let uncertainty bounds Δy_e and $\Delta\theta_e$ be 1 unit and 0.1 radian, respectively. Let the characteristic length $\ell = 10$ units. The controller gain k_1 is chosen to be 10, and the controller gain k_2 is calculated from (6.16) to be 10. The controls v and ω_e are bounded within [-1, 1] unit/second and [-1, 1] rad/second, respectively. The total simulation time is 50 seconds, and the simulation results are shown in Figure 6.2. The plot of the robot trajectory from its initial configuration to its final configuration is shown in Figure 6.2(a). The evolving state y_e and θ_e over the simulation time is presented in Figure 6.2(b). The steady-state error in y_e is within [-0.3, 0.3] units, and the steady-state error in θ_e is within [-0.13, 0.13] radian. These steady-state error results clearly agree with the analysis result obtained previously in this section.



(a)





(b) (c)

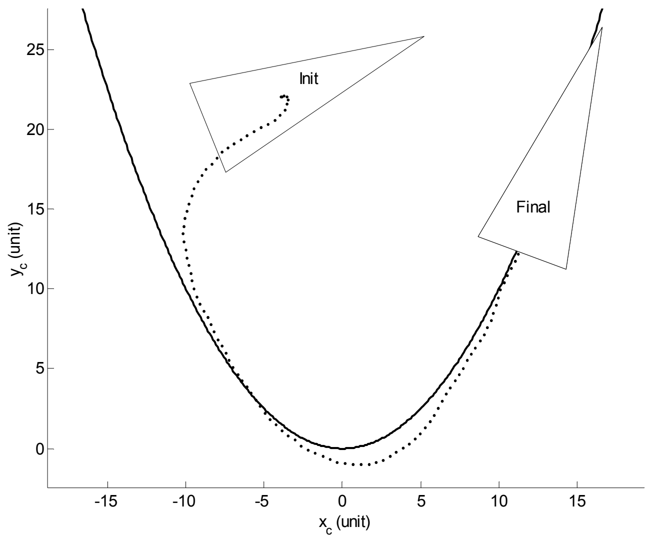
Figure 6.2 The simulation result for a straight path following case (a) the robot trajectory, (b) the system state y_e and θ_e versus time, and (c) the controls v and ω_e versus time.

Figure 6.2(c) shows the plot of the controls v and ω_e versus time. The control v is selected to ramp up (with acceleration +0.1 unit/s^2) from zero at t = 0 second, stay constant at +1 unit/second, and ramp down (with deceleration – 0.1 unit/s^2) to zero again at t = 50 second. The control ω_e becomes negative for the first 5 seconds, and fluctuates around zero after that due to sensory uncertainty.

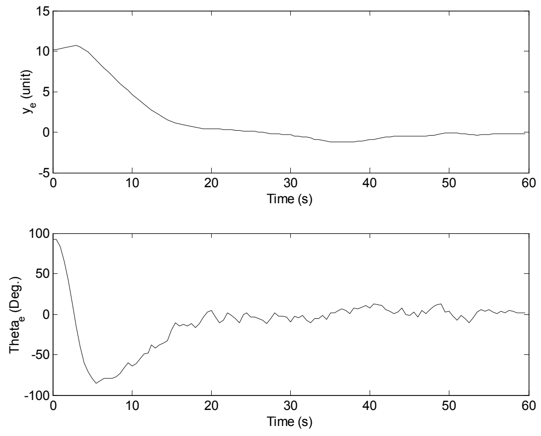
The second simulation is a free-form path following. In this simulation, the robot must start away from a polynomial path at $(y_e, \theta_e) = (10, \pi/2)$, and follow the path for 60 seconds. All others parameters are set to the same values as in the first simulation except that the controller gain k_1 is now set to be 20, while the controller gain k_2 is still 10. The reason is to improve convergence rate in the state y_e of the system. The simulation results are presented in Figure 6.3.

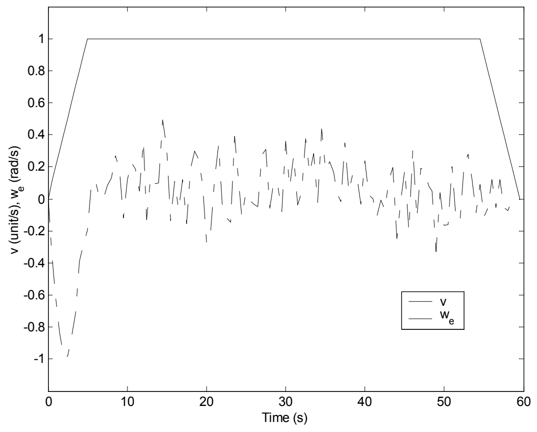
Figure 6.3(a) depicts the robot trajectory along with the polynomial path. The robot illustrates impressive path following capability even with the presence of the sensory uncertainty. This can be clearly seen from the plot of the system state over time in Figure 6.3(b), where the steady-state error in y_e is less than +/- 1.3 units and that in θ_e is less than +/- 0.23 radian. However, the control ω_e in Figure 6.3(c) is on average larger in magnitude than the one in the straight path following case because a larger control effort is required by the robot to follow the polynomial path accurately.

The two simulations have demonstrated the effectiveness of the proposed robust control law (6.6) in controlling a nonholonomic mobile robot to follow a given path. Next, we will use this control law to develop a real-time path following control algorithm for nonholonomic mobile robots subject to state constraints and uncertainties. By utilizing the inherent robustness property of the control law, a more useful and complex path following control algorithm can be derived. The control algorithm guarantees collision-free condition, and can be run in real-time.



(a)





(b) (c)

Figure 6.3 The simulation result for a free-form path following case (a) the robot trajectory, (b) the system state y_e and θ_e versus time, and (c) the controls v and ω_e versus time.

6.2 LCUF-Based Robust Path Following Control

The concept is to obtain the information regarding obstacles in a workspace, including with control uncertainty, and feed these information to the path following controller, so that an appropriate adjustment to the controller can be made. Recall that the LCUF is a field of LCUs over a free configuration space, each LCU essentially represents how far the robot is from the obstacles along some nonholonomic distance and how much control uncertainty is allowed for the robot to move safely in one time period. This means that the information about the obstacles in the workspace can be directly deduced from the value of LCU, and so is the LCUF. In the following subsections, each component of a new robust path following control algorithm—which we call the “LCUF-based robust path following control”—will be discussed, and the algorithm as a whole will be given at the end of this section. The words “LCU” and “estimated LCU” (or “LCUF” and “estimated LCUF”) are used interchangeably, and both words means the linear control uncertainty (or the linear control uncertainty field) calculated by the approximation method proposed in Chapter 5.

Determination of the linear velocity. The proposed robust control law (6.6) requires the linear velocity v as one of its inputs. Most of the prior path following control schemes assumes that the curve of the linear velocity versus time is supplied to the control law by the user. In our work, the user is only required to specify velocities at initial and final time and the direction of motion in between, the rest is automatically determined by the following linear velocity determination scheme to ensure safe navigation.

At any free configuration $q \in \mathcal{C}_{free}$, the value of LCU can be used to determine the value of the linear velocity, v. From the definition of LCU in Chapter 3, the actual controls or the wheel velocities v_r and v_l in (3.2) can take on any value in the following bounds.

$$\begin{aligned} v_r &\in [V\cdot(1-LCU) \quad V\cdot(1+LCU)] \\ v_l &\in [V\cdot(1-LCU) \quad V\cdot(1+LCU)] \end{aligned} \tag{6.18}$$

These bounds correspond to possible linear velocities and robot trajectory curvatures as follows.

$$\begin{aligned} v &\in [V\cdot(1-LCU) \quad V\cdot(1+LCU)] \\ \kappa_{LCU} &\in [-\frac{2\cdot LCU}{B} \quad \frac{2\cdot LCU}{B}] \end{aligned} \tag{6.19}$$

Notice that the range of κ_{LCU} is only a function of *LCU* and the robot wheel base *B*. Without loss of generality, we pick $v = V$ to be the only feasible linear velocity along every trajectory realized by possible wheel velocities (6.18) because we would like to span the range of trajectory curvatures throughout the set κ_{LCU}. This means that any robot traveling from a free configuration *q* with linear velocity *V* along a trajectory with curvature within the set κ_{LCU} for a period of time *T* is guaranteed not to collide with any obstacle in a workspace. However, at the same free configuration, the trajectory curvature κ calculated by the proposed control law (6.6) might not be within κ_{LCU}. Even within κ_{LCU}, one must ensure that the built-in control uncertainty (which results from both sensory and control uncertainty) of the robot system will not perturb the calculated trajectory until the trajectory curvature lies outside the set κ_{LCU}. If such event occurs, the linear velocity must be reduced to increase the value of LCU, which in turn expands the range of trajectory curvature κ_{LCU} in (6.19) to accommodate the set of perturbed calculated trajectory curature, κ_{robot}.

