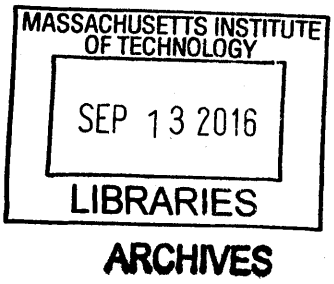
Robust Motion Planning for Autonomous Tracked Vehicles in Deformable Terrain

by

Sang Uk Lee

B.S. Mechanical Engineering
Seoul National University, 2014



Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of
Master of Science in Mechanical Engineering
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2016

Signature redacted

Signature of Author ....
Department of Mechanical Engineering
August 8, 2016

Signature redacted

Certified by ....
Karl Iagnemma
Principal Research Scientist
Thesis Supervisor

Signature redacted

Accepted by ....
Rohan Abeyaratne
Professor of Mechanical Engineering
Chairman, Committee on Graduate Studies

Robust Motion Planning for Autonomous Tracked Vehicles in Deformable Terrain

by

Sang Uk Lee

Submitted to the Department of Mechanical Engineering
on August 8, 2016 in partial fulfillment of the
Requirements for the degree of Master of Science in
Mechanical Engineering

Abstract

Ensuring the safety of autonomous vehicles during operation is a challenging task. Numerous factors such as process noise, sensor noise, incorrect model etc. can yield uncertainty in robot's state. Especially for tracked vehicles operating on rough terrain, vehicle slip due to vehicle terrain interaction affects the vehicle system significantly. In such cases, the motion planning of the autonomous vehicle must be performed robustly, considering the uncertain factors in advance of the real-time navigation.

The primary contribution of this thesis is to present a robust optimal global planner for autonomous tracked vehicles operating in off-road terrain with uncertain slip. In order to achieve this goal, three tasks must be completed. First, the motion planner must be able to work efficiently under the non-holonomic vehicle system model. An approximate method is applied to the tracked vehicle system ensuring both optimality and efficiency. Second, the motion planner should ensure robustness. For this, a robust incremental sampling based motion planning algorithm (CC-RRT*) is combined with the LQG-MP algorithm. CC-RRT* yields the optimal and probabilistically feasible trajectory by using a chance constrained approach under the RRT* framework. LQG-MP provides the capability of considering the role of compensator in the motion planning phase and bounds the degree of uncertainty to appropriate size. Third, the effect of slip on the vehicle system must be modeled properly. This can be done in advance of operation if we have experimental data and full information about the environment. However, in case where such knowledge is not available, the online slip estimation can be performed using system identification method such as the IPEM algorithm.

Simulation results shows that the resulting algorithms are efficient, optimal, and robust. The simulation was performed on a realistic scenario with several important factors that can increase

the uncertainty of the vehicle. Experimental results are also provided to support the validity of the proposed algorithm. The proposed framework can be applied to other robotic systems where robustness is an important issue.

Thesis Supervisor: Karl Iagnemma

Title: Principal Research Scientist

Acknowledgments

Though this thesis has only my name in the front, I would never have been able to do this work on my own. On this page, I'd like to thank everyone who helped me through the first two years of my graduate life.

I would like to thank my advisor Dr. Karl Iagnemma for his guidance and support during the research and writing of this thesis. I could grow passion for robotics thanks to great opportunities he has given me.

I would also like to thank my colleagues Ramon Gonzalez and Junghee Park in Robotic Mobility Group. Ramon has kindly offered me a great help during research and writing of this thesis. I could learn a lot from Junghee's guidance as a senior graduate student.

I would like to thank Samsung Scholarship Foundation for supporting me financially through the Master's degree program. I would like to thank United States Army Research Office (contract W911NF-13-1-0063) for the financial support as well.

Finally, I would like to thank my parents for everything. They made it all possible for me to come to MIT and pursue my dream.

Contents

1. Introduction 17

1.1.Related Works

1.1.1. Motion Planning 18

1.1.2. Tracked Vehicle Navigation in Off-road Condition 24

1.2.Summary of Contribution and Outline 27

2. Non-holonomic Motion Planning for Autonomous Tracked Vehicles 29

2.1.Problem Statement 30

2.2.System model 31

2.3.RRT* with Non-holonomic Steering

2.3.1. RRT* Algorithm 33

2.3.2. Non-holonomic Steering

2.3.2.1. Exact Connection Method 35

2.3.2.2.Approximate Method 37

2.4.Simulation Result

2.4.1. Dubins model 40

2.4.2. Approximate Method 41

3. Robust Global Motion Planner for Autonomous Tracked Vehicles with Chance Constrained Approach 45

3.1.Problem Statement 47

3.2.System Model 48

3.3.Chance Constrained Method

3.3.1. Open-loop Chance Constrained Method 50

3.3.2. Closed-loop Chance Constrained Method with LQG-MP 53

3.4.Experimental Slip Calculation 57

3.5.Simulation Results 60

3.6. Experimental Results for Uncertainty Prediction 64

4. Iterative Motion Planning Methodology Using Online Slip Estimation 70

4.1. Problem Statement and overall Methodology 72

4.2. System Model 75

4.3. Online Slip Estimation 76

4.4.Simulation Results 80

5. Conculsions 87

5.1. Future Work 88

Bibliography 90

List of Figures

Figure 1.1. Stanley and Talos in DARPA challenges 18

Figure 1.2. Cell decomposition and visibility graph 20

Figure 1.3. PRM and RRT 21

Figure 1.4. Obstacle dilation and potential function method 23

Figure 1.5. Minkowski sum for bounded uncertainty method 23

Figure 1.6. CC-RRT* 24

Figure 1.7. Tracked vehicle and track mark 25

Figure 1.8. Slip estimation result using camera vision and D* planner 26

Figure 2.1. Diagram of simple kinematic differential drive 32

Figure 2.2. RRT and RRT* 35

Figure 2.3. Dubins curves 37

Figure 2.4. "Repropagation" step 38

Figure 2.5. Dubins vehicle simulation 41

Figure 2.6. Approximate method simulation 42

Figure 2.7. Number of sampled nodes vs optimal cost plot 43

Figure 2.8. 5 simulation trials with approximate method 44

Figure 2.9. Dubins method and approximate method comparison 44

Figure 3.1. RRT* and CC-RRT* 53

Figure 3.2. Open-loop and closed-loop uncertainty propagation 56

Figure 3.3. Uncertainty measurement for each vehicle model 58

Figure 3.4. The map scenario for simulation 60

Figure 3.5. Simulation result with proposed algorithm 62

Figure 3.6. Distance vs uncertainty plot 62

Figure 3.7. Simulation result with dilated obstacle method 63

Figure 3.8. Simulation result for 5 trials 63

Figure 3.9. iRobot *Packbot* 65

Figure 3.10. The trajectory used for the experiment 65

Figure 3.11. Diagram of trajectory deviation 66

Figure 3.12. Distribution of deviation for 2 m trajectory 66

Figure 3.13. Distribution of deviation for 10 m trajectory 67

Figure 3.14. Comparison of simulated and experimental deviation plot 68

Figure 4.1. Diagram of overall framework 72

Figure 4.2. The map scenario for simulation 81

Figure 4.3. Simulation result without slip estimation 82

Figure 4.4. Simulation result with slip estimation, but without re-planning 83

Figure 4.5. Simulation result with slip estimation and re-planning 84

Figure 4.6. Travelled distance vs parameter value plot (for Figure 4.4 and 4.5) 84

Figure 4.7. Reference trajectory with a straight line and half circles 85

Figure 4.8. Travelled distance vs parameter value plot (for Figure 4.7.) 86

List of Tables

Table 1.1. Equations of different tracked vehicle models 27

Table 2.1. Computation time for Dubins vehicle and approximate method 40

Table 3.1. Uncertainty measurement for each model 58

Table 3.2. Mean and standard deviation of the optimal cost for 5 trials 64

Table 3.3. Comparison of the covariance matrices of distribution plots 68

Table 3.4. Correlation coefficients for distribution plots 68

Table 4.1. Ratio of sum of vehicle's deviation 85

List of Algorithms

Algorithm 2.1. RRT* 33

Algorithm 2.2. "Repropagation" step 38

Algorithm 3.1. CC-RRT* 50

Algorithm 4.1. Overall framework 72

Chapter 1

Introduction

Autonomous driving technology has wide applicability inclusive to self-driving cars, military uses, exploration mission, agriculture, etc. Thanks to the diverse needs, autonomous driving technology has gained much technological maturity recently. The DARPA Grand Challenge [41] and DARPA Urban Challenge [31] have spurred this technological advance. Aside from government-led projects, commercial companies such as Google have begun their own autonomous vehicle researches. The famous "Google car", a self-driving car developed by Google, has autonomously driven over 1 million miles, collecting crucial information for the development of autonomous system.

Though autonomous driving technology has gained much technological maturity, there is still a long way to go when it comes to ensuring the safety of the autonomous system. Numerous factors such as process noise, sensor noise, incorrect model etc. can yield uncertainty in the robot's state.

The planned trajectory should remain safe despite these uncertain factors introduced in real-time operation. For this, robustness should be an important criterion during the motion planning phase.

In this chapter, a brief introduction to robust motion planning is presented. Section 1.1 provides some of the prior works related to the subject. The contribution and outline of the thesis is provided in Section 1.2.

Figure 1.1. Stanley in DARPA Grand Challenge (left) [41] and Talos in DRAPA Urban Challenge (right) [31]

1.1. Related Works

1.1.1. Motion Planning

The motion planning problem in robotics refers to finding the (optimal) trajectory for a robotic system, from its given initial state to the desired final state. There may be intolerable states, due to maximum limit of the robotic system or obstacles. Motion planning converts the high-level specifications of tasks from human to low-level description of how to move, such as control input for robots [26]. First, let's formulate the motion planning problem more formally.

Motion Planning Problem Formulation [26]

---

1. A *world* (or *workspace*) W is either $W = R^2$ or $W = R^3$.

2. $O \subset W$ is the *obstacle region.*

3. A *Robot* is $R \subset W$. It may be a collection of m rigid links, $A_1, A_2, \cdots, A_m$.

4. A *Transformation* is a function $T: R \rightarrow W$ that moves the robot in the *world.*

5. A *Configuration space* C is a set of all possible transformations that may be applied to the robot [8 and 33]. A *Configuration* $q \in C$ is an element in *configuration space. Configuration space* may be divided into C_{free} and C_{obs}, denoting *free configuration space* and *obstacle configuration space* respectively. *Free configuration space* is a set of allowable *configurations* where the *robot* does not intersect with *obstacle region. Obstacle configuration space* is a set of *configurations* where the *robot* intersects with *obstacle region.*

6. *Initial configuration* and *goal configuration* is denoted as q_I and q_G respectively. *Goal configuration* is usually relaxed to *goal region* $Q_G \subset C$, a set of *goal configurations.*

7. The motion planning algorithm must find a *path* $\tau: [0, 1] \rightarrow C_{free}$, such that $\tau(0) = q_I$ and $\tau(1) \in Q_G$. It can be thought of as the sequence of *configurations* the robot must follow in order to reach *goal region.* A *Trajectory* is similar to *path*, but it contains the information about the *input* that should be applied to the *robot* in order to follow the *path.* A *complete* motion planning algorithm must compute a continuous *path* (or *trajectory*) or correctly report in case it is not feasible.

The motion planning problem is sometimes referred to as "piano mover's problem" [38]. The problem is shown to be PSPACE-hard, and thus NP-hard, by Reif [37]. The main difficulty is that

it is highly affected by the dimension of the configuration space, which is related to the number of robotic links given in the problem.

Despite the difficulty, several algorithms have been introduced to solve the problem, which work well for a range of reasonable dimensions. Of course, all the algorithms suffer greatly when the dimension of the configuration space increases. The essence of most of the algorithms is to find a graphical structure within the configuration space such that it connects a given initial state and desired final state. The combinatorial approach finds the trajectory without resorting to approximation, with some restrictions imposed. One important restriction is that the obstacle region in configuration space must be polygonal. To name a few, the cell decomposition is popular among many combinatorial algorithms [7]. It divides the configuration space into partitions (cells) and forms a graphical structure from the decomposed components. Another popular method is to construct a visibility graph that connects the vertices of different obstacles [24]. One important characteristic of the visibility graph is that it has some degree of optimality property, i.e. it finds the minimum distance trajectory in some limited situations. Figure 1.2 shows how the graphical structure is formed for the vertical decomposition method and visibility graph method in 2-D space.

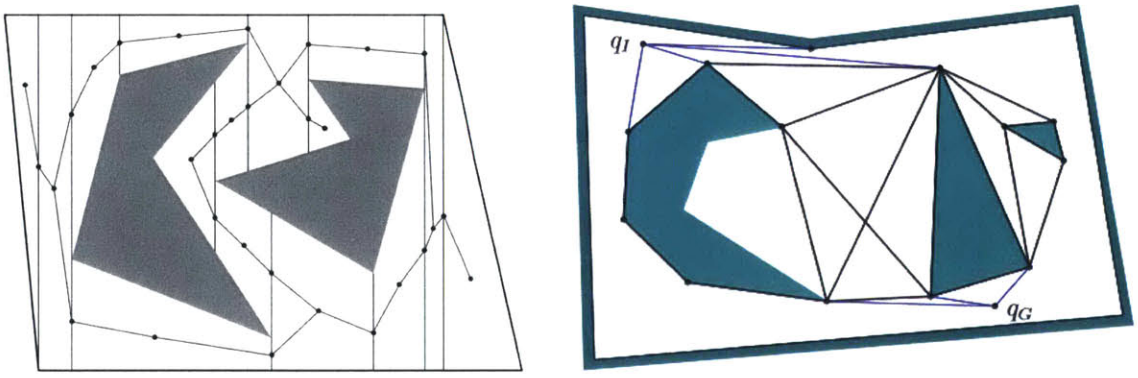


Figure 1.2. Cell decomposition (left) and visibility graph (right) in 2-D space [26]

There has been another class of approaches to the problem, called sampling based approaches [20]. These methods construct the graphical structure using randomly sampled nodes. Probabilistic roadmap (PRM) is one of the popular sampling based algorithm that is used for multi query problems [21]. The algorithm expands the graph by trying to connect the newly sampled nodes to the set of near neighbors in the existing graph. On the other hand, rapidly-exploring random tree (RRT), another popular sampling based algorithm, is used for single query problems [27 and 28]. The algorithm tries to connect the newly sampled node with the nearest neighbor, and the resulting graph is actually a tree. Due to the tree structure, searching for the solution within the tree can be done very quickly, but it is only suitable for single query problems. In addition, the RRT algorithm was originally developed for motion planning under differential constraints, which is another positive result from having a tree structure [28]. For sampling based motion planners, the completeness of the algorithm is weakened because of the random sampling. It is called resolution completeness or probabilistic completeness, which means the probability of the algorithm having the completeness property converges to 1 as the number of samples increases. Figure 1.3 shows the graph (or tree) of PRM and RRT in 2-D space.