Let the maximum overall control uncertainty of a robot system due to both sensory and control uncertainties (from internal and external sources) be a positive constant CU_{robot} applying to both wheels of the robot. At any free configuration *q* and a trajectory curvature κ computed by equation (6.17), possible trajectory curvatures perturbed by CU_{robot} are within the following bound.

$$\kappa_{robot} \in [\frac{2\cdot(B\cdot\kappa - 2\cdot CU_{robot})}{B(2 - B\cdot\kappa\cdot CU_{robot})} \quad \frac{2\cdot(B\cdot\kappa + 2\cdot CU_{robot})}{B(2 + B\cdot\kappa\cdot CU_{robot})}] \tag{6.20}$$

One must make sure that $\kappa_{robot} \subseteq \kappa_{LCU}$ for the collision-free condition can be stated. Given a value of $CU_{robot} \in (0, 1)$, at any sampling period κ is determined by the equation (6.17), and the set κ_{robot} can be calculated and compared to κ_{LCU} in (6.19). If $\kappa_{robot} \subseteq \kappa_{LCU}$, the maximum value of the linear velocity v is *V*. However, if $\kappa_{robot} \not\subset \kappa_{LCU}$, we must at least have $\kappa_{robot} = \kappa_{LCU}$. By equating κ_{LCU} in equation (6.19) and κ_{robot} in equation (6.20) together, the following equations are obtained:

$$LCU_1 = \frac{(2 \cdot CU_{robot} - B \cdot \kappa)}{(2 - B \cdot \kappa \cdot CU_{robot})}$$

$$LCU_2 = \frac{(2 \cdot CU_{robot} + B \cdot \kappa)}{(2 + B \cdot \kappa \cdot CU_{robot})} \tag{6.21}$$

Equation (6.21) can be used to determine the value of *LCU* for κ_{LCU} in (6.19) by choosing from the larger control uncertainty between LCU_1 and LCU_2. There are two extreme cases worth mentioning here. When $\kappa = 0$ (for a straight trajectory), the equation (6.21) becomes:

$$LCU = LCU_1 = LCU_2 = CU_{robot}. \tag{6.22}$$

This is because the definition of *LCU* is also defined on a straight trajectory. However, when $\kappa \to \infty$ (for a small-radius trajectory), the equation (6.21) approaches:

$$LCU = LCU_1 = LCU_2 = \frac{1}{CU_{robot}} \tag{6.23}$$

This means that for $CU_{robot} \in (0, 1)$— CU_{robot} cannot be zero from equation (6.23) and the fact that there is no perfect control, and CU_{robot} cannot be 1; otherwise, the system is uncontrollable according to the matching conditions in robust control theory—the value of *LCU* must be more than one (and might approach infinity) to allow enough perturbation for a straight trajectory to become a small-radius trajectory. Since the value of *LCU* approaches infinity as the value of the linear velocity v approaches zero and vice versa, a smaller-radius trajectory (requiring high *LCU*) means a smaller linear velocity along the trajectory. However, the case of zero-radius turning (when $v = 0$) is impossible since the control law (6.6) yields $\omega_e = 0$, which means that the robot just stops moving.

Once the desired value of the *LCU* at a configuration is obtained. The linear velocity can be calculated by the following approximation scheme. From many experiments of determination of the *LCU*s by varying the linear velocity *V* with a constant *T*, on average, we find that the value of *LCU* at any configuration *q* increases by double when the value of the linear velocity decreases by half. This fact becomes more obvious when *T* is large (slow sampling period) because there is less effect from the nonlinearity perturbation due to smaller average values of *LCU* in the workspace. For this reason, a simple linear velocity estimation for a desired value of *LCU* can be achieved by the following equation.

$$v = \frac{LCU_V}{SF \cdot LCU} V \, , \tag{6.24}$$

where, LCU_V is the *LCU* at the same configuration when the linear velocity is *V* (or –*V*), and *SF* is a safety factor.

A more accurate and sophisticated function to estimate the value of the linear velocity yielding the desired *LCU* may be derived. However, to minimize computation time for the real-time path following controller, the linear velocity estimation using (6.24) is more preferable. An easy remedy to ensure that the equation (6.24) always yields a conservative value of *v* is the use of a safety factor, $SF \in (1, \infty)$. The safety factor can be multiplied to the obtained *LCU* to yield a higher value of *LCU*. However, this might result in a too conservative control law if the safety factor is too high.

Robot recovery process. There are some occasions when a robot deviates too far from an optimal path or moves into some regions of a workspace that possess LCU values below a specified minimum LCU level, which indicates that the robot is operating near obstacles in the workspace. To assure safe path following, one must steer the robot back on an optimal path (or at least in the neighborhood of the optimal path) before a normal path following can resume. We call the process of getting the robot back on the optimal path a “recovery process”. This recovery process is achieved by temporary interrupting the current path following operation and steering the robot along a temporary path leading toward the current optimal path. The temporary path follows the direction of the gradient of the LCUF around the optimal path; therefore, the overall value of LCUs of the robot always increases. After the LCU of the robot is above a specified maximum LCU level, the current path following operation resumes.

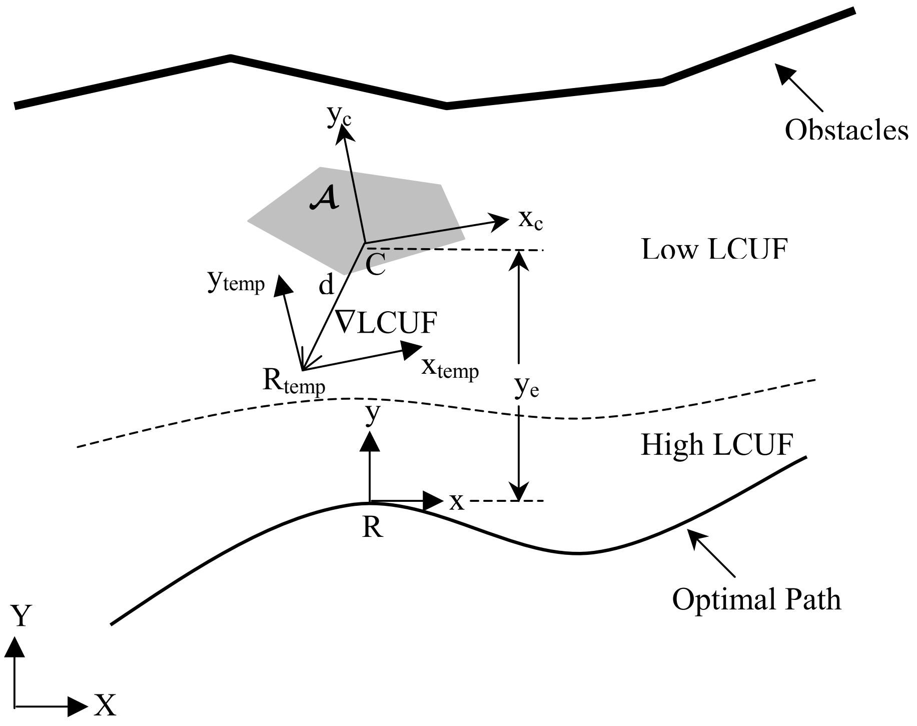


Figure 6.4 Robot recovery process.

Figure 6.4 shows a graphical representation of the robot recovery process just described. The robot recovery process is an iterative process, where the process repeats until a certain condition is met. The process starts when the LCU at a current configuration, configuration *C* in Figure 6.4, is less than a specified minimum LCU level, LCU_{min}. The key is to steer the robot toward a configuration possessing a higher LCU, configuration R_{temp}. This can be accomplished by using a two-phase method. The first phase is an angle-correction phase, where the robot is reoriented to lie up with the new orientation (but the same position) by traveling on a small-radius path or even a zero-radius path. This phase helps increase the LCU since the value of the LCU is sensitive to the orientation. The second phase is to reposition the robot to while trying to maintain the same orientation, which corresponds to moving from configuration *C* to configuration R_{temp}. To do this, the temporary coordinate frame $\{xy_{temp}\}$ is set to be a temporary moving frame toward which the control law (6.6), with controller gains k_{1recov} and k_{2recov}[1], must

[1] A different set of controller gains is used in the robot recovery process.

steer the robot using a parallel parking-like motion. The position of the configuration R_{temp} can be determined from the unit gradient of the LCUF, *∇LCUF*, in X- and Y-directions at configuration *C*. In particular, it can be computed as follows.

$$
\begin{aligned}
x_{R_{temp}} &= x_c - sign(\nabla LCUF \cdot \hat{y}) \cdot \left(\frac{\partial LCUF}{\partial X} \right)_{X=x_c} \cdot d \\
y_{R_{temp}} &= y_c - sign(\nabla LCUF \cdot \hat{y}) \cdot \left(\frac{\partial LCUF}{\partial Y} \right)_{Y=y_c} \cdot d
\end{aligned}
\tag{6.25}
$$

where, d = variable signed distance toward an optimal path.

$\hat{y}$ = y-axis of the moving frame *{xy}*.

The variable distance d is usually chosen to be a portion of the error along the y-axis, y_e, at that instant. This means one can choose $d = k_3 y_e$, where $k_3 \in (0,1]$. The sign of the dot product term $sign(\nabla LCUF \cdot \hat{y})$ is used to determine the sign of *d*. Once the robot has been moving toward the R_{temp} for a specified number of cycles or the specified maximum LCU level, LCU_{max}, is reached, the whole process repeats until the robot enters a high LCUF zone, where the normal path following process can resume. This high LCUF zone is indicated by the region where every configuration in the region possessing the LCU more than or equal to the minimum LCU level. The following algorithm portrays the robot recovery process step by step:

Algorithm 6.1 Robot recovery algorithm

Given: k_{1recov}, k_{2recov}, k_3, LCU_{min}, and LCU_{max}.

Step 1 if *LCU* at current configuration (x_c, y_c, θ) is less than a minimum LCU level, $LCU_{\min}$, go to Step2, else terminates the algorithm.

Step 2 at current position (x_c, y_c), locates the angle θ_m yielding the maximum LCU, and reorients the robot orientation from θ to θ_m. Then, go to Step 3.

Step 3 computes $\nabla LCUF$, determines y_e, and calculate (x_{Rtemp}, y_{Rtemp}) of R_{temp} using (6.25). Then, go to Step 4.

Step 4 stabilize the robot around $(x_{Rtemp}, y_{Rtemp}, \theta_m)$ using the control law (6.6) with cycling linear velocity (varying from $-V$ to $+V$ within a period of time; however, the actual linear velocity is determined by the method described in the previous subsection) until the number of a specified cycle is reached or the current LCU is more than LCU_{max}. Then, go to Step 1.

The robot recovery process always steers the robot toward the optimal path when the following condition is satisfied.

Condition 6.1 The LCUF around an optimal path must contain only one global maximum at the optimal path.

The Condition 6.1 is required to prevent the mobile robot from being trapped at local maxima before reaching the optimal path. This condition is always true if the following geometrical condition of the workspace is true.

Condition 6.2 The workspace must have a unique orthogonal projection on a path.

The Condition 6.2 can be presented graphically in Figure 6.5 below. The workspace and path given in Figure 6.5(a) yields a unique orthogonal projection of the workspace on the path. On the other hand, the workspace and path given in Figure 6.5(b) represents a non-unique orthogonal projection case (within path region AB and CD). In this case, there exist local maxima in both region of the path, and the Algorithm 6.1 is no longer guaranteed to steer the robot toward the optimal path. If both the workspace and the path satisfy Condition 6.2, the Condition 6.1 is automatically satisfied, which is the result from the gradient property of the LCUF discussed below.

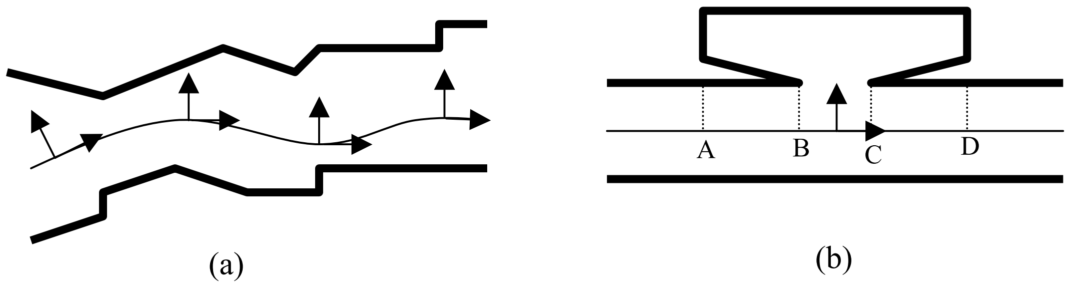


Figure 6.5 Projections of workspaces on paths (a) unique case and (b) non-unique case.

Gradient property of a LCUF[2]. An important property of a LCUF (or equivalently an estimated LCUF) is its gradient property. This property is proved to be very useful in locally steering a mobile robot toward an optimal path as previously discussed. We will use the global optimality property of a LCUF discussed in Chapter 5 to prove that if Condition 6.2 is true, with any fixed value of the orientation coordinate θ, the gradient vectors at every position (x, y) in a LCUF always points toward an optimal path. In section 5.3, we have proved that the estimated LCU for a line in a workspace is a radially unbounded function of (x, y) with any constant θ. If one take derivative of the LCU value along a radial line, the gradient vector always points outward from the origin along the radial line. Furthermore, we have also proved that, in the case for a line, the maximum estimated LCU of configuration (x, y, θ) for all $\theta \in (-\pi, \pi]$ is also a radially unbounded function. From these proofs, we have illustrated in section 5.3 that for a general workspace satisfying the Condition 6.2 there exists only one global maximum in which an optimal path lies, as shown in Figure 5.8(d). Proceeding from this Figure, one can determine a vector field representing the gradient vectors at every point in Figure 5.8(d), and the result is presented in Figure 6.6.

[2] The same property also applies to an estimated LCUF, which is an approximation of a LCUF.

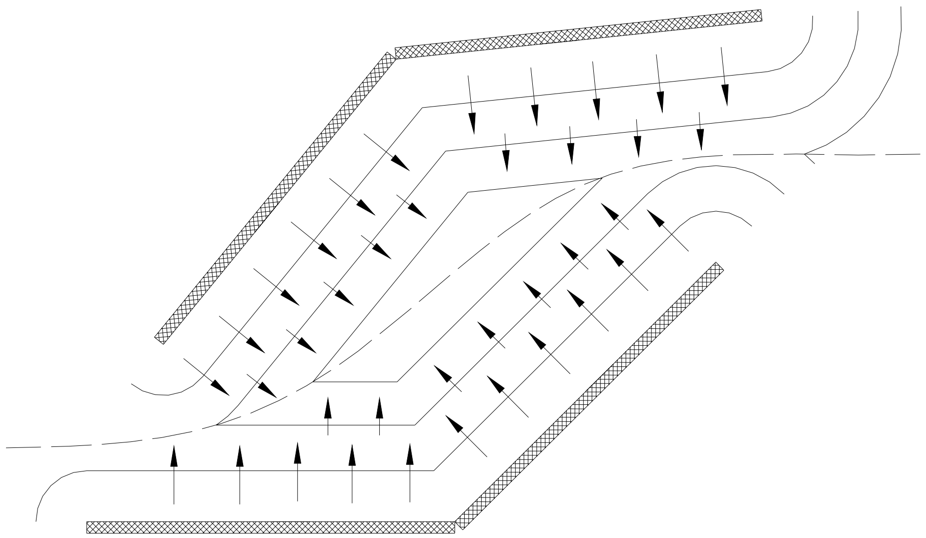


Figure 6.6 A gradient vector field of a LCUF (or an estimated LCUF).

It is clear that the gradient vectors are always perpendicular to the isocontour, and their direction cosines always point toward the optimal path. An example of a gradient vector field of the maximum estimated LCUF in Figure 5.9 is shown in Figure 6.7. Notice that every gradient vector in the gradient vector field points toward the location of an optimal path. Dots shown in the figure represent zero gradients.

In the case that a planned path for a mobile robot is not an optimal path in a given workspace, steering the robot back on the planned path by following the gradient vector field of the LCUF (or the estimated LCUF) is no longer guaranteed. In this case, a slightly different form of equation (6.25) is used instead. Since we cannot rely on the gradient direction of the LCUF, the following equation is used to calculate the position (x_{Rtemp}, y_{Rtemp}) of configuration R_{temp}.

$$\begin{aligned} x_{R_{temp}} &= x_c + \sin(\theta_R)\cdot d \\ y_{R_{temp}} &= y_c - \cos(\theta_R)\cdot d \end{aligned} \qquad (6.26)$$

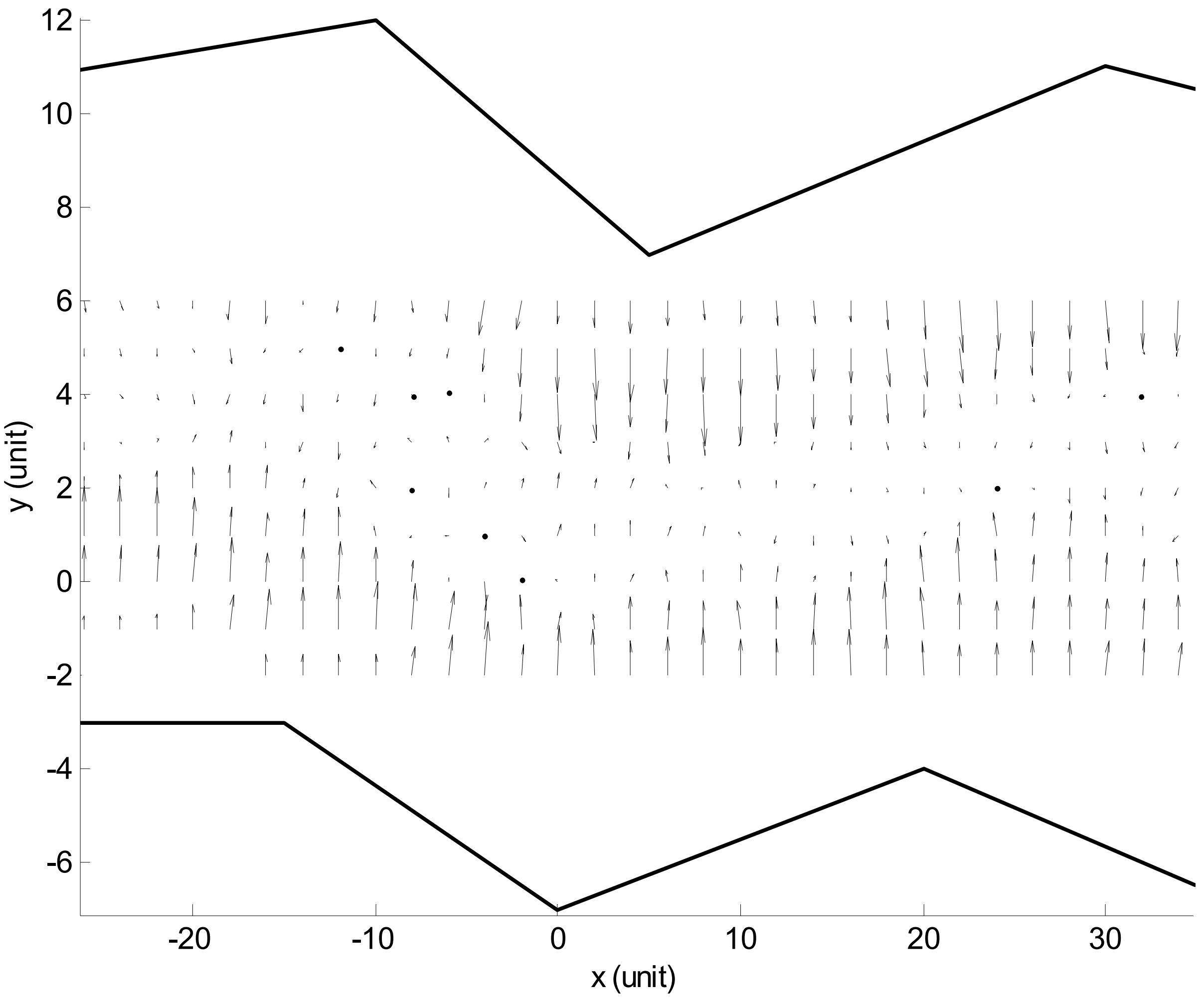


Figure 6.7 A gradient vector field for the maximum estimated LCUF in Figure 5.9.

Equation (6.26) does not utilize the information from the gradient vector field, but uses only geometrical information. The introduction of d in both equations (6.25) and (6.26) is another control for a user to adjust the rate of convergence of the path following.

LCUF-based robust path following control. We have discussed every component constituting the LCUF-based robust path following control, and we are now ready to state its algorithm. The algorithm of the LCUF-based robust path following control is as follows.

Algorithm 6.2 LCUF-based robust path following control
Given: a robot, a workspace, a path, initial condition, final time, moving direction, control limits, control acceleration limits, CU_{robot}, *SF*, *V*, *T*, k_1, k_2, ℓ, and *LCUFtemplate*.

```
Step 1  estimates LCU at the current configuration and determines the linear velocity using
        the equation (6.24) with respect to control and acceleration limits. Then, go to Step 2.

Step 2  if LCU from Step 1 is lower than the specified minimum LCU level, go to Step 3,
        else, go to Step 4.

Step 3  executes Algorithm 6.1 (robot recovery), and then go to Step 1.

Step 4  computes the angular velocity using the control law (6.6) with respect to control and
        acceleration limits. Then, go to Step 5.

Step 5  updates the current robot configuration. Then, go to Step 6.

Step 6  if final time is reached, terminates the algorithm,
        else, go to Step 1.
```

The Algorithm 6.2 has been implemented and simulated on a personal computer using MATLAB®, listed in Appendix B. The simulation results are presented in the next section.