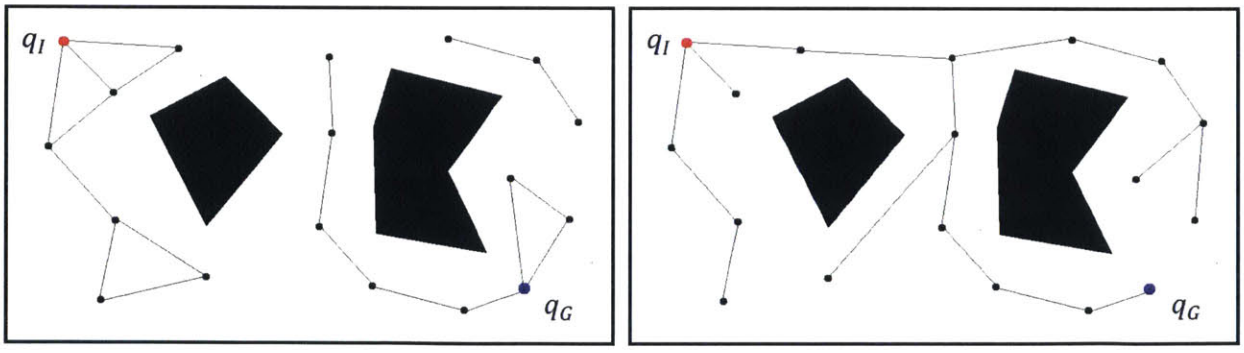


Figure 1.3. PRM (left) and RRT (right) in 2-D space. Note that PRM has loop since each new node is connected to all of its neighbors. On the other hand, RRT has tree structure since each new node is connected to only the nearest neighbor.

Recently, there has been a breakthrough with sampling based algorithms. An approach to guarantee the optimality of the solution (with respect to the specified cost function) has been developed using a "rewiring" step [17]. The resulting algorithms, called PRM* and RRT* respectively, have an asymptotic optimality property. Comparison of RRT and RRT* can be found in Figure 2.2. Asymptotic optimality means the solution converges to the optimal one as the number of nodes increases. The optimality property is not lost even when there is a non-holonomic differential constraint [18].

One important criterion for motion planning algorithms other than the completeness and optimality, is the robustness. The robustness ensures the solution found to be still valid even when there is some degree of perturbation to the robotic system. A naive way to increase the robustness of the solution is to dilate the obstacles. Feedback motion planning is another approach [26]. It tries to find the feedback control law for each state of the robotic system. The motion planning algorithm using the potential function is a good example of feedback motion planning. In fact, motion planning using potential function has different mechanism from the algorithms explained above. Rather than trying to make the graphical structure, it finds a potential function that is defined over the whole configuration space. Then, the trajectory is found using the gradient of the potential function [3 and 24]. That is, the feedback law is that the vehicle should move following the gradient of the potential function. Bounded uncertainty method tries to find the robust solution assuming the disturbance is bounded [12]. Finally, chance constrained approach assumes the state of the robotic system to be a random variable [36]. It calculates the probability of collision based on the distribution of the state. If the trajectory has the probability of collision less than some specified value, the trajectory is assumed to be safe. The chance constrained method offers some quantitative support for the solution found. Figure 1.4 shows obstacle dilation and potential function method in

2-D space. Figure 1.5 shows Minkowski sum of robot's uncertainty and disturbance, which is the essence of bounded uncertainty method. Figure 1.6 shows the diagram of chance constrained motion planning algorithm (CC-RRT*).

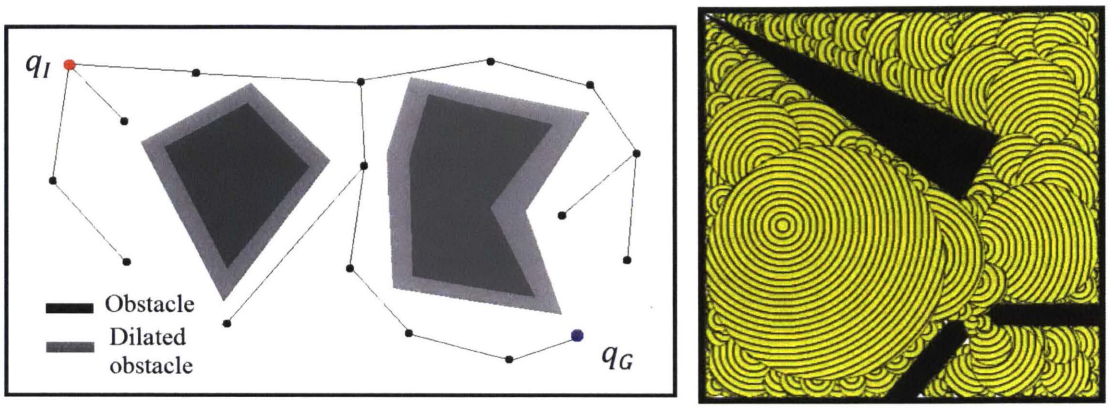


Figure 1.4. Obstacle dilation (left) and potential function method (right) [26] in 2-D space. In potential function method, circles indicate equipotential curves. Feedback law for potential function method is that the vehicle should move perpendicular to the equipotential curves.

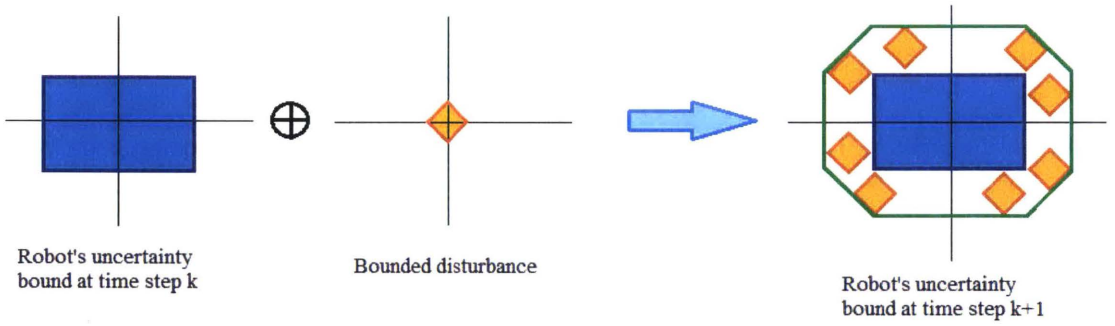


Figure 1.5. Minkowski sum of robot's uncertainty and disturbance in bounded uncertainty method

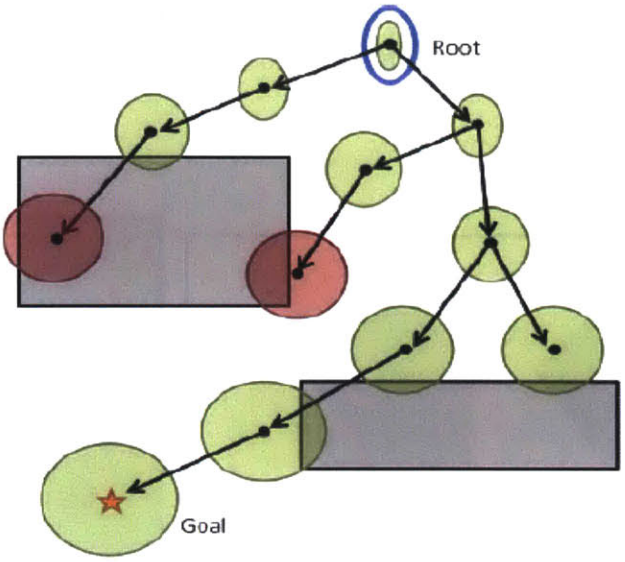


Figure 1.6. Diagram of chance constrained motion planning algorithm (CC-RRT*) [36]. Ovals along the path indicate probabilistic distributions around nodes. They can be thought of as 1-σ regions.

1.1.2. Tracked Vehicle Navigation in Off-road Condition

Robust motion planning is a huge issue especially for the tracked vehicles navigating through a rough deformable terrain. This is due to numerous uncertain factors that affect the mobility and yield vehicle state uncertainty. Slip phenomenon due to vehicle-terrain interaction can have an especially significant influence [44]. One good example for visualizing the vehicle-terrain interaction is the track mark on the ground as shown in Figure 1.7. The mark on the ground means the energy from the vehicle has not been fully converted to the mobility of the vehicle, causing the vehicle to slip [11 and 44]. In addition, each terrain interacts with the vehicle differently, resulting varying slip properties among different terrains.

Figure 1.7. Figure of a tracked vehicle (left) and its track mark (right) [11]

We can resort to Figure 3.12 to understand the significance of slip in tracked vehicle navigation. It represents the distribution of the vehicle state deviation for every 2 m piecewise reference trajectory. We can observe that the vehicle deviates as much as 0.25 m when it is traveling only for 2 m trajectory.

There have been several prior works that dealt with this issue. [14] is a well-known work for tracked vehicle navigation in high slip terrain. This work, along with many other works [1, 2, and 42], faces a big limitation, however. It presents a good slip property estimation scheme but rarely focuses on motion planning method, which is a crucial step. It estimates slip property using camera vision and plans trajectory using D* algorithm, a derivative of A* algorithm. Figure 1.8 shows some of the figures in [14]. Above one shows slip property estimation result using camera vision. Below one shows the planned trajectory using the D*. D* planner works only on grid map and it doesn't consider the dynamics of the vehicle. On the other hand, this thesis presents a rigorous motion planning scheme using CC-RRT* framework as well as an online slip estimation method.

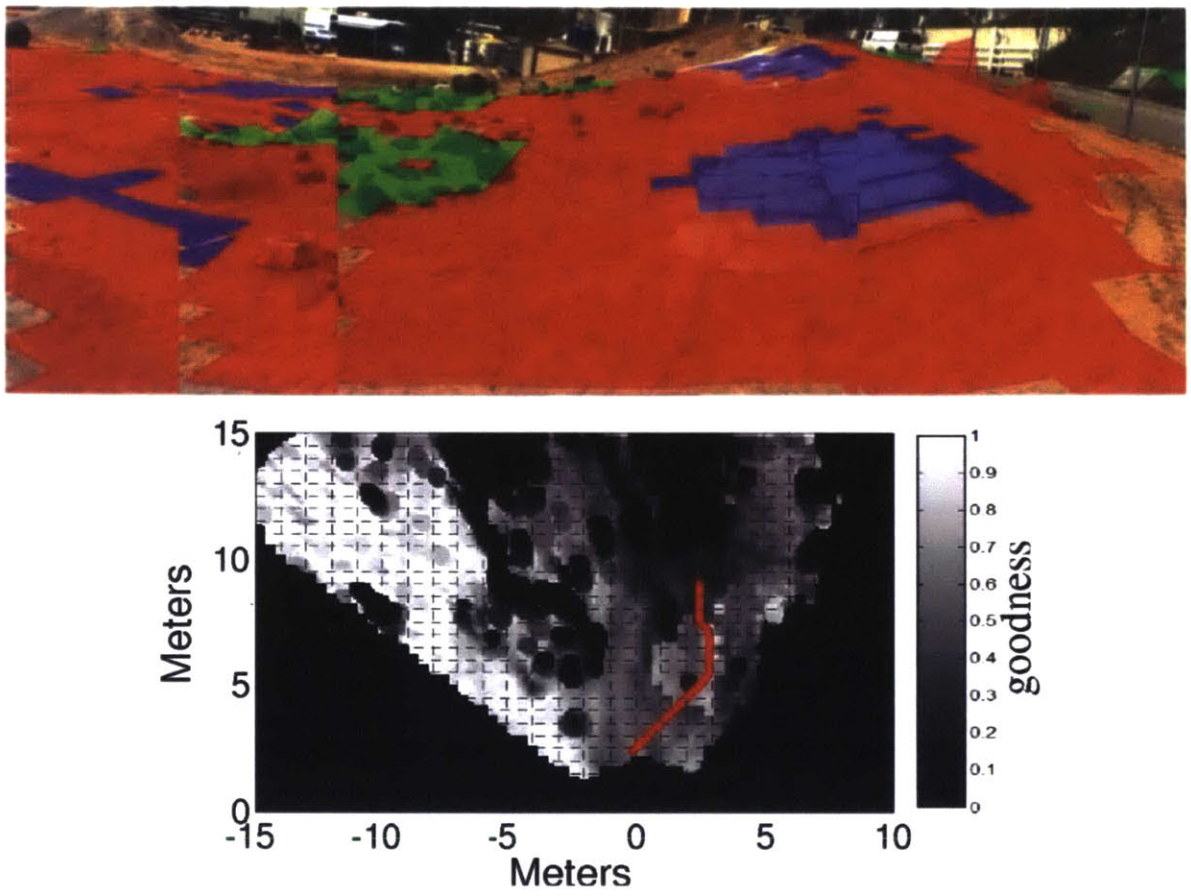


Figure 1.8. Figures from [14]. Above one shows slip property estimation result using camera vision. Different colored regions indicate regions with different slip properties. Below one shows the planned path using D* algorithm.

In addition, it is important which tracked vehicle model we use. The simplest tracked vehicle model is the kinematic differential drive model (or ideal differential drive model) shown in equation (2.3). However, this model is very simple and it doesn't consider the slip effect, there has been several tracked vehicle models developed in order to compensate for the slip effect. Some of the models are presented in Table 1.1 [9, 22, and 46].

Table 1.1. Equations of different tracked vehicle models [9]. For each variable, refer to Figure 2.1. v_{rf} and ω_{rf} denote longitudinal and angular velocity respectively. In general kinematic slip model, $C(v_{rf}, \omega_{rf})$ is a 1 by 9 matrix composed of 9 parameters.

Model	Equation
Kinematic differential drive model	$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}cos\theta & -sin\theta & 0\\ sin\theta & cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}v_{rf}\\ 0\\ \omega_{rf}\end{bmatrix}$
Effective wheel base model	$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}cos\theta & -sin\theta & 0\\ sin\theta & cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}v_{rf}\\ 0\\ \omega_{rf}/\alpha\end{bmatrix}$
General kinematic slip model	$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}cos\theta & -sin\theta & 0\\ sin\theta & cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}\left(\begin{bmatrix}v_{rf}\\ 0\\ \omega_{rf}\end{bmatrix} + C(v_{rf}, \omega_{rf}) \cdot \alpha\right)$

1.2.Summary of Contribution and Outline

As stated above, robust motion planning is a huge issue for the tracked vehicles navigating through a rough deformable terrain due to slip phenomenon. The problem becomes even more complicated when there is no prior information about the terrain that the vehicle must travel through. In this case, we cannot model the effect of the slip in advance. These difficulties should be taken into account during the planning phase of autonomous vehicle navigation.

This thesis provides a robust motion planning framework for autonomous tracked vehicles operating on off-road terrain. It is a practical methodology with wide versatility, and can be applied to other robotic systems where robust motion planning is an important issue.

In this thesis, the robust motion planning of tracked vehicles is divided into three sub-problems. First sub-problem is the optimal motion planning for non-holonomic vehicle system, since the tracked vehicle system is a non-holonomic system. A computationally efficient motion planning strategy for non-holonomic system is required. Second, modeling the uncertainty and performing robust planning is another task. The slip should be properly modeled in order to be able to predict

its impact in advance and plan the robust trajectory accordingly. The third sub-problem is the motion planning under the limited prior information about the environment. In this case, the tracked vehicle must be able to properly take the unknown factors into account and perform the motion planning.

Each of the sub-problems listed above is assigned as a topic of each chapter. Chapter 2 presents a computationally efficient sampling-based motion planning strategy for the non-holonomic tracked vehicle system using RRT*. Modeling the uncertain factors of the tracked vehicle system and ensuring robustness of the solution trajectory with chance constrained method is discussed in Chapter 3. Chapter 4 introduces an iterative robust motion planning methodology for the autonomous tracked vehicle in situation where there is not enough prior knowledge about the environment. Finally, the thesis is concluded in Chapter 5.

Chapter 2

Non-holonomic Motion Planning for Autonomous Tracked Vehicles

Many autonomous robots, such as tracked vehicles, wheeled vehicles, underwater vehicles, drones, etc., have non-holonomic dynamics, which prevents the robot to move freely in omnidirection. Unfortunately, non-holonomic dynamics introduce significant difficulty to the motion planning problem. In this case, combinatorial algorithms such as cell decomposition [7], visibility graph [24], potential function method [24], etc. are not suitable, since they are mainly developed for holonomic cases, where the robotic system can transform in the configuration space without any restriction.

As was briefly stated in Chapter 1, RRT, a sampling based motion planning algorithm has been developed for non-holonomic robotic systems, and it works well in many cases [28]. However, the problem becomes exhaustive when we need to find the optimal trajectory with respect to some specified cost function. One good news is that the optimal sampling based motion planning

algorithms, such as PRM* and RRT*, does not lose their optimality property even with non-holonomic constraints [18]. On the other hand, we need a non-holonomic steering function that finds the optimal trajectory that connects given two end nodes. This class of problems, which is called a two point boundary value problem (TPBVP) is non-trivial. It introduces a huge computational complexity to the motion planning problem [6, 23, and 25].

Due to high complexity of TPBVP, an approximate method can be used in order to gain computational efficiency. This approach assumes a constant input along the trajectory between two nodes. Then the connection to the goal node is assumed to be successful if it is close enough. Any discrepancy that occurs due to the approximation is resolved using "repropagation" step.

In this chapter, an optimal motion planning algorithm for tracked vehicle system based on RRT* is provided. It uses the approximate method together with "repropagation" step, as described above.

This chapter is organized as follows. Section 2.1 summarizes the problem statement in general form. The vehicle system model is presented in Section 2.2. Basic knowledge about the algorithms used is provided in Section 2.3. More detailed explanation about the RRT* algorithm is presented, which is the backbone of the methodology introduced in this thesis. The application of approximate method to the tracked vehicle system is also explained in this section. Section 2.4 presents simulation results, comparing the approximate method with one of the exact connection method.

2.1. Problem Statement

The motion planning problems in general can be formulated mathematically in the following way. Let the state space be $X \subset R^n$ and the input space be $U \subset R^m$. The system model can be formulated in the following general form:

$$\dot{z} = f(z, u) \tag{2.1}$$

where $z \in X$ is the state vector, $u \in U$ is the input vector. For non-holonomic vehicle system, the above system equation is a non-holonomic system equation.

The motion planning algorithm should find the optimal and collision free trajectory from the initial state z_{init} to the goal region X_{goal} satisfying the system equation. The trajectory can be described as the function $\tau : [0, 1] \rightarrow X$. The optimal trajectory means that the trajectory must be the optimal one with respect to the specified cost function, $J(\tau(s))$ where $(0 \leq s \leq 1)$. The collision free trajectory means that, given the obstacle space X_{obs}, and the collision free space defined as $X_{free} = X \backslash X_{obs}$, every point in the trajectory must lie within collision free space, that is, the trajectory must be given as $\tau : [0, 1] \rightarrow X_{free}$. Thus, the problem can be formulated as follows.

$$\begin{gathered} \text{find a trajectory } \tau^* : [0, 1] \rightarrow X_{free} \text{ subject to} \\ \tau^*(0) = z_{init}, \tau^*(1) \in X_{goal} \\ \dot{z} = f(z, u) \\ \tau^* = \underset{\tau(s)}{\operatorname{argmin}} J(\tau(s)), (0 \leq s \leq 1) \end{gathered} \tag{2.2}$$

2.2. System model

In this thesis, a tracked vehicle is used as the system model. There exists many complicated tracked vehicle models as shown in Table 1.1, but the complexity of the vehicle model greatly

increases the computation time. Thus, a simple kinematic differential drive model is of interest. Figure 2.1 shows the diagram of the vehicle.

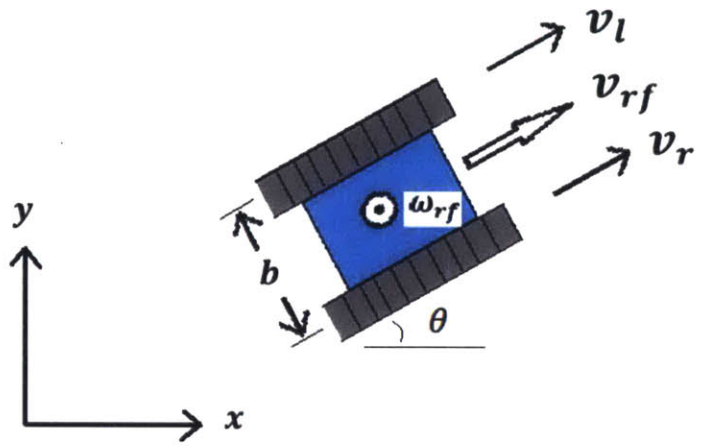


Figure 2.1. Diagram of simple kinematic differential drive [30]

Equation (2.3) shows the system equation.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{v_r + v_l}{2} cos\theta \\ \frac{v_r + v_l}{2} sin\theta \\ \frac{v_r - v_l}{b} \end{bmatrix} \tag{2.3}$$

where $z = [x, y, \theta]$ is the state vector, $u = [v_r, v_l]$ is the input vector. Parameter b indicates the distance between the vehicle's two tracks. The cost function to be optimized is the following:

$$J(\tau) = J(z, u) = t_f + \int_0^{t_f} a(t)^T R a(t) dt \tag{2.4}$$

where t_f is the final time, $a(t)$ is the acceleration, and R is the cost parameter matrix.

2.3. RRT* with Non-holonomic Steering

2.3.1. RRT* Algorithm

RRT* is a sampling based motion planning algorithm with an asymptotic optimality property [17]. The overall algorithm is shown in Algorithm 2.1.

Algorithm 2.1 : RRT*

---

1 $V \leftarrow \{z_{init}\}; E \leftarrow \emptyset;$

2 **for** i = 1, ⋯ , n **do**

3 $z_{rand} \leftarrow Sample(i);$

4 $z_{nearest} \leftarrow NearestVertex(G = (V, E), z_{rand});$

5 $Traj_{nearest} \leftarrow Steer(z_{nearest}, z_{rand});$

6 **if** $CollisionFree(Traj_{nearest})$ **then**

7 $X_{near} \leftarrow NearVertices\left(G = (V, E), z_{rand}, \min\{\gamma * \left(\frac{\log(card(V))}{card(V)}\right)^{\frac{1}{d}}, \eta\}\right);$

8 $(z_{min}, Traj_{min}) \leftarrow (z_{nearest}, Traj_{nearest});$

9 $c_{min} \leftarrow Cost(z_{nearest}) + CostFunc(Traj_{nearest});$

10 **foreach** $z_{near} \in X_{near}$ **do** // Choose parent step

11 $Traj_{near} \leftarrow Steer(z_{near}, z_{rand});$

12 **if** $CollisionFree(Traj_{near}) \wedge Cost(z_{near}) + CostFunc(Traj_{near}) < c_{min}$

13 $(z_{min}, Traj_{min}) \leftarrow (z_{near}, Traj_{near});$

14 $c_{min} \leftarrow Cost(z_{near}) + CostFunc(Traj_{near});$

15 $V \leftarrow V \cup \{z_{rand}\}; E \leftarrow E \cup \{Traj_{min}\};$

16 **foreach** $z_{near} \in X_{near}$ **do** // Rewire step

17 $Traj_{rewire} \leftarrow Steer(z_{rand}, z_{near});$

18 **if** $CollisionFree(Traj_{rewire}) \wedge Cost(z_{rand}) + CostFunc(Traj_{rewire}) < Cost(z_{near})$

19 $(z_{parent}, Traj_{parent}) \leftarrow Parent(z_{near});$

20 $E \leftarrow (E \setminus \{Traj_{parent}\}) \cup \{Traj_{rewire}\};$

21 **return** $G = (V, E)$

To briefly explain the RRT* algorithm, the algorithm returns a graph (actually, a tree for RRT* algorithm) consisting of vertices and edges. A vertex is a node that is sampled in every iteration in line 3, and an edge is a trajectory that connects two vertices. After a new vertex (z_{rand}) is sampled, the nearest vertex ($z_{nearest}$) from the existing graph is found by *NearestVertex* function. Then, the trajectory to connect from the nearest node to new node ($Traj_{nearest}$) is calculated by the *Steer* function. Here, the trajectory data structure, $Traj$ contains the path (sequence of states) and input data. If the calculated trajectory is not in collision with any of the obstacles, set of nodes that are near to z_{rand} (X_{near}) is found by *NearVertices* function. X_{near} is a set of vertices (z_{near}) whose distance from z_{rand} is smaller than $r = \min\{\gamma * \left(\frac{\log(card(V))}{card(V)}\right)^{\frac{1}{d}}, \eta\}$, where η is a constant that limits the radius in initial iterations of the algorithm. $\gamma * \left(\frac{\log(card(V))}{card(V)}\right)^{\frac{1}{d}}$ is a value that ensures the algorithm to have the asymptotically optimal property without X_{near} being a too large set [17]. Among X_{near}, the one that connects to z_{rand} with minimum cost (z_{min}) is found in line 8 – 14 ('choose parent' step). z_{rand} and the trajectory ($Traj_{min}$) from z_{min} to z_{rand} is stored into the graph in line 15. Finally, the algorithm goes through "rewiring" step, where the vertices in X_{near} is tried to be improved by using z_{rand}. If the cost to z_{near} is lowered via z_{rand}, the existing graph is modified and the new trajectory ($Traj_{rewire}$) is added.

RRT* is based on the RRT algorithm, which has no optimality property. The RRT and RRT* algorithm is compared for 2 dimensional motion planning problem in Figure 2.2. As it is stated above, the solution returned by RRT* algorithm is close to the optimal one (shortest path), whereas the solution from RRT algorithm is far from the optimal.

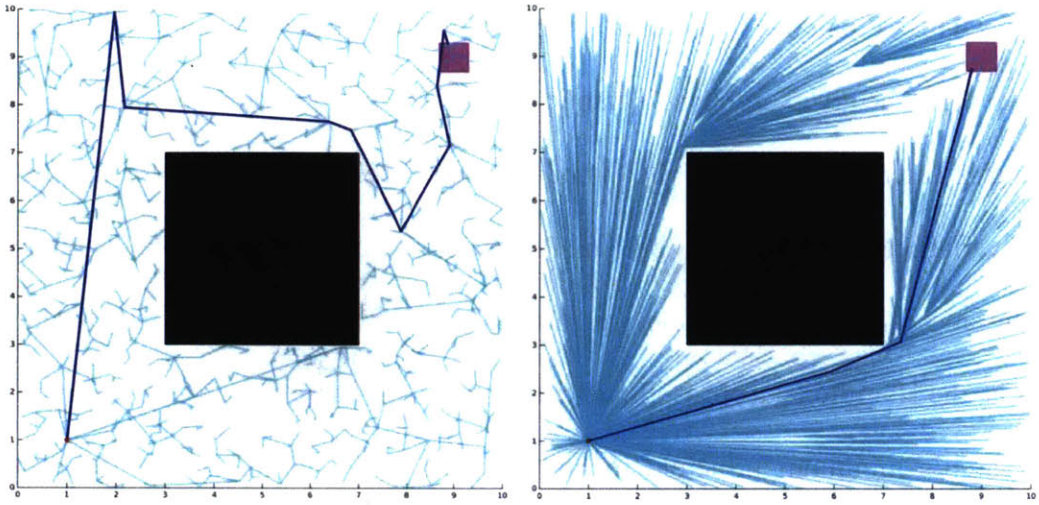


Figure 2.2. Comparison of RRT (left) and RRT* (right). Magenta region indicates goal region.

RRT* doesn't lose the optimality property even for non-holonomic systems [18]. In this case, the steering function in Algorithm 1 must solve the two point boundary value problem following the given non-holonomic dynamics.

2.3.2. Non-holonomic Steering

2.3.2.1. Exact Connection Method

The non-holonomic steering function is a TPBVP solver that returns the optimal trajectory between given two end nodes, without considering collision with obstacles. For the usual RRT* algorithm, the steering function should connect given two nodes exactly, that is, the trajectory must satisfy the given initial and final state of the system exactly. This is due to the "rewiring" step of the algorithm. If the connection from z_{rand} to z_{near} is not exact, there is going to be a discrepancy between $Traj_{rewire}$ and the trajectory between z_{near} and its children nodes.

Solving the TPBVP belongs to the area of optimal control. In optimal control, there exists two approaches to solve the given problem, dynamic programming and calculus of variations. Dynamic programming is extended to give us the Hamilton-Jacobi-Bellman equation for the problem [5]. Calculus of variations is extended to give us Pontryagin's minimum principle. Though dynamic programming and calculus of variations seem to take different approaches, they yield the same result. That is, the Hamilton-Jacobi-Bellman equation has the same form as the equation obtained from Pontryagin's minimum principle [6 and 23]. When we formulate the Hamilton-Jacobi-Bellman equation for given system equation and cost function, it is very likely that there is no analytical solution for the problem. Even for the system given in Section 2.2, which is relatively simple, there is no analytical solution. In this case, we need to use a numerical method to find the solution [6 and 23].

Though it is computationally expensive, there has been several works that used a numerical method to find the exact solution for non-holonomic steering function of RRT*. Work of Webb et al. [43] used a steering function that linearize the system equation in every small time step and find the optimal path with respect to the linearized dynamics. Though the solution uses linearization approximation, it connects given two end points exactly. In addition, if the time step is small enough, the deviation from the actual optimal solution due to linearization is not too big. Work of Ha et al. [13] used a steering function that finds the accurate optimal path following the non-linear dynamics, using an iterative method.

Those approaches have very low computation efficiency even for a simple vehicle model, such as the one in Section 2.2. References to the computational complexity can be found in [43]. Thus, in many practical implementations, a slightly different vehicle model is used [19]. Many works use

a vehicle model called Dubins vehicle, which allow us to compute the minimum distance trajectory extremely fast. The system equation for the Dubins vehicle is given as follows.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\,cos\theta \\ v\,sin\theta \\ u \end{bmatrix} \tag{2.5}$$

where v is the constant speed and $u \in [-u_{max}, u_{max}]$ is the angular input. Since the vehicle moves at constant speed, the minimum distance trajectory also becomes the minimum time trajectory. It has been proven that the solution for the minimum distance trajectory consists of only three actions, full left turn (L), straight line (S), and full right turn (R). That is, $u \in \{-u_{max}, 0, u_{max}\}$ must hold. In addition, it is also proven that the minimum distance trajectory must fall into one of the six cases, {LRL, RLR, LSL, LSR, RSL, RSR}. The distance that the vehicle must travel for each action (L, S, R) can be obtained with simple geometric calculation.