6.3 Simulation Results and Discussions

A convex polygon mobile robot is chosen for simulations, with wheel base B = 3 units, characteristic length ℓ = 14.32 units, and its dimension and fixed coordinated frame are similar as shown in Figure 5.3. The robot linear velocity and acceleration range are within [-1, 1] unit/s and [-0.2, 0.2] unit/s^2, respectively. The robot angular velocity and acceleration range are within [-1, 1] rad/s and [-0.2, 0.2] rad/s^2, respectively. The overall control uncertainty of the robot,

CU_{robot}, is set to be 0.2 or 20 percent for common working conditions. A general workspace is selected to be the one as shown in Figure 6.7. The controller gains k_1 and k_2 are chosen to be 10 and 5. The sampling period T and the safety factor SF of the controller are 0.5 second and 1.5. The robot recovery controller gains k_{1recov}, k_{2recov}, and k_3 are 2, 1, and 0.33, respectively. The minimum LCU level, LCU_{min}, to activate the recovery process is set at 0.2, and the maximum LCU level, LCU_{max}, to terminate the recovery process is set at 1.0. All of the parameters above are listed in Table 6.1.

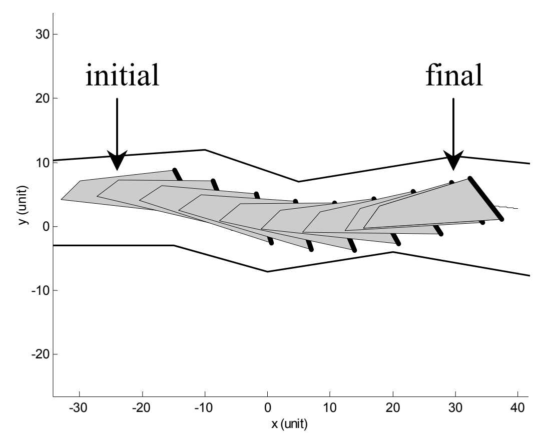
Table 6.1 The values of parameters used in the simulations.

Parameter	Value
k_1	10
k_2	5
ℓ	14.32 unit
B	3 unit
k_{1recov}	2
k_{2recov}	1
k_3	0.33
CU_{robot}	0.2
LCU_{min}	0.2
LCU_{max}	1.0
SF	1.5
v	[-1, 1] unit/s
$\dot{v}$	[-0.2, 0.2] unit/s^2
ω_e	[-1, 1] rad/s
$\dot{\omega}_e$	[-0.2, 0.2] rad/s^2
T	0.5 second

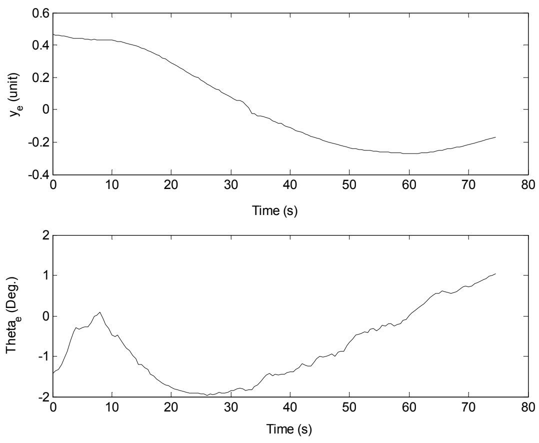
There are five simulations presented in this section. The first simulation illustrates the performance of the LCUF-based robust path following control for a given path (an optimal path in this case) when the initial configuration of the robot is close to the path. The second simulation shows the performance of the path following control in steering the robot out of a tight corner and in resuming the path following task. The third simulation presents the effectiveness of the path following control in getting the robot out of another difficult initial configuration in the workspace and in controlling the robot to follow the path backward. The fourth simulation is a revisit of the second simulation when the robot possesses higher control uncertainty (more noisy measurement and slippery floor). The fifth simulation is also a revisit of the second simulation when a new set of controller gains is employed to improve the path following performance of the existing controller. In each simulation, the control uncertainty is introduced by corrupting the control command from the path following controller with uniformly distributed random numbers generated from the interval [$-CU_{robot}$, CU_{robot}].

First Simulation. The initial configuration is set at $[-25, 5, -0.02]^T$, as shown in Figure 6.8(a). The total simulation time is 75 seconds (150 control cycles with 0.5 second sampling period). Each control cycle takes only 0.086 second of real-time (on a 1 GHz PC) to compute the feedback control. Taking into account the time for gathering the information from the robot sensors, we can easily update the feedback control for the next control cycle within one sampling period. Therefore, this path following control algorithm can be used in implementing a real-time feedback controller.

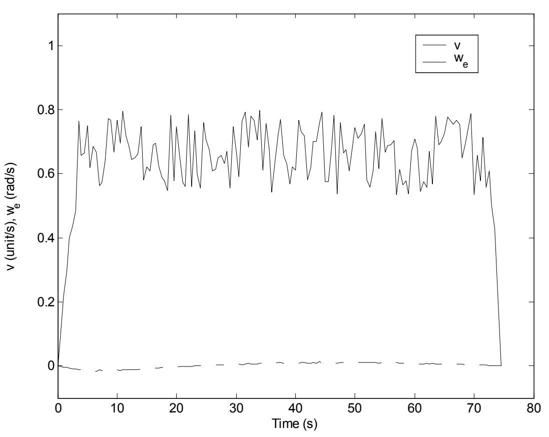
The simulation results are shown in Figure 6.8. The motion of the robot throughout the entire simulation is presented in Figure 6.8(a). The robot moves forward and follows a given path from the left to the right of the workspace. The thick line located at the front of the robot polygon indicates its heading direction, and each plot of the robot picture is 10 seconds apart. Regardless of the presence of the robot control uncertainty (CU_{robot}), the robot clearly follows a given path accurately by observing that the steady-state error in y_e and θ_e ranges are within [-0.4, 0.5] unit and [-0.035, 0.018] radian, see Figure 6.8(b). The control versus time is shown in Figure 6.8(c). The linear velocity starts from zero, fluctuates around an almost constant 0.7 unit/s due to the control uncertainty, and becomes zero again at the end of simulation time. Notice that we also take into account the dynamics of the robot system thru the velocity and acceleration limits of the robot actuators to prevent problems such as actuator saturation and overload. The robot recovery process does not occur in this simulation since the value of the estimated LCU along the robot trajectory always stays above 0.2, where the minimum LCU level is set.



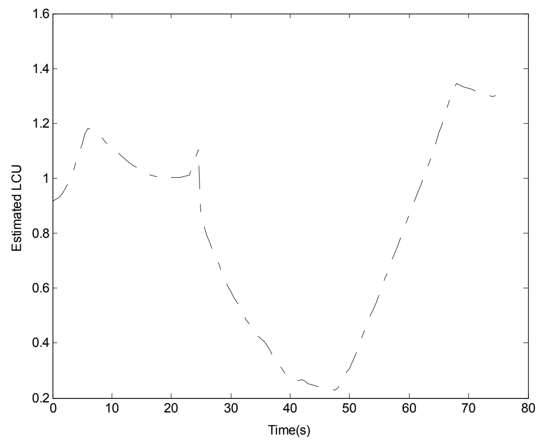
(a)



(b)



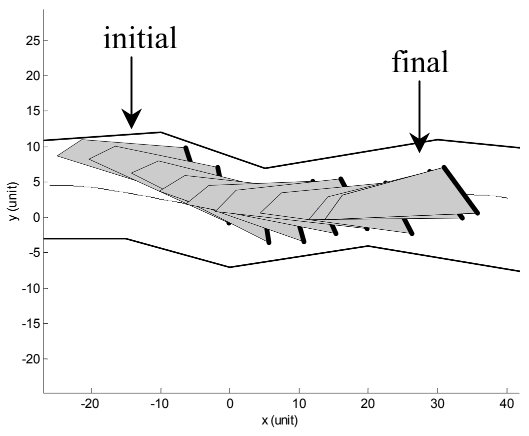
(c)



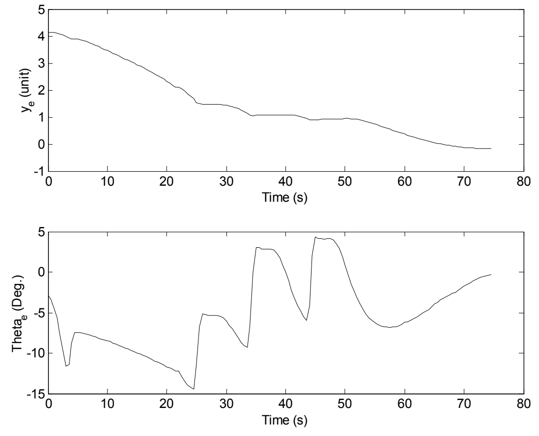
(d)

Figure 6.8 Simulation results of a general polygon robot in a general workspace when the robot initial condition is closed to a given path and moves forward, (a) robot motion, (b) system state, (c) feedback control, and (d) estimated LCU.

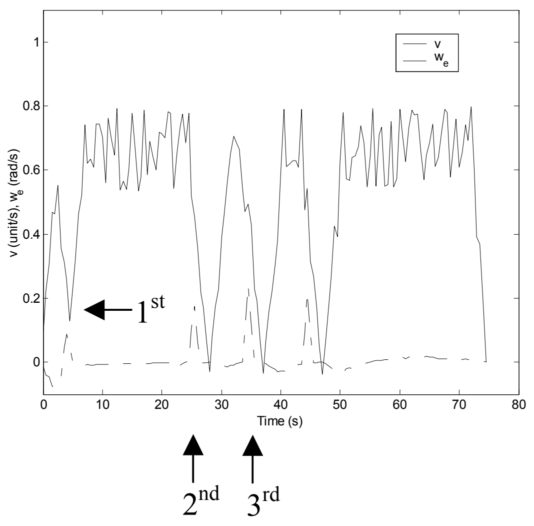
Second simulation. In this simulation, an initial configuration is set at $[-17, 8, -0.2]^T$, as shown in Figure 6.9(a). The total simulation time is still 75 seconds. The robot is commanded to move forward along the path in the workspace from left to right. A conventional path following controller would have caused a collision when it steers the robot toward the path. However, the proposed path following controller adjusts its control according to the geometry of the workspace and the robot control uncertainty, and is able to steer the robot back on the path safely, as shown in Figure 6.9(a).