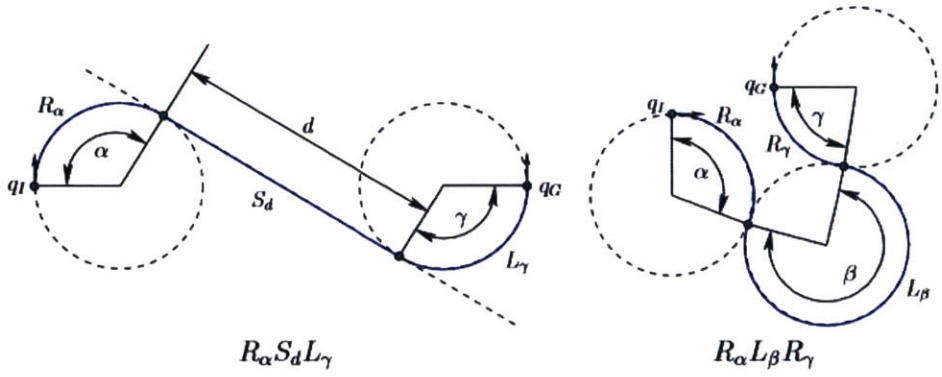


Figure 2.3. Examples of Dubins curves for RSL and RLR cases [26]

2.3.2.2.Approximate Method

An approximate steering function for non-holonomic system has been introduced to avoid heavy computation of exact connection [15]. Let's assume we are to find a trajectory between two given nodes z_{start} and z_{end}. In the approximate method, the input is assumed to be constant along the

trajectory. This assumption greatly simplifies the problem, however it also makes the exact connection to the endpoint difficult. Instead of trying to make the exact connection, the method considers the local steering to be successful if z_{end} (desired endpoint) and z'_{end} (approximate) are close, that is, $\|z_{end} - z'_{end}\| < \varepsilon$ for some specified threshold ε, and apply a corrective measure afterward to resolve the discrepancy. This corrective measure, called the "repropagation" step, is shown in Algorithm 2.2. In the "repropagation" step, all children nodes to z_{end} are regenerated from z'_{end} using inputs and time durations saved previously by calling 'ReproSteer' function. Since $\|z_{end} - z'_{end}\|$ is very small, the regenerated children nodes will not deviate much from the original ones. The "repropagation" step should be located after line 20 of Algorithm 1, within the iteration of the "rewiring" step. The "repropagation" step is visualized in Figure 2.4.

Algorithm 2.2 : Repropagate(z, z_{new})

1 **for** *all* $z' \in$ Children(z) **do**
2 $z'_{new} \leftarrow$ ReproSteer$(z_{new}$, Time(z, z'), Input$(z, z'))$;
3 Repropagate(z', z'_{new})
4 Delete z' in Tree;
5 Add z'_{new} in Tree;

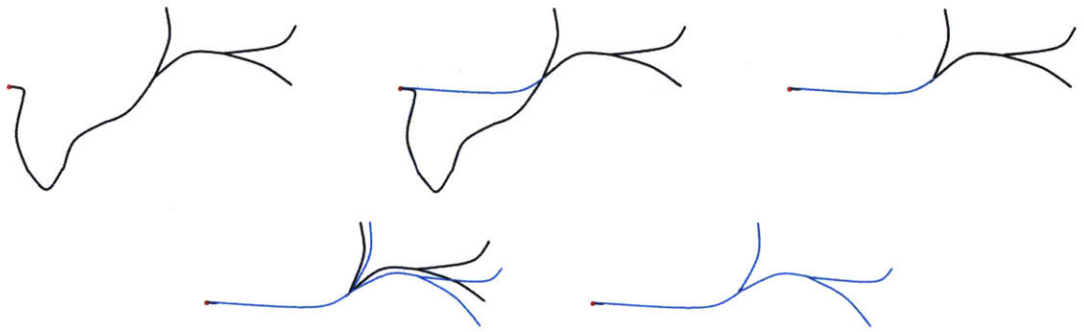


Figure 2.4. Diagram of how "repropagation" step process works

Let's apply the approximate method to our tracked vehicle system in Section 2.2. Consider the case when we are steering the vehicle from z_1 to z_2. With the system equation from equation (2.3), the trajectory becomes a circular arc when we apply the constant input assumption and neglect process noise. That is,

$$(x(t) - x_1 + \frac{b(v_r + v_l)}{2(v_r - v_l)} sin\theta_1)^2 + (y(t) - y_1 - \frac{b(v_r + v_l)}{2(v_r - v_l)} cos\theta_1)^2 = \left(\frac{b(v_r + v_l)}{2(v_r - v_l)}\right)^2 \quad (2.6)$$

When the vehicle starts from z_1 and reaches z_2 satisfying only (x_2, y_2) coordinate of z_2 under constant input assumption, there exists only one circular arc path. Equation (13) shows that the radius of the circular arc is $r = \frac{b(v_r+v_l)}{2(v_r-v_l)}$. Since the radius can be obtained purely geometrically, it specifies the velocity ratio of the right and left tracks. The vehicle can follow the given path with various speeds as long as the ratio requirement is met. To find the appropriate velocities, we can turn to the cost function. Under constant input and constant acceleration assumption, the cost function becomes the following:

$$J(X, u) = t_f + a^T Ra \times t_f \quad (2.7)$$

The above cost function has three variables, v_r, v_l, and t_f. It becomes a function of only one variable, t_f, when we use the following two relations. First, the radius of the arc equals to $\frac{b(v_r+v_l)}{2(v_r-v_l)}$. Second, the length of the arc equals to $\left(\frac{v_r+v_l}{2}\right) \times t_f$. Then, we can differentiate the cost function with respect to t_f and find the optimal t_f that minimizes the cost. The entire path and input (v_r and v_l) can be obtained using the optimal t_f.

This steering function does not connect the given two states exactly. The (x_2, y_2) coordinate of z_2 is met, but θ_2 might be different. Thus, reapply "repropagation" step to all the children nodes and the "rewiring" step is complete.

2.4. Simulation Result

2.4.1. Dubins model

In this section, an exact connection method is presented. We must formulate Hamilton-Jacobi-Bellman equation of the vehicle system given in Section 2.2 and solve it numerically in order to be strict. However, as it was stated a number of times, the computation time is extremely large. Reference to the computational complexity can be found in [43]. Thus, we will compare the approximate method with the exact method using Dubins vehicle model. The simulation result with Dubins vehicle model is shown in Figure 2.5. Table 2.1 shows the computation time for two cases, the Dubins vehicle model method and the approximate method both with 5000 nodes.

Table 2.1. Table of computation time for two cases, the Dubins vehicle model method and the approximate method. 5000 nodes were sampled for both methods.

	Exact method with Dubins vehicle	Approximate method
Computation time (sec.)	821.9	1018.4

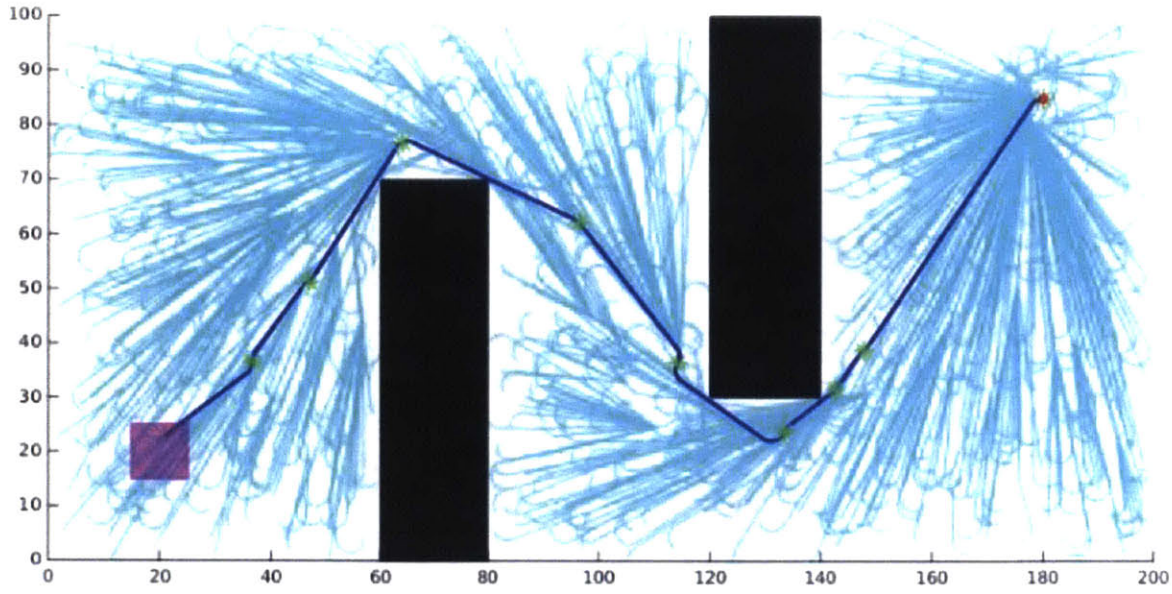


Figure 2.5. Figure of the Dubins vehicle simulation

Dubins vehicle model offers computational efficiency. However, using Dubins vehicle model has several drawbacks. First, the trajectory calculated makes turns only in fixed radius. Second, the cost function has to be the distance of the trajectory. The cost function lacks a term related to the energy efficiency, which is usually a function of the input.

2.4.2. Approximate Method

An approximate method using "repropagation" step is presented in this section. The vehicle model from Section 2.2 was used. The detailed technique is presented in Section 2.3.2.2. The approximate method has several advantages over the exact connection method. First, it is computationally efficient. This can be shown in Table 2.1 above. Second, it has very smooth path. This is because the turning radius is not fixed, unlike the case of Dubins vehicle model method.

Third, it can consider the energy efficiency term in the cost function. Figure 2.6 shows the result of the approximate method.

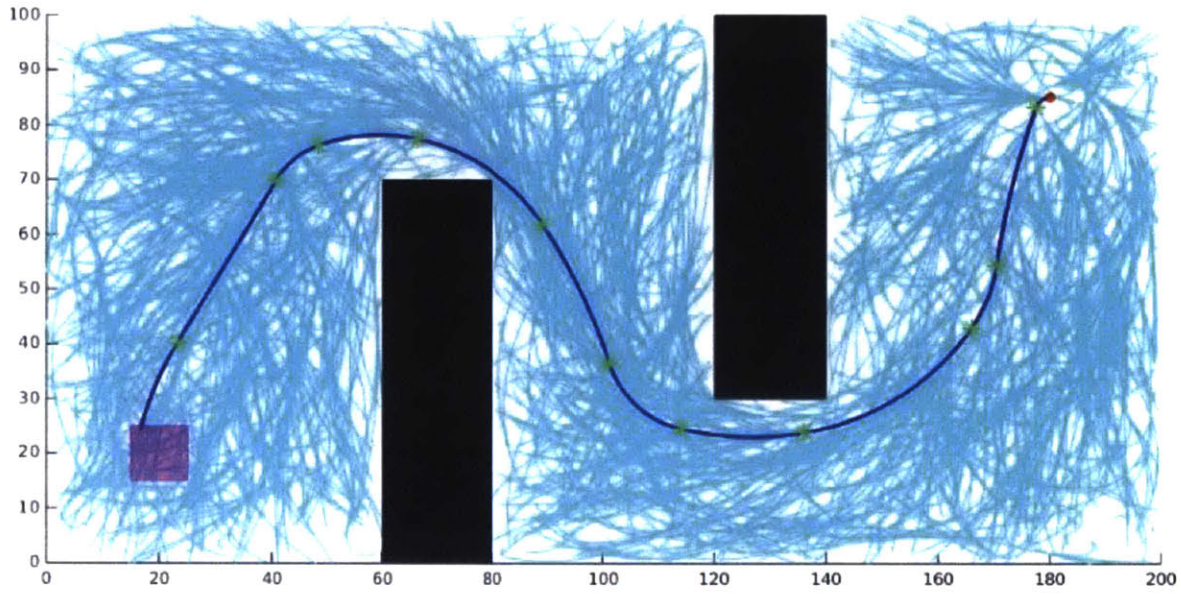


Figure 2.6. Figure of the approximate method simulation

One drawback of the approximate method is that it needs more nodes to converge to the true optimal solution than the exact method. This is because of two reasons. Firstly, there is an additional constraint for the "rewiring" to happen for the approximate method, which is the condition $\|z_{end} - z'_{end}\| < \varepsilon$. This significantly reduce the number of "rewiring" occurrence. For example, when both exact method with Dubins vehicle model and the approximate method were run for 5000 nodes, the total "rewiring" happened were 1163 times and 3227 times respectively. Secondly, because the "repropagation" step slightly changes the existing tree, previously collision free nodes sometimes can become colliding nodes. Thus, even when we sample 5000 nodes, the actual number of collision free nodes in approximate method is slightly smaller. Since the steering function for both method takes similar time, this means the computation time is bigger. For our map scenario,

the exact method needs about 3000 nodes and the approximate method needs about 5000 nodes for the convergence of optimal costs calculated. This is shown in Figure 2.7.

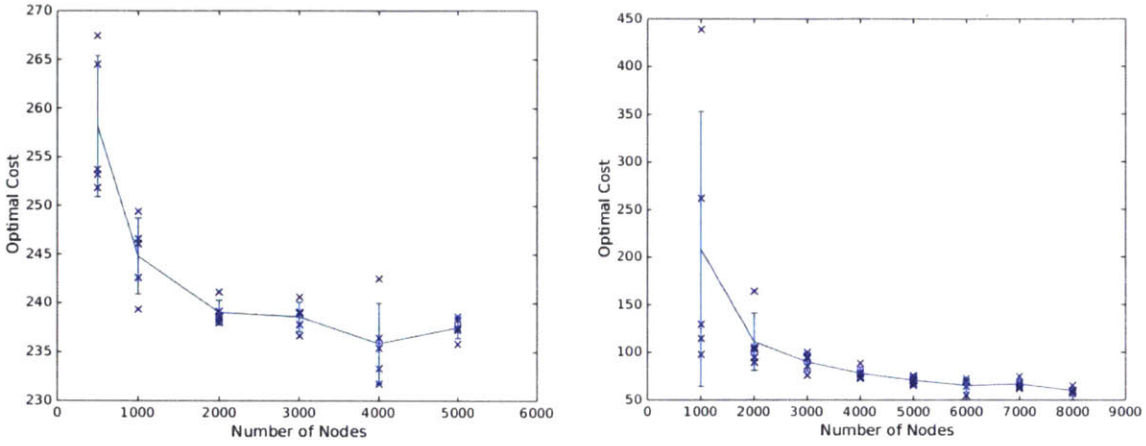


Figure 2.7. Calculated optimal cost with respect to the number of nodes sampled, for both exact method (left) and approximate method (right). Each number of nodes is tested for 5 trials indicated as x-shaped points. The error bar indicates 1-sigma region. The mean for each number of nodes in connected as a graph. The exact method needs about 3000 nodes and the approximate method needs about 5000 nodes for convergence.

Figure 2.8 presents the optimal trajectories calculated with 5000 nodes for 5 trials. The calculated optimal trajectories show some variance because they haven't fully converged to the optimum.

Figure 2.9 presents the optimal trajectories calculated for a different map scenario. Blue curve indicates the result with the approximate method and red curve indicates the result with the exact method using Dubins vehicle model. Optimal costs calculated were 94.2133 and 283.4188 respectively.

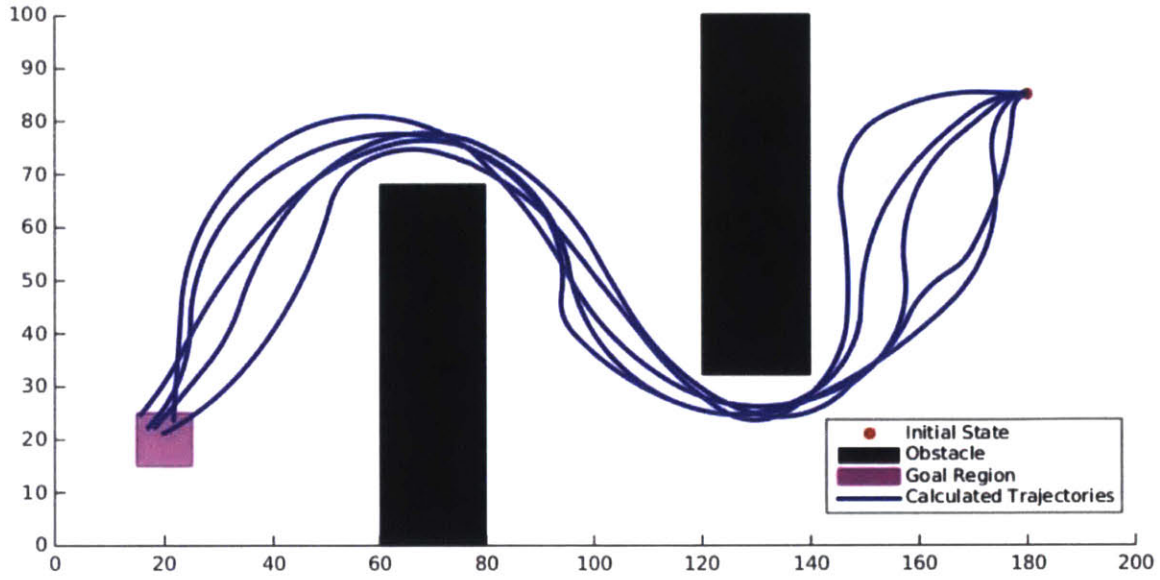


Figure 2.8. Figure of 5 trials for 5000 nodes

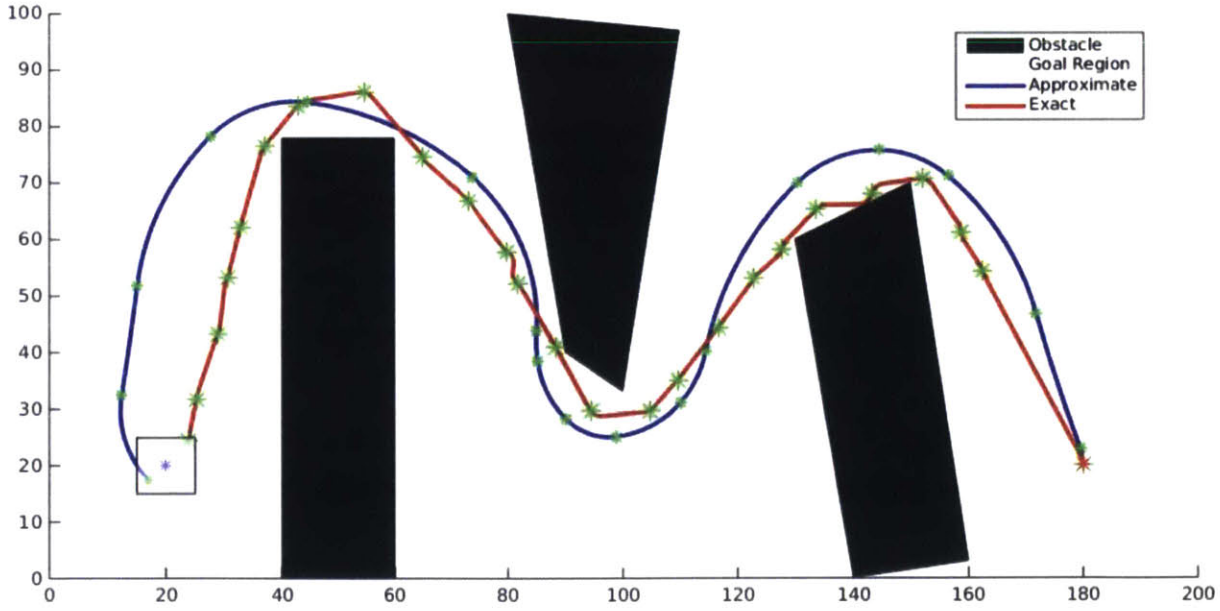


Figure 2.9. Results for approximate and exact methods with a different map scenario

Despite the drawback explained above, the approximate method holds great advantages. Thus, the approximate method is going to be used in this thesis.

Chapter 3

Robust Global Motion Planner for Autonomous Tracked Vehicles with Chance Constrained Approach

RRT* algorithm is unsuitable for motion planning of autonomous tracked vehicles operating on off-road condition, due to numerous uncertain factors that affect the vehicle mobility. Slip phenomenon due to vehicle-terrain interaction can have an especially significant influence. The slip effect can be modeled as a process noise with high uncertainty, and the resulting vehicle state uncertainty can be taken into account in the planning phase of autonomous vehicle operation. Using a chance constrained approach and calculating the chance of collision by considering the vehicle state as a random variable rather than a deterministic variable can ensure the robustness of the motion planner.

CC-RRT* combines a chance constrained approach and the RRT* algorithm to obtain robust and optimal trajectory [35]. This algorithm has been proven to be very effective. However, CC-RRT* is usually implemented in open loop fashion and the vehicle's state uncertainty is propagated

without considering the role of a compensator. The problem is that uncertainty can grow exponentially due to accumulating process noise. This is usually not the case in real operation where a closed-loop controller and state estimator (e.g. relying on GPS) are used. The role of a compensator can be taken into account by combining the CC-RRT* framework with another algorithm called LQG-MP [4]. When the two algorithms are combined, the degree of uncertainty in vehicle's state is bounded, thus offering a more realistic and less conservative planning strategy.

In order to use the CC-RRT* (or LQG-MP) framework, the influence of slip phenomenon on the vehicle state must be modeled prior to the planning phase. A simple experiment shown in [9] provides an effective way to represent the uncertainty. Assuming that the slip affects the vehicle system as a process noise, method in [9] is used to obtain the distribution of the process noise term. The modeled process noise then can be used to calculate the distribution of the vehicle state at time t.

In this chapter, the CC-RRT* algorithm is combined with LQG-MP framework to find the optimal and robust trajectory for an autonomous tracked vehicle traveling on flat, deformable terrain. Experimental data presented in [9] is used to model the influence of slip on the vehicle state. This chapter is based on the work of Lee et al. [30].

This chapter is organized as follows. Section 3.1 demonstrates the problem statement in general form. Vehicle system model is presented in Section 3.2. Basic knowledge about the algorithms used is provided in Section 3.3. The application of LQG-MP to CC-RRT* framework is explained. How to acquire the distribution of the slip process noise from an experimental data is presented in Section 3.4. Section 3.5 presents simulation results. The results from two different approaches of considering uncertainty are compared. A rigorous experimental validation of the framework is presented in Section 3.6.

3.1. Problem Statement

Motion planning problems of chance constrained method has slight different form than the one presented in Section 2.1. This is due to the fact that we are considering the state of the vehicle as a random variable, rather than a deterministic one. Let the state space be $X \subset R^n$ and the input space be $U \subset R^m$. The system model can be formulated as the following with slight modification from Section 2.1:

$$\dot{z} = f(z, u) + w \tag{3.1}$$

where $z \in X$ is the state vector, $u \in U$ is the input vector and w is the process noise into the system. Here, the slip of the vehicle is included as the simplest form, it is assumed to be an additive process noise. In addition, when the vehicle moves along the trajectory that traverses through more than one type of terrain, the process noise should be different for each terrain types. This is because each terrain type will interact with the vehicle differently and this will result in the varying slip effects among different terrain types.

The motion planning algorithm should find the optimal and probabilistically feasible trajectory from the initial state z_{init} to the goal region X_{goal} satisfying the system equation. Probabilistically feasible trajectory means that, given the obstacle space X_{obs}, every point in the trajectory given as $\tau : [0, 1] \rightarrow X$ must always have smaller probability of collision than some threshold value δ, that is, $\mathrm{P}(\tau(s) \in X_{obs}) < \delta, \forall t, (0 \leq s \leq 1)$.

Thus, the problem can be formulated as follows.

$$
\begin{aligned}
&\text{find a trajectory } \tau^* : [0,1] \rightarrow X \text{ subject to} \\
&\tau^*(0) = z_{init}, \text{(where } \tau^*(0) \text{ is assumed to be perfectly determined)} \\
&\mathrm{P}\left(\tau^*(1) \in X_{goal}\right) > \alpha, \text{(for some parameter } \alpha) \\
&\dot{z} = f(z,u) + w_k, \text{(different } k \text{for each terrain type)} \\
&\mathrm{P}(\tau(s) \in X_{obs}) < \delta, (0 \leq s \leq 1, \delta \text{ a parameter}) \\
&\tau^* = \underset{\tau(s)}{\operatorname{argmin}} J(\tau(s)), (0 \leq s \leq 1)
\end{aligned}
\tag{3.2}
$$

3.2. System Model

The system model is very similar to the one presented in Section 2.2, with slight modification. A simple kinematic differential drive model is of interest with the additive slip process noise. Equation (3.3) shows the system equation of the vehicle.

$$
\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{v_r + v_l}{2} cos\theta \\ \frac{v_r + v_l}{2} sin\theta \\ \frac{v_r - v_l}{b} \end{bmatrix} + w_k \tag{3.3}
$$

where w_k is Gaussian white noise with zero mean, which differs with respect to the type of terrain. The slip is has the additive process noise form.

The cost function has the same form as the one in Section 2.2. In some cases, the degree of uncertainty of the vehicle state is merged into the cost function (mainly by using the expected value

of the vehicle's state). However, in order to simplify the problem, the cost function and the uncertainty of the vehicle is decoupled and considered separately.

In order to use the CC-RRT* and LQG-MP framework, we need to linearize and discretize the system equation and the cost function. The linearization must be done to maintain the assumption that the state distribution follows Gaussian distribution. This is a strong approximation, but the validity of this assumption is presented in Section 3.6. The linearized and discretized system equation and the cost function are as follows [40].

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} =$$

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \Delta t \times \begin{bmatrix} 0 & 0 & -\frac{v_r + v_l}{2} sin\theta_t \\ 0 & 0 & \frac{v_r + v_l}{2} cos\theta_t \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \Delta t \times \begin{bmatrix} \frac{1}{2} cos\theta_t & \frac{1}{2} cos\theta_t \\ \frac{1}{2} sin\theta_t & \frac{1}{2} sin\theta_t \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix} \begin{bmatrix} v_r \\ v_l \end{bmatrix} \tag{3.4}$$

$$+ C_t + w_{discrete,k,t}$$

where C_t is a constant matrix due to linearization and $w_{discrete,k,t}$ is the discrete time Gaussian white noise. The cost function becomes:

$$J(X,u) = N \times \Delta t + \sum_{i=1}^{N} a_i{}^T R a_i \times \Delta t \tag{3.5}$$

where $N \times \Delta t = t_f$.

3.3. Chance Constrained Method

3.3.1. Open-loop Chance Constrained Method

Chance constrained method is used for robust motion planning, considering the state of the robot to have a probabilistic distribution. In many cases, the distribution is assumed to be Gaussian in order to simplify the calculation. CC-RRT* is an extended version of RRT* algorithm using chance constrained method [35]. Thus, it has both optimality and robustness property. The diagram of CC-RRT* is presented in Figure 1.6. Brief pseudo code for the CC-RRT* algorithm is provided in Algorithm 3.1.

Algorithm 3.1 : CC-RRT*

---

1 $V \leftarrow \{z_{init}\}; E \leftarrow \emptyset;$

2 **for** i = 1, ⋯ , n **do**

3 $z_{rand} \leftarrow Sample(i);$

4 Connect to *ProbFeas* nearest node → $(z_{nearest}, Traj_{nearest})$;

5 $X_{near} \leftarrow NearVertices(G = (V, E), z_{rand});$

6 Choose *ProbFeas* and min cost parent node among X_{near} → $(z_{min}, Traj_{min})$;
// Choose parent step

7 **if** z_{rand} reduces cost to any nodes in X_{near} and *ProbFeas*

8 Rewire and Repropagate;
// Rewire step

9 **return** $G = (V, E)$

The overall algorithm is almost the same as the RRT* algorithm presented in Section 2.3.1. The only change that needs to be made is to change the $CollisionFree$ function to *ProbFeas* function,

which checks the probabilistic feasibility of the node. Thus, the trajectory data structure must contain the covariance data along the trajectory.

Before we check the probabilistic feasibility, we need to calculate the probabilistic distribution of the given node. Since the state of the vehicle (z_t) is assumed to follow a Gaussian distribution, $z_t \sim N(Ez_t, P_{x_t})$ holds. First, we can write the equation (3.4) as the following.

$$z_{t+1} = A_t z_t + B_t u_t + C_t + w_{discrete,k,t} \tag{3.6}$$

Then, the mean and covariance at time $t + 1$ is given as follows.

$$\begin{aligned} E(z_{t+1}) &= A_t E(z_t) + B_t u_t + C_t \\ P_{x_{t+1}} &= A_t P_{x_t} {A_t}^T + P_{w_{discrete,k,t}} \end{aligned} \tag{3.7}$$

where $P_{w_{discrete,k,t}}$ is the covariance matrix of $w_{discrete,k,t}$. Here, we make an approximation: If the time step is small enough, the expected value of the state obtained from equation (3.7), $E(z_{t+1})$, should accurately approximate that from nonlinear equation, equation (3.3), at the same instance. Thus, the state obtained from the nonlinear equation is going to be used to obtain the mean state, $E(z_{t+1})$, and equation (3.7) is going to be used to obtain the covariance matrix only.

Next, let's check the probabilistic feasibility of the given node with distribution $z_t \sim N(E(z_t), P_{x_t})$. Suppose all obstacles are convex polyhedral, then the state z_t being inside of the obstacles (and thus colliding) can be expressed as $\bigwedge_{i=1}^{n_j} {a_{ij}}^T (z_t - c_{ij}) < 0$ where i stands for i^{th} face of polyhedron, j stands for j^{th} obstacle, and n_j is the number of faces in j^{th} polyhedron. Then the probability of collision with j^{th} obstacle satisfies the following inequality.

$$\mathrm{P}\left(\bigwedge_{i=1}^{n_j} {a_{ij}}^T(z_t - c_{ij}) < 0\right) \leq \mathrm{P}\left({a_{ij}}^T(z_t - c_{ij}) < 0\right) = \Delta_{ijt} = \frac{1}{2}\left(1 - \mathrm{erf}\left(\frac{{a_{ij}}^T(Ez_t - c_{ij})}{\sqrt{2{a_{ij}}^T(P_{x_t})a_{ij}}}\right)\right), \forall i \tag{3.8}$$

Here, Δ_{ijt} is the probability of collision of the vehicle with i^{th} face of polyhedron and j^{th} obstacle at time t. The upper bound for the probability of collision with j^{th} obstacle at time t (Δ_{jt}) is the minimum of Δ_{ijt} for all i, that is $\Delta_{jt} = \min_{i=1,\cdots,n_j} \Delta_{ijt}$. Finally, The probability of collision at time t satisfies the following equation

$$\mathrm{P}(\text{Collision at time } t) \leq \sum_{j=1}^{n_o} \mathrm{P}(\text{Collision with obstacle } j \text{ at time } t) \leq \sum_{j=1}^{n_o} \Delta_{jt} = \Delta_t \tag{3.9}$$

where n_o is the number of obstacles. Then, the *ProbFeas* function returns 'safe' if $\Delta_t < \delta$ for some threshold δ. $\Delta_t < \delta$ only checks the probabilistic feasibility of a specific node at time t. We can also check the probabilistic feasibility of the trajectory as a whole. The probability of collision for the trajectory from $t = 0$ to $t = t_f$ (for discrete time sequence) satisfies the following equation.

$$\begin{aligned} \mathrm{P}\left(\text{Collision of the trajectory from } t = 0 \text{ to } t = t_f\right) &= \mathrm{P}\left(\bigvee_{t=0}^{t_f} z_t \in X_{obs}\right) \\ &\leq \sum_{t=0}^{t_f} \mathrm{P}(z_t \in X_{obs}) \leq \sum_{t=0}^{t_f} \Delta_t = \Delta \end{aligned} \tag{3.10}$$

Then, the trajectory from $t = 0$ to $t = t_f$ is said to be probabilistically feasible if Δ is smaller than some threshold, δ_p, that is, $\Delta < \delta_p$. However, this method is overly conservative and considers only discrete time sequence whereas the trajectory is defined over a continuous time interval. The method becomes conservative when we replace the probability of collision from time 0 to t_f with

the summation in equation (3.10) (at the first inequality). Thus, in this thesis, only the probabilistic feasibility of each node at time t is going to be checked.