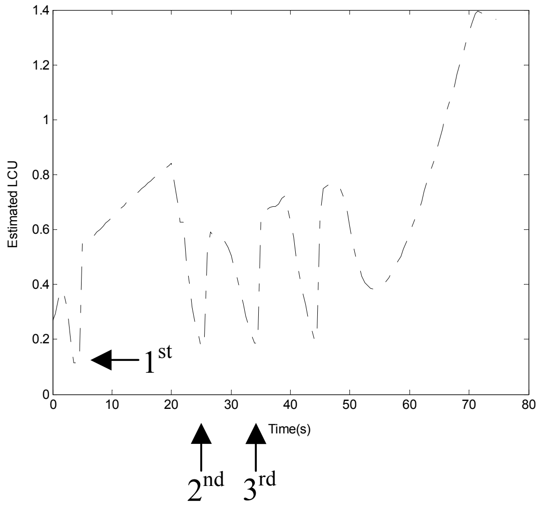
(a)



(b)



(c)



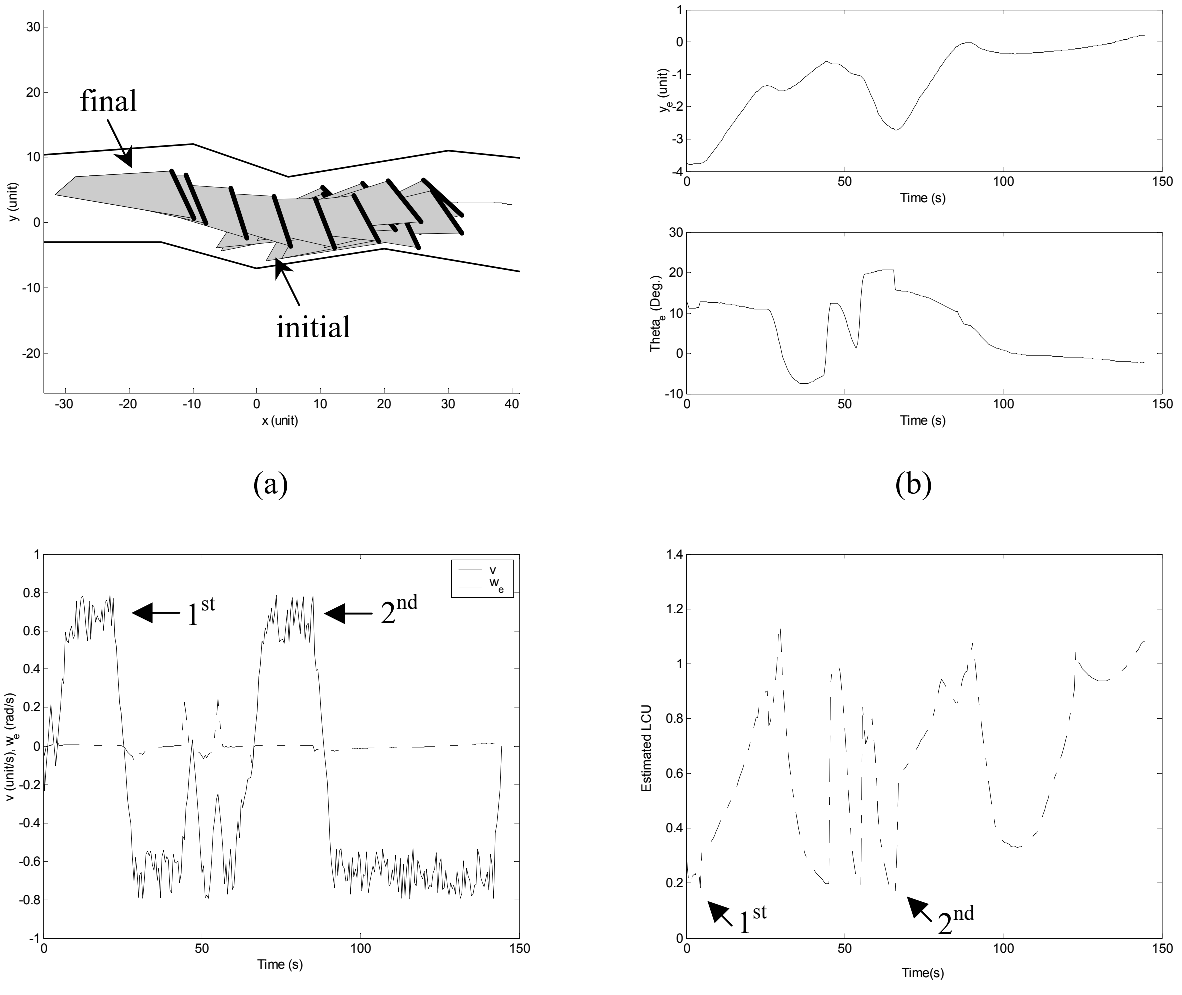
(d)

Figure 6.9 Simulation results of a general polygon robot in a general workspace when the robot initial condition is closed to obstacles and moves forward, (a) robot motion, (b) system state, (c) feedback control, and (d) estimated LCU.

The error in the system state y_e and θ_e ranges are within [-2, 2] unit and [-0.12, 0.12] radian after t = 35 seconds, see Figure 6.9(b). During the simulation, the path following control enters the recovery process three times, approximately at t = 4, 25, and 34 seconds, as indicated in Figure 6.9(d). These times are when the value of the LCU of the robot drops below the minimum LCU level (in this case *LCUmin* = 0.2). When the path following controller is in the recovery process, the controller temporary neglects the moving direction command (in this case the moving direction is forward). As a result, the linear velocity can be widely varied during the recovery process. This can be observed from Figure 6.9(c), where the main moving direction (linear velocity) is reversed temporarily. Also, notice from Figure 6.9(d) that the dip in the value of the LCU at t = 44 seconds is not low enough to activate the recovery process. However, the dip causes the robot to slow down as one can see the corresponding dip in the linear velocity in Figure 6.9(c).

Third simulation. In this simulation, an initial configuration is close to the bottom of the workspace at $[9, -3, 0.25]^T$, as shown in Figure 6.10(a). The robot is commanded to move backward along the same path in the workspace from right to left. The total simulation time is 145 seconds, which is almost twice the simulation time for the previous simulation. This happens because the robot spends almost a third of its total time in the recovery processes, as shown in Figure 6.10(c). The robot enters the recovery process twice, at t = 1 and 65 seconds in Figure 6.10(d), and takes a long time in each recovery process to get itself close to the path. As a result, the robot takes much more time to move from the initial configuration to the final configuration.

After two recovery processes, the robot is able to follow the path accurately. The error in the system state y_e and θ_e ranges are within [-1, 1] unit and [-0.17, 0.17] radian after t = 87 seconds, see Figure 6.10(b).

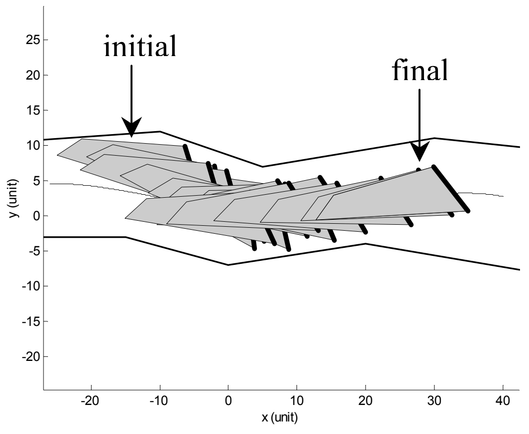


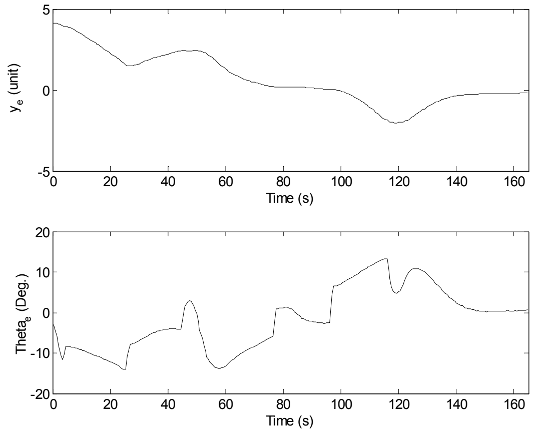
(a) (b)

(c) (d)

Figure 6.10 Simulation results of a general polygon robot in a general workspace when the robot initial condition is closed to obstacles and moves backward, (a) robot motion, (b) system state, (c) feedback control, and (d) estimated LCU.

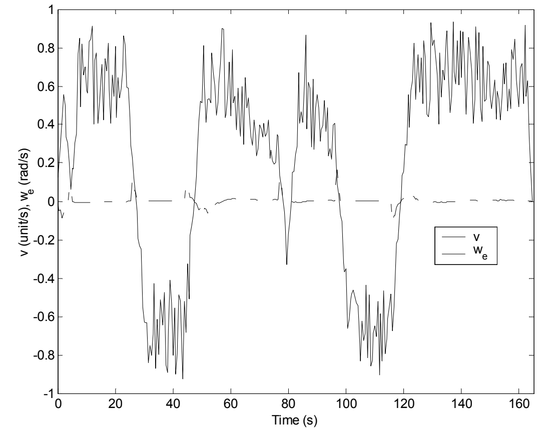
Fourth simulation. The second simulation is revisited here to determine the performance of the LCUF-based robust path following control under the presence of a high overall control uncertainty. In this simulation, we double the value of the CU_{robot} to 0.4 or 40 percent, which represents the case when the robot operates on very slippery floor. Although, the robot takes more time to move from the initial to the final configurations (165 seconds) than the second simulation, the simulation results illustrate an impressive capability of the path following controller to steer the robot along the path safely despite the high control uncertainty, see Figure 6.11(a).