Figure 3.1 compares the solution obtained from CC-RRT* with that from RRT*. Non-holonomic vehicle model presented in equation (2.3) and (3.3) is used. The green oval along the trajectory indicates 95% confidence region of the vehicle state. Thus, CC-RRT* algorithm offers a quantitative robustness of the solution in terms of collision with obstacles.

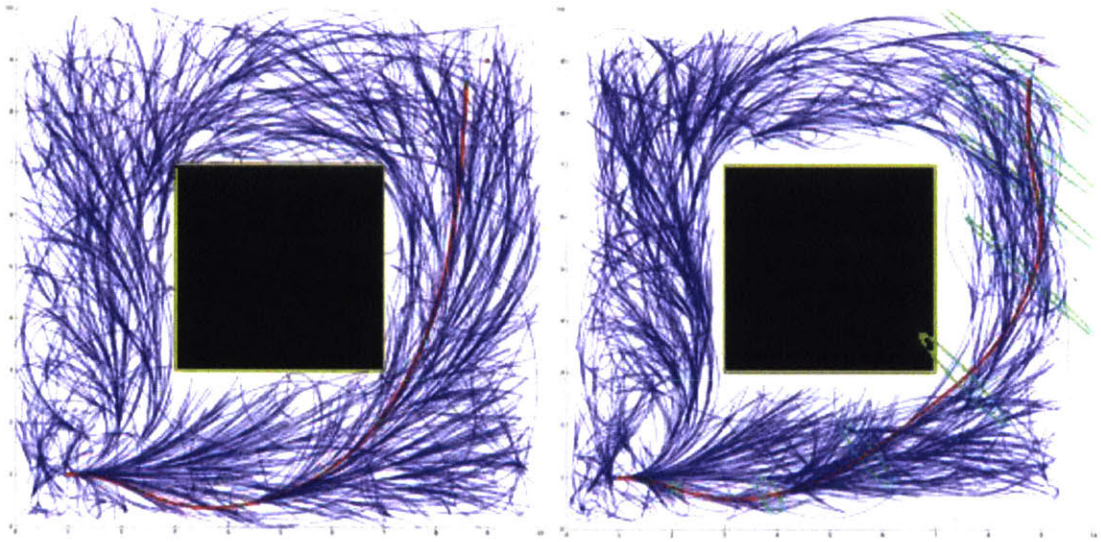


Figure 3.1. RRT* (left) and CC-RRT* (right) in 2-D space.

3.3.2. Closed-loop Chance Constrained Method with LQG-MP

The chance constrained method is usually implemented in open loop fashion, i.e. doesn't consider closed loop compensator that is used in most real world scenarios. This often makes the solution overly conservative. Closed-loop uncertainty propagation resolves this issue. Closed-loop uncertainty propagation can be done by combining CC-RRT* algorithm with LQG-MP framework. LQG-MP algorithm can be merged in the part where we calculate the probabilistic distribution of

the given node, $z_t \sim N(Ez_t, P_{x_t})$. Using the discretized and linearized equation in equation (3.4), the vehicle and measurement equations can be represented as follows.

$$\begin{aligned} z_{t+1} &= A_t z_t + B_t u_t + C_t + w_t \\ y_{t+1} &= H_{t+1} z_{t+1} + v_{t+1} \\ \hat{z}_{t+1} &= A_t \hat{z}_t + B_t u_t + C_t + L_t(y_{t+1} - \hat{y}_{t+1}) \\ \hat{y}_{t+1} &= H_{t+1} \hat{z}_{t+1} \\ \bar{z}_{t+1} &= A_t \bar{z}_t + B_t u_{r,t} + C_t \\ \bar{y}_{t+1} &= H_{t+1} \bar{z}_{t+1} \\ u_t &= u_{r,t} - K_t(\hat{z}_t - \bar{z}_t) \end{aligned} \tag{3.11}$$

where y_t is the measurement, w_t is the process noise ($w_t \sim N(0, P_{w,t})$), v_t is the sensor noise ($v_t \sim N(0, P_{v,t})$), $\hat{z}_t$ is the estimate of z_t, and $\bar{z}_t$ is the reference state at time t. L_t is the estimator gain which will be assumed to be the optimal Kalman filter gain and K_t is the optimal feedback controller gain [10]. The distribution of z_t, Kalman gain, and controller gain can be obtained as follows. First, let $Z_t = \begin{bmatrix} z_t \\ \hat{z}_t \end{bmatrix}$. Then, manipulating equation (3.11), we get:

$$\begin{aligned} Z_{t+1} &= \bar{A}_t Z_t + \bar{B}_t u_{r,t} + \bar{C}_t + \bar{G}_t W_t \\ \bar{A}_t &= \begin{bmatrix} A_t & -B_t K_t \\ L_{t+1} H_{t+1} A_t & (I - L_{t+1} H_{t+1}) A_t - B_t K_t \end{bmatrix} \\ \bar{B}_t &= \begin{bmatrix} B_t \\ B_t \end{bmatrix}, \bar{C}_t \text{ is some constant term} \\ \bar{G}_t &= \begin{bmatrix} I & 0 \\ L_{t+1} H_{t+1} & L_{t+1} \end{bmatrix}, W_t = \begin{bmatrix} w_t \\ v_{t+1} \end{bmatrix} \end{aligned} \tag{3.12}$$

Then, we get state mean ($E(z_t)$) and covariance (P_t) by:

$$E(Z_{t+1}) = \bar{A}_t E(Z_t) + \bar{B}_t u_{r,t} + \bar{C}_t, E(z_t) = [1^T\ 0^T]E(Z_t)$$

$$\bar{P}_{t+1} = \bar{A}_t \bar{P}_t \bar{A}_t{}^T + \bar{G}_t \bar{P}_{W,t} \bar{G}_t{}^T, P_{t+1} = [I\ 0]\bar{P}_{t+1}\begin{bmatrix} I \\ 0 \end{bmatrix}$$

$\bar{P}_t$: covariance matrix of Z_t (3.13)

P_t : covariance matrix of z_t

$\bar{P}_{W,t}$: covariance matrix of W_t

Again, we can use the nonlinear equation from equation (3.3) to obtain the mean ($E(z_t)$) of the state instead of the linearized equation in equation (3.13), assuming that the time step is small enough. It is the same way as the one described in Section 3.3.1, right below the equation (3.7).

We get the Kalman gain by:

$$L_t = \left(A_t P_{t-1} A_t{}^T + P_{w,t-1}\right) H_t{}^T \left[H_t \left(A_t P_{t-1} A_t{}^T + P_{w,t-1}\right) H_t{}^T + P_{v,t}\right]^{-1} \quad (3.14)$$

The optimal controller gain can be obtained using the standard LQR controller.

controller cost function :

$$E\left(\sum_{t=0}^{l} ((z_t - \bar{z}_t)^T M (z_t - \bar{z}_t) + (u_t - u_{r,t})^T N (u_t - u_{r,t}))\right)$$

Solving the matrix Riccati equation backward, (3.15)

$$S_l = M,$$

$$K_t = (B_t{}^T S_t B_t + N)^{-1} B_t{}^T S_t A_t,$$

$$S_{t-1} = M + A_t{}^T S_t A_t - A_t{}^T S_t B_t K_t$$

In this thesis, however, the LQG-MP algorithm is implemented with slight modification. The distribution of z_t and Kalman gain, L_t, is obtained as explained above. On the other hand, a suboptimal controller gain is used in order to avoid solving the matrix Riccati equation recursively

and reduce the computation load. This is based on the works of Gonzalez et al. [11] and Kanayama et al. [16]. The suboptimal controller gain is of the following form:

$$K_t = \begin{bmatrix} 2\xi\beta cos\theta_t - \frac{1}{2}\gamma v_{rf,t} sin\theta_t & 2\xi\beta sin\theta_t + \frac{1}{2}\gamma v_{rf,t} cos\theta_t & \xi\beta \\ 2\xi\beta cos\theta_t + \frac{1}{2}\gamma v_{rf,t} sin\theta_t & 2\xi\beta sin\theta_t - \frac{1}{2}\gamma v_{rf,t} cos\theta_t & -\xi\beta \end{bmatrix} \tag{3.16}$$

where $\beta = \left(\left(\omega_{rf,t}\right)^2 + \gamma\left(v_{rf,t}\right)^2\right)^{\frac{1}{2}}$, γ and ξ are tuning parameters

$$v_{rf,t} = \frac{v_r + v_l}{2}, \omega_{rf,t} = \frac{v_r - v_l}{b}$$

Though this is a suboptimal controller, the results in Section 3.5 suggest that it works well in practice.

Figure 3.2 compares the open-loop and closed-loop uncertainty propagation on a simple curved trajectory.

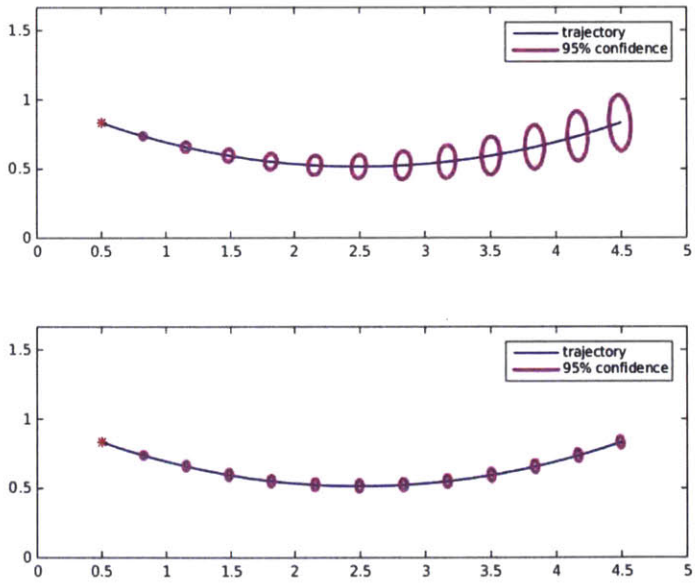


Figure 3.2. open-loop (upper) and closed-loop (lower) uncertainty propagation on a simple curved trajectory

3.4. Experimental Slip Calculation

In order to use the CC-RRT* framework, the slip uncertainty must be modeled properly. In our case, the slip uncertainty is assumed as the process noise to the tracked vehicle system, thus we need to specify the probability distribution of the noise term, $w_{discrete,k,t}$. This can be done experimentally. The process noise due to slip is assumed to be a zero mean Gaussian white noise. Thus we only need to obtain the covariance matrix, $P_{w_{discrete,k,t}}$.

The result presented in this section is based on the experimental result from trajectory generation for tracked vehicles by Fink et al. [9]. When the vehicle is assumed to be traveling on flat, off-road terrain and slip becomes the dominant source of process noise, several models already exist that can compensate for the slip effect as shown in Table 1.1 [9]. Among many, the effective wheel base model will be used in this chapter because it is both simple and powerful. This model multiplies a parameter α to the wheel base b from equation (3.3) to compensate for slip in the vehicle's turning action. This can easily be included in the framework of this thesis. The appropriate parameter value is referenced in [9]. The parameter is different with respect to different soil types.

Using the referenced parameter, the vehicle's deviation from reference trajectory can be obtained in a form of a distribution. That is, for every piecewise reference trajectory, we can calculate how much the vehicle's real state is deviated from the reference trajectory. This distribution is illustrated in Figure 3.3 and Table 3.1 for two soil types, concrete and turf. The deviation is measured for every 4 m trajectory segment.

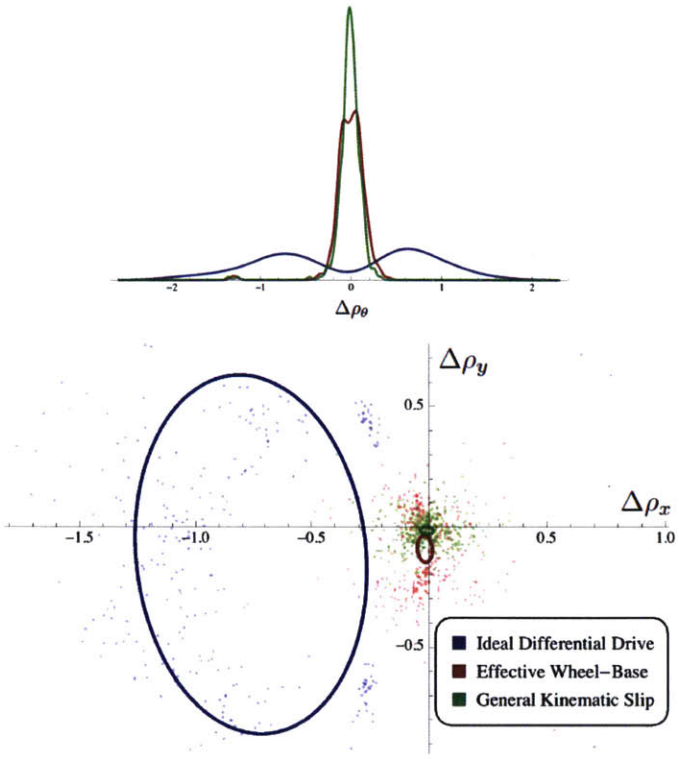


Figure 3.3. Uncertainty measurement for each vehicle model [9]

Table 3.1. Uncertainty measurement for each model [9]

	Model	$\Delta\rho_x\ (m)$		$\Delta\rho_y\ (m)$		$\Delta\rho_\theta\ (rad)$	
		μ	σ^2	μ	σ^2	μ	σ^2
Concrete	Ideal	-0.762	0.248	-0.115	0.372	-0.043	0.768
	Eff. WB	-0.019	0.016	-0.092	0.028	-0.006	0.026
	Gen. KS	-0.011	0.014	-0.015	0.007	-0.016	0.021
Turf	Ideal	-0.371	0.056	-0.021	0.088	-0.006	0.166
	Eff. WB	0.023	0.009	-0.043	0.007	0.019	0.009
	Gen. KS	0.016	0.008	-0.002	0.006	0.002	0.007

Eff. WB : Effective Wheel Base, Gen. KS : General Kinematic Slip

Now we need to convert the distribution from Figure 3.3 and Table 3.1 to the appropriate form so that it can be used in the CC-RRT* framework. For every Δt, the following approximate relation will be used to obtain $P_{w_{discrete,k,t}}$.

$$P_{w_{discrete,k}} \approx \left(\left(\frac{v_r + v_l}{2}\right) \times \Delta t \times \frac{1}{4}\right)^2 \times P_{exp}$$

$$P_{exp} = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix} \tag{3.17}$$

where P_{exp} is the experimental covariance matrix obtained from Table 3.1. Note $\frac{1}{4}$ is multiplied in the parentheses because the experimental data in Table 3.1 is for 4 m trajectory segments as stated previously. The algorithm is also slightly adjusted for the mean values from Table 3.1. It is because the process noise is assumed to have zero mean, even though the mean values are not exactly zero from Table 3.1. It doesn't affect much since the mean values are very small.

3.5. Simulation Results

Simulations were conducted for a differential drive vehicle operating on an artificially created scenario. Figure 3.4 shows the scenario map used in the simulation. The map size is 200 by 100 meters. It is a simulated region with obstacles and various terrain types. Black polygons indicate obstacles, such as sand dunes, ponds, buildings etc. Green (dark gray) polygons indicate wooded area, thus there is no GPS signal. In these regions, the uncertainty grows the same as it would in open loop CC-RRT*. Yellow (light gray) polygons indicate roads or parking lot, which consist of concrete. The rest of the map consists of turf. The vehicle's starting pose is (180 m, 85 m, π rad) and the vehicle's goal region is 8 m × 8 m box around the point (40 m, 30 m).

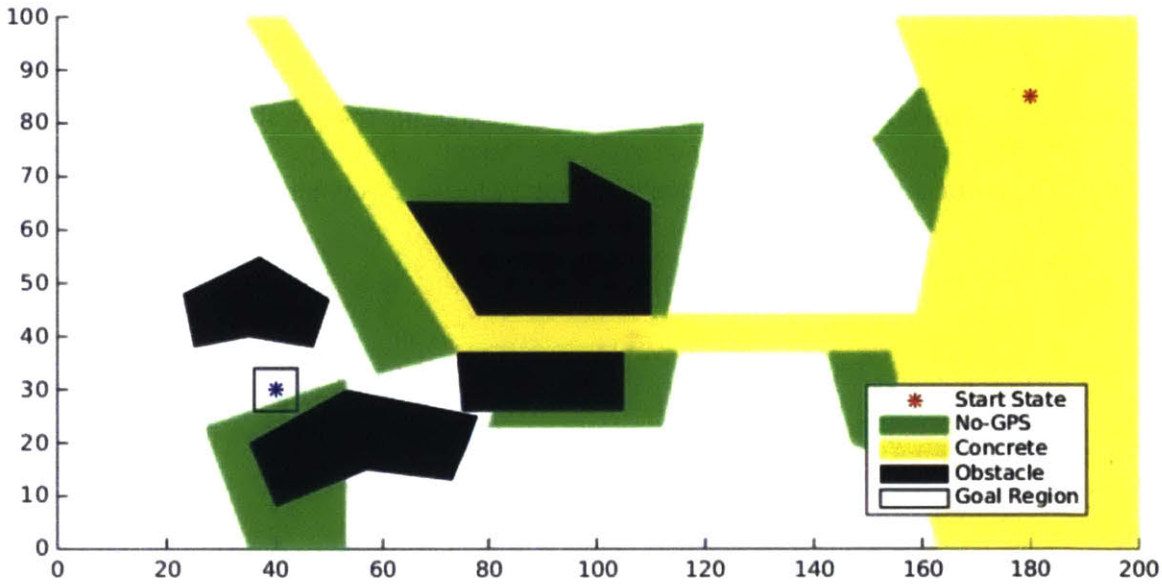


Figure 3.4. The map scenario for simulation [30]

Figure 3.5 shows the result with proposed algorithm. The figure shows the entire tree, and the optimal trajectory is thickened. Ovals around the optimal trajectory indicate the 95 % confidence region of the vehicle's state. Since the concrete is not covered with woods, GPS signals can be

received and the covariance is bounded. The optimal trajectory is through the narrow concrete corridor. In addition, after the vehicle emerged from the GPS-denied region (green or dark gray region), the uncertainty oval shrinks. It can be seen from Figure 3.5. The same phenomenon can be seen more clearly in Figure 3.6. When the vehicle is in GPS-denied region, the uncertainty grows exponentially as it would have in open loop CC-RRT*. The uncertainty shrinks after the vehicle emerged from GPS-denied regions. The change is more drastic in the second time. The uncertainty reduction is because the vehicle can compensate for path deviation once it can receive GPS signal again. There is a delay in uncertainty reduction after the vehicle emerges from no-GPS region. This is because the vehicle needs to receive GPS signal for some amount of time in order to utilize the closed loop controller to converge near the reference trajectory. Receiving GPS signal once couldn't make the closed loop controller converge to the reference trajectory right away. Finally, we can safely assume that the calculated trajectory would be robust since the uncertainty ovals do not intersect with obstacles, meaning that it is at least 95% safe (the threshold used in chance constrained method).

Figure 3.7 compares the above result with a simple robust planning strategy. Here, each obstacle is dilated by a fixed distance. Observe that the maximum standard deviation of vehicle's pose along the optimal trajectory in Figure 3.5 is ~1.6 m. The no-collision threshold used for chance constrained method is 95%, corresponding to approximately twice the standard deviation. Thus, the obstacles are extended by ~3.2 m from each face. The resulting optimal path obtained is much more conservative compared to the result of Figure 3.5.

The simulation result thus illustrates that the proposed algorithm can obtain a robust and optimal trajectory that is not prone to being overly conservative.

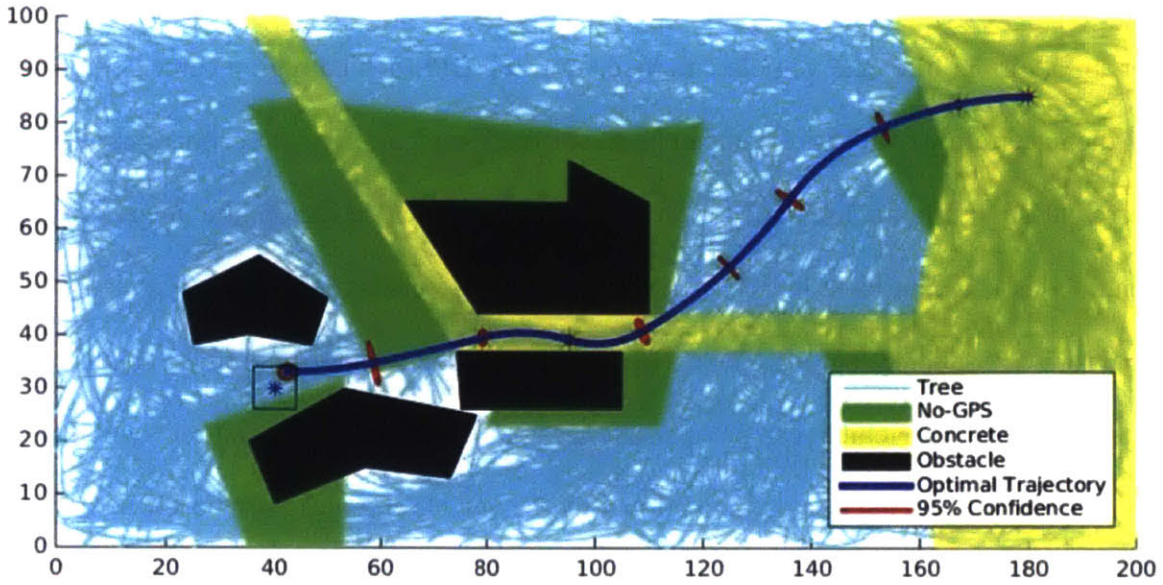


Figure 3.5. Result with proposed algorithm [30]

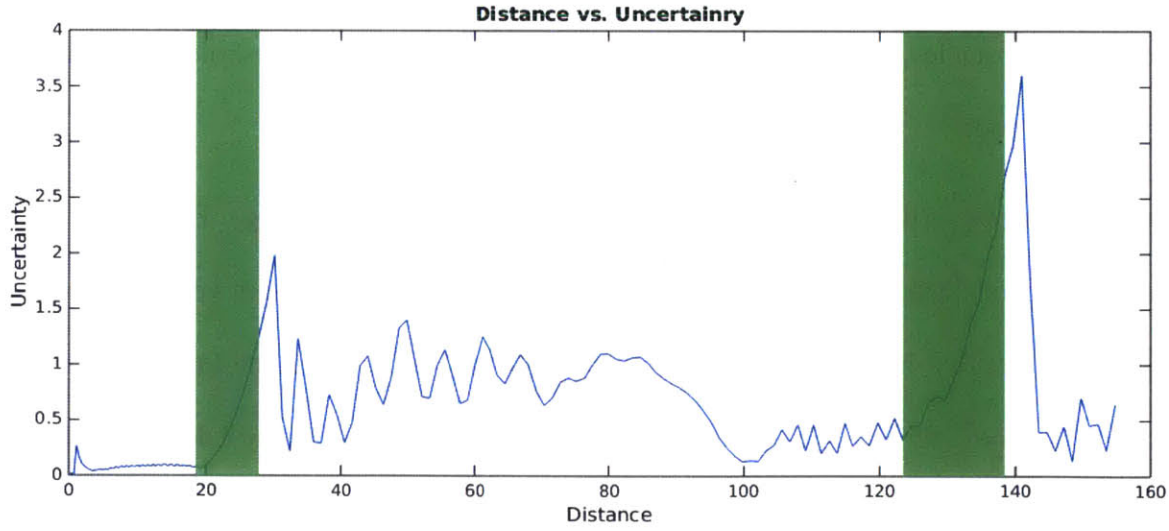


Figure 3.6. Trace of covariance matrix for state distribution z_t (only for x, y coordinates) with respect to distance traveled. The shaded area represents GPS-denied regions. See the uncertainty grows exponentially in these regions and shrinks when the vehicle exits [30].

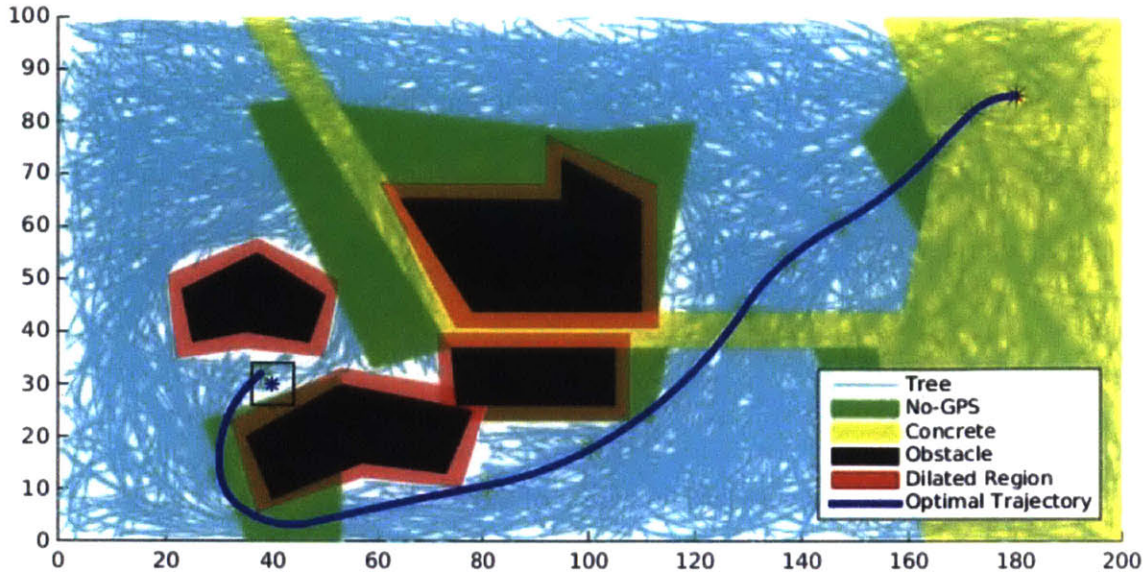


Figure 3.7. Result with dilated obstacles [30]

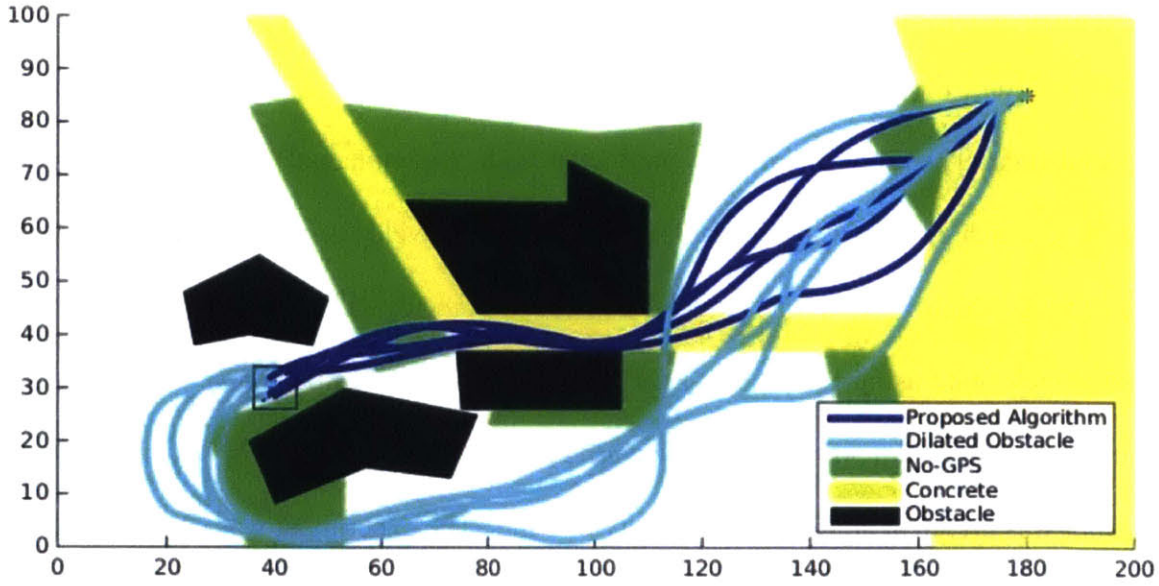


Figure 3.8. Simulation result for 5 trials [30]

Figure 3.8 shows the simulation result for 5 trials. Table 3.2 shows the mean and standard deviation of the optimal cost obtained for 5 trials. The optimal trajectory and cost calculated by the algorithm vary each trial slightly and thus fails to meet the strict definition of optimum. However, the variation is tolerable considering the fact that the algorithm succeeds to find a trajectory through the corridor.

Table 3.2. Mean and standard deviation of the optimal cost for 5 trials [30]

Scenario	**Mean optimal cost**	**Standard deviation**
Proposed algorithm	55.4488	4.8049
Obstacle dilation	75.4797	10.0830

3.6. Experimental Results for Uncertainty Prediction

In this section, the uncertainty propagation using equation (3.7) is validated through experimental data. The experiment was conducted with a commercial tracked vehicle, iRobot *Packbot* [45], which is shown in Figure 3.9. The *Packbot* traveled an arbitrary trajectory on flat concrete terrain, which is shown in Figure 3.10. Green trajectory along with grey dots show the estimated vehicle state using RTK GPS. Red trajectory shows the estimated vehicle state using a motion tracking system. An odometer was also used for tracking the velocity of left and right tracks. The *Packbot* was driven at approximate velocity of 2 m/s. The experiment was conducted with Dr. Jonathan R. Fink from the Army Research Laboratory (ARL).