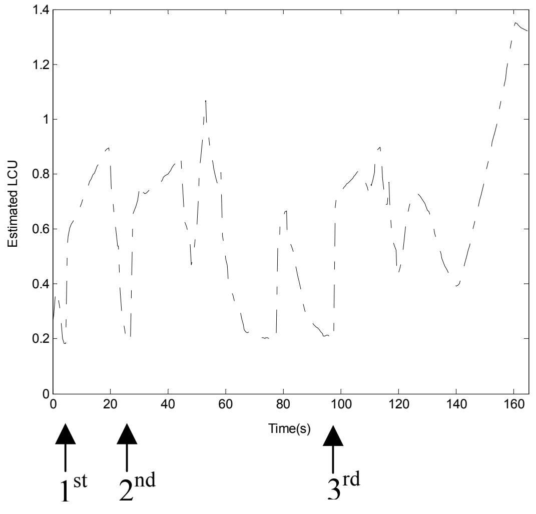




(a)

(b)





(c)

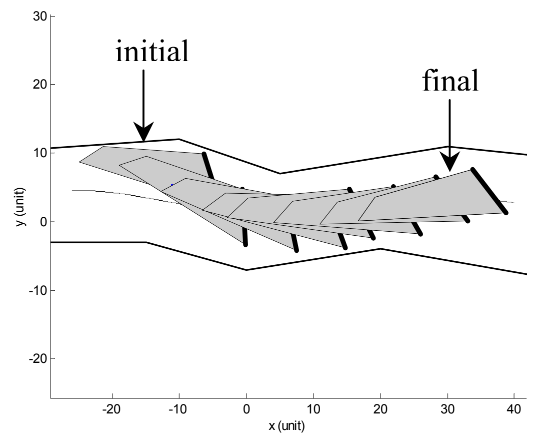
(d)

Figure 6.11 Simulation results of a general polygon robot in a general workspace when the robot initial condition is the same as that of the simulation in Figure 6.9 and moves forward with $CU_{robot} = 0.4$, (a) robot motion, (b) system state, (c) feedback control, and (d) estimated LCU.

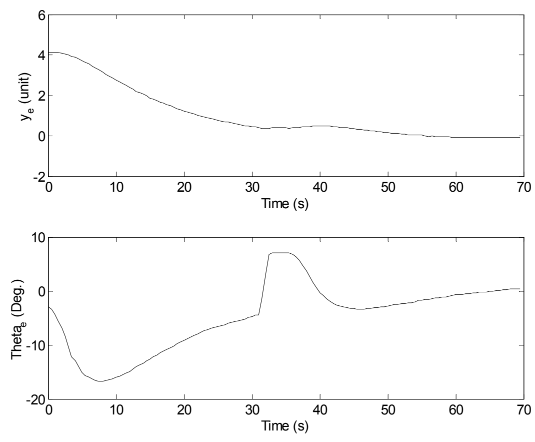
The robot enters the recovery process three times, see Figure 6.11(d), as same as the second simulation does. However, each recovery process takes much longer time due to the higher control uncertainty. The error in the system state y_e and θ_e ranges are within [-2.5, 2.5] unit and [-0.26, 0.26] radian after t = 20 seconds, as shown Figure 6.11(b). These errors are higher than the ones obtained in the second simulation as we expected.

Fifth simulation. The second simulation is revisited again. This time, our objective is to determine the improvement of the performance of the LCUF-based robust path following control with a new set of controller gains. Most of the parameters used in this simulation are the same as shown in Table 6.1 except that some of the controller gains are changed. In particular, the gain k_2 and k_{2recov} are increased to 7.5 and 1.5, respectively. The total simulation time is 70 seconds. The robot moves from left to right as shown in Figure 6.12(a). The robot enters the recovery process only once (shown in Figure 6.12(c) and (d)) as oppose to three times in the second simulation. Moreover, the path following performance of the robot is far better than that in the second simulation where the error in the system state y_e and θ_e ranges are within [-2, 2] unit and [-0.12, 0.12] radian only after t = 27 seconds, see Figure 6.12(b).

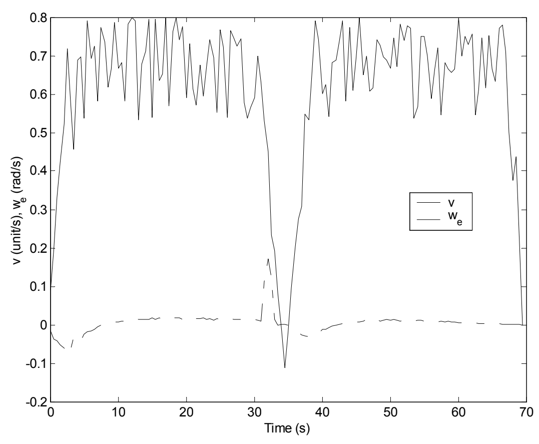
The larger values of both gains, k_2 and k_{2recov}, that control how quickly the robot corrects its error in orientation cause the shape of the system trajectory to approach the given path smoothly. Therefore, the robot does not enter the recovery process as much as it does in the second simulation, and, as a result, the robot takes much less time to converge to the given path. This shows the possibility for optimizing the controller gains to best perform the path following task for given paths and environments. However, optimal path-dependent or environment-dependent gains can be very complex and computationally expensive compared to a set of fixed gain values.



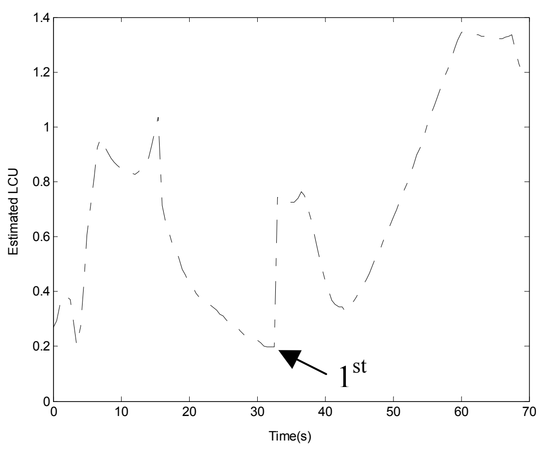
(a)



(b)



(c)



(d)

Figure 6.12 Simulation results of a general polygon robot in a general workspace with a new set of the path following controller gains, (a) robot motion, (b) system state, (c) feedback control, and (d) estimated LCU.

6.4 Conclusion

The first real-time robust path following control algorithm for uncertain nonholonomic mobile robots operating amidst a cluttered workspace is proposed. The control algorithm consists of two separate controllers: a main controller and a robot recovery controller. The main controller controls the robot in the vicinity of a given path while the robot recovery controller controls the robot in the vicinity of obstacles by making use of the gradient of the LCUF. Also built-in the control algorithm is the method, based on the information from the LCUF, for determining a maximum safe speed (linear velocity) that the mobile robots with sensory and control uncertainty must obey to ensure safe navigation in the tight workspace. Since the control algorithm relies on the information from the LCUF, it is called the "LCUF-based robust path following control algorithm". The simulations illustrate impressive performance of the control algorithm by navigating, in real-time, a general polygon robot in a tight workspace accurately and safely despite the presence of high uncertainty.

Chapter 7
Conclusions

All the material presented in this dissertation is summarized below. Main contributions from this work are also given. Some suggestions for future work are identified and provided at the end of this chapter.

7.1 Summary and Contributions

We have proposed two motion planning strategies, based on the concept of the maximum allowable uncertainty, for autonomous nonholonomic mobile robots operating in cluttered environment and subject to uncertainties. The motion planning strategies guarantee to return a global optimal path if one exists, and the likelihood of collision is always eliminated when the robots traversing the path accurately. A method for determining the maximum allowable degree of uncertainty (from both internal and external) of the robot, to assure safe navigation in a workspace, was developed. In addition, a means for calculating the maximum safe velocity for path following operation of the robots was also formed. We also constructed the first robust path following control algorithm that guaranteed no collision during the path following execution of the robots along a given path in tight workspace.

Our main goal is to develop efficient techniques for motion planning and control of nonholonomic mobile robots. To this end, we have made contributions toward that goal as follows.

- A general concept of lumping both control and sensory uncertainties together through feedback control to form a single measurement of uncertainty level of a given robot system, in chapter 3.
- A means to measure an acceptable uncertainty level at different configuration, called "Linear Control Uncertainty (LCU)", in a given workspace for a particular robot, in chapter 3.
- In chapter 3, a new motion planning technique based on the concept of a field of LCU, called "LCUF-based motion planning", which guarantees to return a global optimal path if one exists.
- A means to measure an acceptable uncertainty level at different configuration and control, called "Circular Control Uncertainty (CCU)", in chapter 4.
- A new motion planning method based on the concept of a field of CCU, called "CCUF-based motion planning", which guarantees to return a global optimal path if one exists, in chapter 4.
- A criteria and method for determining the allowable maximum level of uncertainty for the mobile robots to follow a given path safely, in chapter 4.
- A method to estimate an acceptable uncertainty level for each configuration in a given workspace for a particular robot, in chapter 5. The method estimates the value of LCU accurately and quickly. When used with the LCUF-based motion planning, the estimation method helps decreasing the motion planning run-time by more than an order of magnitude.
- A method for computing the safe maximum velocity for the mobile robots to follow a given path safely, in chapter 6.
- Also in chapter 6, the first real-time robust path following control algorithm for the mobile robots to safely follow a given path in a tight workspace.

7.2 Recommendations for Future Work

The following topics are recommended for future work.

- Since both proposed motion planning techniques cannot plan optimal paths in real-time, a real-time motion planning algorithm based on a database of pre-computed optimal paths obtained from running the motion planning techniques off-line for variety of environmental cases is a viable alternative, and deserves more study.
- It would be interesting to study how to estimate the LCU of a concave polygon robot, so that both motion planning and control schemes can take full advantages of it.
- An adaptive or optimal control method based on the workspace information in the form of LCU should be investigated. It is possible to use the LCU to adjust the gains of a path following controller in an optimal way to yield an optimal path following performance.

Bibliography

Aguilar, L.E., Hamel, T., and Souéres, P. (1997), "Robust Path Following Control for Wheeled Robots via Sliding Mode Techniques," *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems*, pp. 1389-1395.

Aguilar, L.E., Souères, P., Courdesses, M., and Fleury, S. (1998), "Robust Path-Following Control with Exponential Stability for Mobile Robots," *Proc. of the 1998 IEEE Int. Conf. on Robotics and Automation*, Belgium, pp. 3279-3284.

Aicardi, M., Casalino, G., Balestrino, A., and Bicchi, A. (1994), "Closed Loop Smooth Steering of Unicycle-Like Vehicles," *Proc. of the 33rd Conf. on Decision and Control*, Florida, pp. 2455-2458.

Alexander, J.C., Maddocks, J.H., and Michalowski, B.A. (1998), "Shortest distance paths for wheeled mobile robots," *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 5, pp. 657-662.

Barraquand, J. and Latombe, J-C. (1991), "Nonholonomic Multibody Mobile Robots: Controllability and Motion Planning in the Presence of Obstacles," *Proc. of the 1991 IEEE Int. Conf. on Robotics and Automation*, Sacramento, pp. 2328-2335.

Brauer, G.L., Cornick, D.E., and Stevenson, R. (1977), "Capabilities and Applications of the Program to Optimize Simulated Trajectories," NASA CR-2770.

Bryson, A. E. (1999), *Dynamic Optimization*, Addison Wesley Longman, Inc., California.

Cameron, J.M. and Book, W.J. (1994), "Optimal Path Planning for the Motion of a wheel," *Proc. of the 1994 IEEE Int. Conf. on Robotics and Automation*, pp. 1574-1580.

Chung, Y., Park, C., and Harashima, F. (2001), "A Position Control Differential Drive Wheeled Mobile Robot," *IEEE Transactions on Industrial Electronics*, Vol. 48, No. 4, pp. 853-863.

De Luca, A., Oriolo, G., and Samson, C. (1998), "Feedback Control of a Nonholonomic Car-Like Robot," *Robot Motion Planning and Control*, Springer-Verlag, London, pp. 171-253.

Divelbiss, A.W. and Wen, J.T. (1997), "A Path Space Approach to Nonholonomic Motion Planning in the Presence of Obstacles*," IEEE Transactions on Robotics and Automation*, Vol. 13, No. 3, pp. 443-451.

Díaz del Río, F., Jiménez, G., Sevillano, J.L., Amaya, C., and Civit Balcells, A. (2002), "Error Adaptive Tracking for Mobile Robots," *Proc. of the 2002 Industrial Electronics Conference (IECON)*, Vol. 3, pp. 2415-2420.

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*, No. 79, pp. 497-516.

Dudek, G. and Jenkin, M. (2000), *Computational Principles of Mobile Robotics*, Cambridge University Press, Cambridge.

Eberly, D. (2001), "Intersection of Convex Objects: The Method of Separating Axes", Magic Software, Inc., http://www.magic-software.com.

Fernandes, C., Gurvits, L., and Li, Z.X. (1991), "A Variational Approach to Optimal Nonholonomic Motion Planning," *Proc. of the 1991 IEEE Int. Conf. on Robotics and Automation*, pp. 680-685.

Fierro, R. and Lewis, F.L. (1998), "Control of a Nonholonomic Mobile Robot Using Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 9, No. 4, pp. 589-600.

Fraichard, Th. and Mermond, R. (1998), "Path Planning with Uncertainty for Car-Like Robots," *Proc. of the 1998 IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, pp. 27-32.

Franklin, G.F., Powell, J.D., and Workman, M. (1997), *Digital Control of Dynamic Systems*, 3rd Edition, Addison-Wesley Longman, Inc., California.

Gill, P.E., Murray, W., and Wright, M.H. (1981), *Practical Optimization*, Academic Press, Inc., New York.

Gottschalk, S. (2000), Collision Query using Oriented Bounding Boxes, Ph.D. thesis, University of North Carolina at Chapel Hill, USA.

Gurvits, L. and Li, Z.X. (1993), "Smooth Time-Periodic Feedback Solutions for Nonholonomic Motion Planning," *Nonholonomic Motion Planning*, Kluwer Academic Publishers, pp. 53-108.

Hait, A., Simeon, T., and Taix, M. (1999), "Robust Motion Planning for Rough Terrain Navigation," *Proc. of the 1999 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 11-16.

Hamel, T., Souéres, P., and Meizel, D. (2001), "Path Following with a Security Margin for Mobile Robots," *Intl. Journal of Systems Science*, Vol. 32, No. 8, pp. 989-1002.

Homaifar, A., Battle, D., and Tunstel, E. (1999), "Soft Computing-Based Design and Control for Mobile Robot Path Tracking," *Proc. of the 1999 IEEE Int. Symposium on Computational Intelligence in Robotics and Automation*, pp. 35-40.

Jacobs, P. and Canny, J. (1993), "Planning smooth paths for mobile robots," *Nonholonomic Motion Planning*, Kluwer Academic Publishers, pp. 271-342.

Khalil, H. K. (1996), *Nonlinear Systems*, 2nd Edition, Prentice-Hall, Inc., New Jersey.

Kolmanovsky, I. and McClamroch, N.H. (1995), "Developments in Nonholonomic Control Problems," *IEEE Control Systems Magazine*, Vol.15, Issue 6 , pp. 20–36.

Kondak, K. and Hommel, G. (2001), "Computation of Time Optimal Movements for Autonomous Parking of Non-Holonomic Mobile Platforms," *Proc. of the 2001 IEEE Int. Conf. on Robotics and Automation*, pp. 2698-2703.

Lafferriere, G. and Sussmann, H.J. (1993), "A differential geometric approach to motion planning," in *Nonholonomic Motion Planning*, Kluwer Academic Publishers, pp. 235-270.

Lagarias, J.C., Reeds, J.A., Wright, M.H., and Wright, P.E. (1998), "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions," *SIAM Journal of Optimization*, Vol. 9, No. 1, pp. 112-147.

Lambert, A., Hamel, T., and Le Fort-Piat, N. (1998), "A Safe and Robust Path Following Planner for Wheeled Robots," *Proc. of the 1998 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Canada, pp. 600-605.

Lambert, A. and Le Fort-Piat, N. (1999), "Safe Actions and Observations Planning for Mobile Robots," *Proc. of the 1999 IEEE Intl. Conf. on Robotics and Automation*, Michigan, pp. 1341-1346.

Lambert, A. and Le Fort-Piat, N. (2000), "Safe Task Planning Integrating Uncertainties and Local Maps Federations," *The Intl. Journal of Robotics Research*, Vol. 19, No. 6, pp. 597-611.

LaSalle, J. P. and Lefschetz, S. (1961), *Stability by Liapunov's Direct Method*, Academic Press, New York.

Latombe, J-C. (1991), *Robot Motion Planning*, Kluwer Academic Publishers, Massachusetts.

Laumond, J-P., Jacobs, P.E., Taix, M., and Murray, R.M. (1994), "A Motion Planner for Nonholonomic Mobile Robots," *IEEE Transactions on Robotics and Automation*, Vol. 10, No.5, pp. 577-593.

Laumond, J-P. (1995), "Nonholonomic Motion Planning via Optimal Control," *Algorithmic Foundations of Robotics*, A K Peters, Ltd., pp. 227-238.

Lazanas, A. and Latombe, J-C. (1995), "Motion planning with uncertainty: a landmark approach," *Artificial Intelligence*, No. 76, pp. 287-317.
Mirtich, B. and Canny, J. (1992), "Using Skeletons for Nonholonomic Path Planning among Obstacles," *Proc. of the 1992 IEEE Int. Conf. on Robotics and Automation*, France, pp. 2533-2540.

Moon, I., Miura, J., and Shirai, Y. (1999), "On-Line Viewpoint and Motion Planning for Efficient Visual Navigation under Uncertainty," *Robotics and Autonomous Systems*, Vol. 28, pp. 237-248.

Murray, R.M. and Sastry, S.S. (1993), "Nonholonomic motion planning: steering using sinusoids," *IEEE Transaction on Automatic Control*, Vol. 38, No. 5, pp. 700-716.

Murray, R.M., Li, Z., and Sastry, S.S. (1993), *A Mathematical Introduction to Robotic Manipulation*, CRC Press, Ann Arbor.

Nelder, J.A. and Mead, R. (1965), "A Simplex Method for Function Minimization," *Computer Journal*, Vol. 7, pp. 308-313.

Nilsson, N.J. (1980), *Principles of Artificial Intelligence*, Tioga Publishing Co., California.

Normey-Rico, J.E., Alcala, I., Gomez-Ortega, J., and Camacho, E.F. (2001), "Mobile Robot Path Tracking Using a Robust PID Controller," *Control Engineering Practice*, Vol. 9, pp. 1209-1214.

Oelen, W. and van Amerongen, J. (1994), "Robust Tracking Control of Two-Degrees-of-Freedom Mobile Robots," *Control Engineering Practice*, Vol. 2, No. 2, pp. 333-340.

Ollero, A., Garcia-Cerezo, A., and Martinez, J.L. (1994), "Fuzzy Supervisory Path Tracking of Mobile Robots," *Control Engineering Practice*, Vol. 2, No.2, pp. 313-319.

Ollero, A. and Heredia G. (1995), "Stability Analysis of Mobile Robot Path Tracking," *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems*, Vol. 3, pp. 461-466.

Oriolo, G., Stefano, P., and Giovanni, U. (2000), "Learning optimal trajectories for non-holonomic systems," *Int. Journal of Control*, Vol. 73, No. 10, pp. 980-991.

Page, L.A. and Sanderson, A.C. (1995), "A Path-Space Search Algorithm for Motion Planning with Uncertainties," *Proc. of the IEEE Int. Symposium on Assembly and Task Planning*, pp. 334-340.

Papadopoulos, E. and Poulakakis, I. (2001), "Planning and Obstacle Avoidance for Mobile Robots," *Proc. of the 2001 IEEE Int. Conf. on Robotics and Automation*, Korea, pp. 3967-3972.

Pears, N.E. (2001), "Mobile Robot Tracking of Pre-Planned Paths," *Advanced Robotics*, Vol. 15, No. 1, pp. 97-107.

Podsedkowski, L. (1998), "Path Planner for Nonholonomic Mobile Robot with Fast Replanning Procedure," Proc. of the 1998 IEEE Int. Conf. on Robotics and Automation, Belgium, pp. 3588-3593.

Podsedkowski, L., Nowakowski, J., Idzikowski, M., and Vizvary, I. (2001), "A New Solution for Path Planning in Partially Known or Unknown Environment for Nonholonomic Mobile Robots," *Robotics and Autonomous Systems*, Vol. 34, Issues 2-3, pp. 145-152.

Pruski, A. and Rohmer, S. (1997), "Robust Path Planning for Nonholonomic Robots," *Journal of Intelligent and Robotic Systems*, No. 18, pp. 329-350.

Qu, Z. (1998), *Robust Control of Nonlinear Uncertain Systems*, John Willey & Sons, Inc., New York.

Reeds, J.A. and Shepp, L.A. (1990), "Optimal Paths for a Car that Goes Both Forwards and Backwards," *Pacific Journal of Mathematics*, 145(2), pp. 367-393.

Renaud, M. and Fourquet, J-Y. (1997), "Minimum time motion of a mobile robot with two independent, acceleration-driven wheels," *Proc. of the 1997 IEEE Int. Conf. on Robotics and Automation*, pp. 2608-2613.

Reyhanoglu, M., McClamroch, N.H., and Bloch, A.M. (1993), "Motion planning for nonholonomic dynamic systems", in *Nonholonomic Motion Planning*, Kluwer Academic Publishers, pp. 201-234.

Samson, C. (1992), "Path Following and Time-varying Feedback Stabilization of a Wheeled Mobile Robot," Proc. of ICARCV'92, pp. RO-13.1.1-13.1.5.

Sarkar, N., Yun, X., and Kumar, V. (1993), "Dynamic Path Following: A New Control Algorithm for Mobile Robots," *Proc. of the 32nd Conf. on Decision and Control*, pp. 2670-2675.

Sekhavat, S. and Laumond, J-P. (1998), "Topological property for collision-free nonholonomic motion planning: the case of sinusoidal inputs for chained form systems," *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 5, pp. 671-680.

Sørdalen, O.J. and Canudas de Wit, C. (1993), "Exponential Control Law for a Mobile Robot: Extension to Path Following," *IEEE Transactions on Robotics and Automation*, Vol. 9, No.6, pp. 837-842.

Souères, P., Hamel, T., and Cadenat, V. (1998), "A Path Following Controller for Wheeled Robots which Allows to Avoid Obstacles during Transition Phase," *Proc. of the 1998 IEEE Int. Conf. on Robotics and Automation*, pp. 1269-1274.

Souéres, P., Balluchi, A., and Bicchi, A. (2000), "Optimal Feedback Control for Route Tracking with a Bounded-Curvature Vehicle," Proc. of the 2000 IEEE Int. Conf. on Robotics and Automation, San Francisco, pp. 2473-2478.

Sussmann, H.J. (1992), "New Differential Geometric Methods in Nonholonomic Path Finding," *Systems, Models, and Feedback: Theory and Applications*, A. Isidori and T. J. Tarn Eds., Birkhäuser, Boston, pp. 365-384.

Takeda, H., Facchinetti, C., and Latombe, J-C. (1994), "Planning the motions of a mobile robot in a sensory uncertainty field," *IEEE Transactions on pattern analysis and machine intelligence*, Vol. 16., No. 10, pp. 1002-1017.

Tayebi, A. and Rachid, A. (1996), "Path Following Control Law for an Industrial Mobile Robot," *Proc. of the 1996 IEEE Int. Conf. on Control Applications*, Michigan, pp. 703-707.

Timcenko, A. and Allen, P. (1994), "Probability – Driven Motion Planning for Mobile Robots," *Proc. of the 1994 IEEE Int. Conf. on Robotics and Automation*, San Diego, pp. 2784-2789.

Wang, Y. (1996), "Nonholonomic Motion Planning: a Polynomial Fitting Approach," *Proc. of the 1996 IEEE Int. Conf. on Robotics and Automation*, Minnesota, pp. 2956-2961.

Xu, W.L., Ma, B.L., and Tso, S.K. (1999), "Curve Fitting Approach to Motion Planning of Nonholonomic Chained Systems," *Proc. of the 1999 IEEE Int. Conf. on Robotics and Automation*, Michigan, pp. 811-816.

Yang, X., He, K., Guo, M., and Zhang, B. (1998), "An Intelligent Predictive Control Approach to Path Tracking Problem of Autonomous Mobile Robot," *Proc. of the IEEE Int. Conf. on Systems, Man, and Cybernetics*, pp. 3301-3306.

Appendix A

Separating Axis Theorem is the dual of a well-known theorem, separating plane theorem, which states that two polytopes are disjoint if and only if there exist a separating plane which is parallel to a face of one of the polytopes or is parallel to an edge taken from each. If a plane is a separating plane, then any axis perpendicular to it is a separating axis [Gottschalk, 2000].

In a two-dimensional case, a test for disjoint of a pair of convex polygons can be achieved by determining whether there exists a line perpendicular to the edges of the two polygons such that there is no intersection between the intervals of projection of the polygons onto the line. The left picture in Figure A.1 shows two disjoint convex polygons that are separated along the direction of a normal vector, $\hat{n}$, to an edge of one polygon. The right picture shows two polygons that intersect [Eberly, 2001].

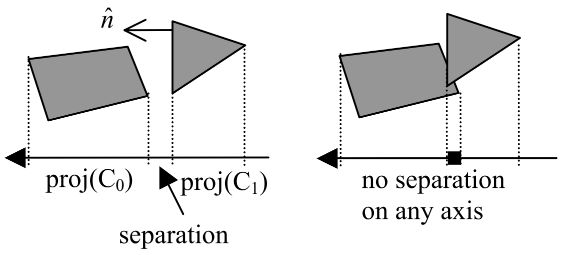


Figure A.1 Nonintersecting convex polygons (left). Intersecting convex polygons (right).

In our application, one can apply the separating axis theorem to detect a collision between a mobile robot and its environment by representing both the robot and the environment with convex polygons. Then, every pair of robot polygons and object polygons in the environment is checked, one by one, for intersection. If there does not exist any intersection, no collision occurs.

For example, to perform a collision detection of a problem depicted in Figure 3.7(d) in Chapter 3, the robot is represented by a convex polygon $\mathcal{A}$ and the environment is represented by three convex polygons, as shown in Figure A.2. In this case, three pairs of polygons: {$\mathcal{A}$, $\mathbf{C_0}$}, {$\mathcal{A}$, $\mathbf{C_1}$}, {$\mathcal{A}$, $\mathbf{C_2}$} must be checked for intersection.

This collision detection scheme is selected for our application due to its simplicity and efficiency. As studied in [Gottschalk, 2000], the separating axis approach outperforms other previous approaches in term of computation time, and was also implemented in a well-known interference detection software package called "RAPID (Rapid and Accurate Polygon Interference Detection)".

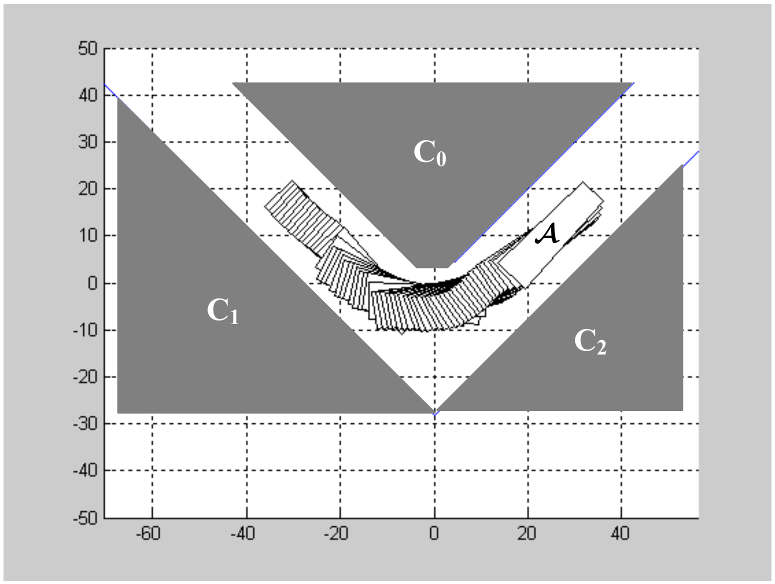


Figure A.2 A convex polygon representation of a problem in Figure 3.7(d).

Appendix B

A list of names, including short descriptions, of computer programs used in Chapters 3, 4, 5, and 6 of this dissertation is given below. All of the programs are implemented in MATLAB® programming language. Their source codes can be found in the attached CD-ROM.

Chapter 3

Section 3.5

- best_path.m : This program determines an optimal control based on the approximation-and-optimization approach using LCUF.
- obj_cuf.m : This program evaluates the value of the objective function for each control candidate.
- init_guess.m : This program helps a user in determining an initial guess for a control candidate.
- animation.m : This program displays the results from the obtained optimal control.

Chapter 4

Section 4.3

- A_star_search.m : This program constructs and searches through a directed graph to determine the optimal path using CCUF.
- A_star_analysis.m : This program displays the results from the obtained optimal path.

Chapter 5

Section 5.2

- LCUFTempLineConvex.m : This program determines a line sub template for a general convex polygon.

- LCUFTempLtEndConvex.m : This program determines a left end sub template for a general convex polygon.

- LCUFTempRtEndConvex.m : This program determines a right end sub template for a general convex polygon.

Section 5.3

- best_path_estCUF.m : This program determines an optimal control based on the approximation-and-optimization approach using estimated LCUF.
- obj_estCUF.m : This program evaluates the value of the objective function for each control candidate based on estimated LCU.
- init_guess_estCUF.m : This program helps a user in determining an initial guess for a control candidate used in best_path_estCUF.m.
- animation_estCUF.m : This program displays the results from the obtained optimal control.

Chapter 6

Section 6.1

- StraightPathTracking.m : This program simulates a mobile robot following a straight path using the proposed robust path following control law.
- PolyPathTracking.m : This program simulates a mobile robot following a polynomial path using the proposed robust path following control law.

Section 6.3

- LCUFBasedRobustControl.m : This program simulates a mobile robot following a polynomial path in a tight workspace using the LCUF-based robust path following control.

Vita

Amnart Kanarat was born in Nakhonratchasima, Thailand on August 26, 1975. He graduated from King Mongkut's Institute of Technology Ladkrabang (KMITL) with a Bachelor of Engineering Degree in Mechanical Engineering in May 1996. After working as a Teaching Assistant in a Mechanical Engineering Laboratory at KMITL for a year, he won a scholarship that is awarded by the Royal Thai Government to further his studies in the United States, leading to M.S. and Ph.D. degrees in the field of Robotics and Automation. He first attended Virginia Tech in Fall 1997 to pursue a Master of Science in Mechanical Engineering, and received the degree in Fall 1999. He also continued his study in the Ph.D. program in Mechanical Engineering at Virginia Tech. After graduation, he will assume a faculty position in the department of Mechanical Engineering at KMITL, Bangkok, Thailand.