Figure 3.9. iRobot *Packbot* on two different terrains; concrete and turf [9]

Figure 3.1. The trajectory used for the experiment

The vehicle was assumed to run the trajectory in open-loop manner. We can reconstruct the piecewise reference trajectories of arbitrary length using the odometer data. The data from a motion tracking system was assumed to represent the true state of the vehicle. The data from RTK GPS was not reliable. Then, for arbitrary length of the reference trajectories, we can calculate the deviation. The diagram for the concept is shown in Figure 3.11. Figure 3.12 represents the distribution of the vehicle state deviation for every 2 m piecewise trajectory. 2 m was chosen for convenience, it was the distance the vehicle travelled for 1 second.

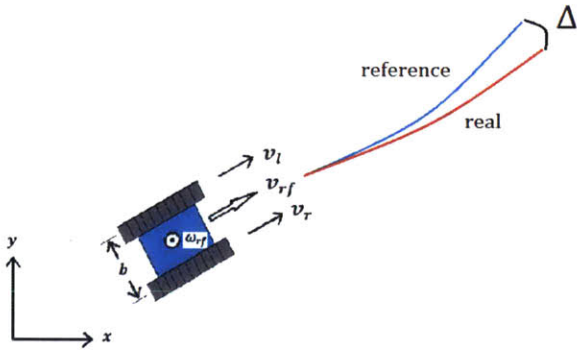


Figure 3.2. Diagram of trajectory deviation

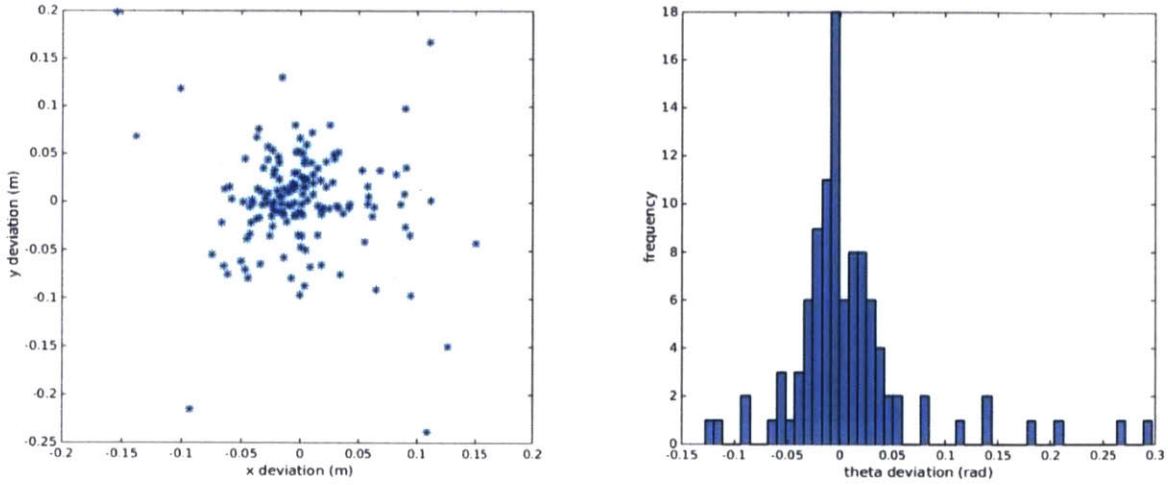


Figure 3.3. Distribution of deviation for 2 m trajectories; (x, y) position (left) and θ (right)

A similar plot can be obtained for longer piecewise trajectories (10 m). Figure 3.13 represents the distribution for 10 m piecewise trajectories.

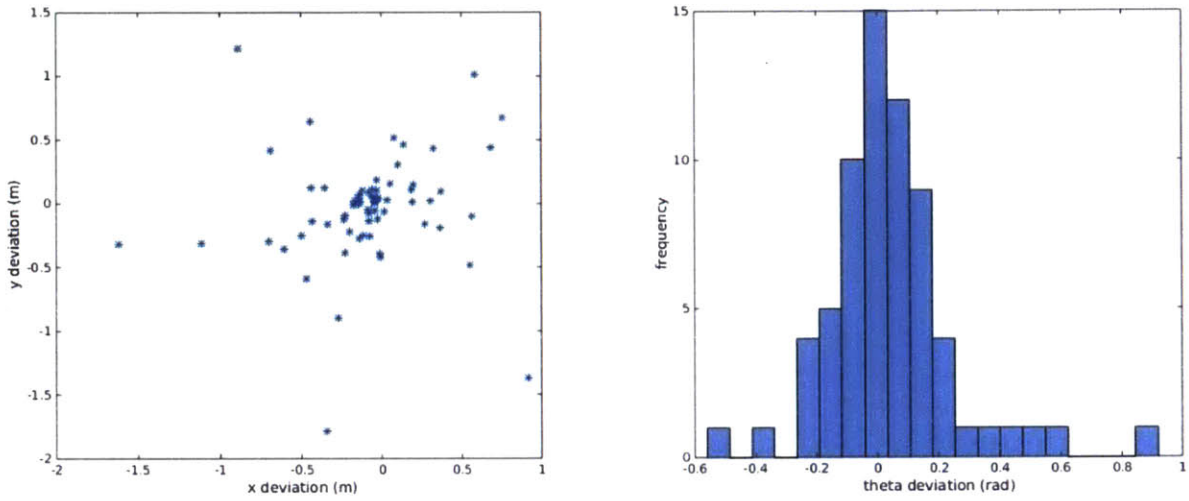


Figure 3.4. Distribution of deviation for 10 m trajectories; (x, y) position (left) and θ (right)

Using the experimental result of distribution of the deviations for 2 m piecewise trajectories from Figure 3.12, we can calculate the covariance matrix of process noise as described in Section 3.4. Again, the process noise is assumed to be zero mean white Gaussian noise. Then, we can use the calculated process noise to simulate the distribution of the vehicle state deviation plot for 10 m piecewise trajectories using the process described in Section 3.3.1. The mean and covariance matrix of the vehicle state can be calculated from equation (3.7) for each 10 m piecewise reference trajectory. The input to the vehicle is the odometer data that was used to reconstruct the reference trajectories. Then, for each piecewise trajectories, a point can be randomly sampled from the distribution with calculated mean and covariance matrix. The simulated distribution of deviation

can be compared with that of experimental data. Figure 3.14 shows the comparison. Table 3.3 represents the covariance matrices of both distribution plots.

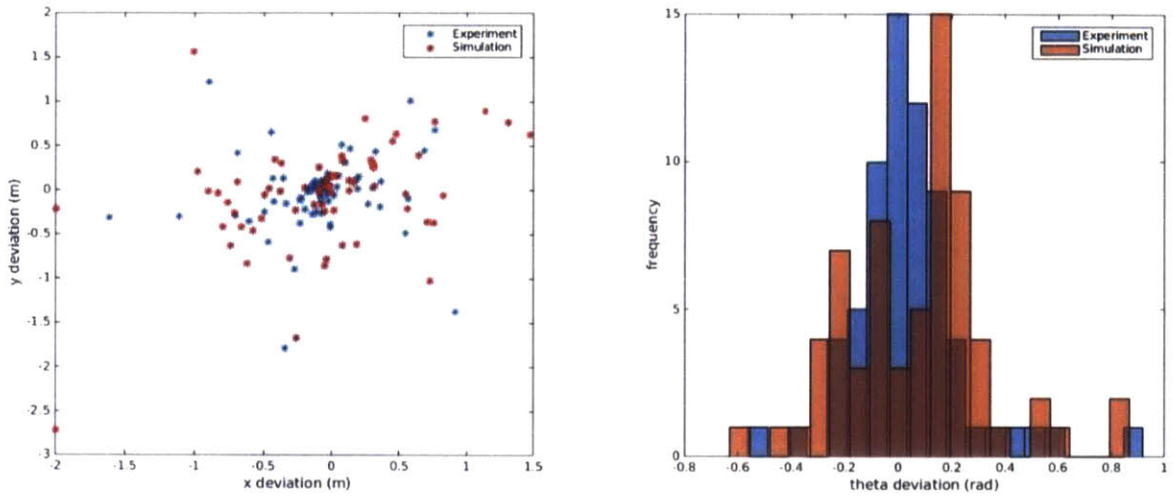


Figure 3.5. Comparison of simulated and experimental deviation; (x, y) position (left) and θ (right)

Table 3.3. Comparison of the covariance matrices of distribution plot in Figure 3.14.

	Experiment			Simulation		
Covariance matrices	0.1666	0.0154	-0.0158	0.2968	0.0208	-0.0311
	0.0154	0.1892	-0.0091	0.0208	0.2450	-0.0117
	-0.0157	-0.0091	0.0455	-0.0311	-0.0117	0.0619

Table 3.4. Correlation coefficients for the distribution plot in Figure 3.14.

	Experiment		Simulation	
Correlation coefficients	x and y	0.0867	x and y	0.0771
	x and θ	-0.1803	x and θ	-0.2293
	y and θ	-0.0981	y and θ	-0.0950

As it can be seen in both Figure 3.14 and Table 3.3, the simulated distribution matches the experimental distribution well. To be more specific, Table 3.4 shows the correlation coefficients of each coupled variables (x and y, x and θ, y and θ) for both experiment and simulation data. The coefficients are close to zero, meaning that the distribution of variables (x, y, θ) are almost independent. Assuming that they are independent, the p-values between experiment and simulation data are 0.6348 for x variable, 0.9789 for y variable, and 0.8486 for θ variable. They are calculated separately because we assumed the variables are independent. From the calculated p-values, we can safely state that experiment and simulation data are drawn from the same distribution. In fact, the simulated distribution is slightly more dispersed but this is tolerable since the simulated the process noise would make the algorithm slightly more conservative. This result validates our approach, even with a strong approximation such as linearization of the system dynamics. Since the distribution of the vehicle state can be predicted accurately, the solution obtained from the motion planner would actually be robust and also non-conservative. The consonance between simulated prediction and the experimental result is expected to be more accurate if we include the closed-loop controller. This is because the closed-loop controller would make the distribution of deviation be more clustered around the mean, adding more validity to the Gaussian assumption.

Chapter 4

Iterative Motion Planning Methodology Using Online Slip Estimation

As it was discussed previously, motion planning for autonomous vehicles in off-road condition is a complex problem due to slip phenomenon. A robust planning framework was presented in Chapter 3 to resolve this issue, but it requires prior knowledge about the environment that the vehicle is going to travel. To be more specific, it needs the information about the types of terrain the environment consists of, and their slip properties (mainly the effective wheel-base parameter α and the covariance matrix of the process noise term due to slip). When we don't have much information about the slip property of the environment, the robust planner in Chapter 3 is not suitable. Moreover, when the slip property changes due to changing terrain condition, information obtained a priori is inadequate. In such cases, online slip estimation would improve the result greatly.

The integrated prediction error minimization (IPEM) method is an effective algorithm [39] to estimate the slip property. For a given differential system, IPEM calculates the parameter (in our case, the effective wheel-base parameter α) that minimizes the deviation between prediction and measurement. In addition, it calculates the probabilistic distribution of the minimized deviation (the covariance matrix of the process noise term due to slip), which can be used in the motion planning step. The fact that it uses integrated prediction rather than derivative directly makes the algorithm more stable and accurate [39].

This chapter provides a robust motion planning methodology for tracked vehicles operating in a rough terrain environment with unknown slip property. An iterative motion planning scheme that uses online slip estimation is introduced. The environment is assumed to be flat and to consist of multiple soil types. The slip estimator calculates the slip property of the soil that the vehicle is currently operating on in an online fashion using IPEM algorithm. Then, the robust motion planner, presented in Chapter 3, plans the trajectory for the vehicle iteratively based on the most recent slip property calculated by the slip estimator. This chapter is based on the work of Lee et al. [29].

This chapter is organized as follows. Section 4.1 provides the problem statement and an overview of the iterative planning and estimation methodology. The vehicle system model is provided in Section 4.2. Basic information about the online slip estimator using IPEM approach is presented in Section 4.3. The simulation result for the operation of a tracked vehicle in a realistic scenario is presented in Section 4.4.

4.1. Problem Statement and Overall Methodology

The overall framework consists of three components: i) planner, ii) trajectory tracker, and iii) slip estimator. Given an initial state and slip estimate, the planner finds the optimal trajectory assuming that the entire environment has a homogeneous slip property. The obtained optimal trajectory is passed to the trajectory tracker. The tracker follows the trajectory and meanwhile gathers sensor measurements. For tracked vehicles, these would typically be GPS and odometer measurements. The slip estimator calculates the new slip estimate based on the measurements and feeds it back to the motion planner along with the new start state. The planner re-plans the optimal trajectory based on the new slip estimate, and the process repeats itself until the vehicle reaches the goal region. Figure 4.1 and Algorithm 4.1 show an overview of the entire iterative methodology.

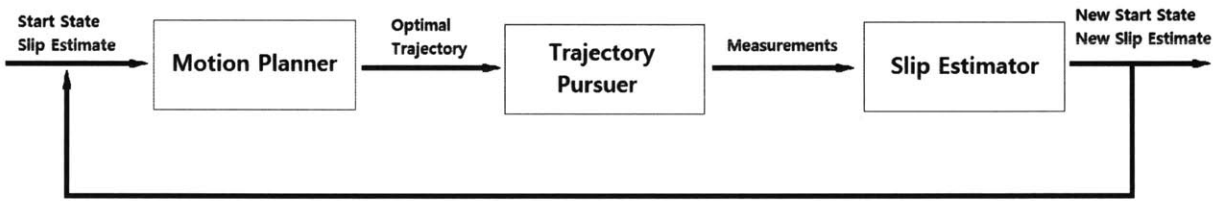


Figure 4.1. Diagram of overall framework [29]

Algorithm 4.1: Overall Framework [29]

1 Initialize starting state and slip estimate;
2 **while** (vehicle not in X_{goal})
3 MotionPlanner → Optimal trajectory;
4 TrajPursuit → Measurements;
5 SlipEstimator → Slip estimate;
6 New start state; New slip estimate;
7 **return**

Next, the problem statement for each part in Figure 4.1 is introduced. The problem statement for the motion planner is similar to the one in Section 3.1. Let the state space be $X \subset R^n$ and the input space be $U \subset R^m$. The vehicle model can be formulated in the following general form:

$$\dot{z} = f(z, u, p, w) \tag{4.1}$$

where $z \in X$ is the state vector, $u \in U$ is the (nominal) input vector, p is the parameter used to compensate for the slip, and w is the white process noise with zero mean. The noise term due to slip, w, has a different form from the system model introduced in equation (3.1), where w was included as an additive noise. In fact, this different modeling w doesn't affect much in our case, which is going be discussed in Section 4.2. p and Σ_w, the covariance matrix of w, vary with respect to the terrain type and they are calculated by the slip estimator. In this chapter, p and Σ_w together will be referred to as slip estimate. p and w are often considered together with input u to form the following equation:

$$\dot{z} = f(z, u_a(p, w)) \tag{4.2}$$

where u_a is the augmented input.

The vehicle should reach the goal region X_{goal}, starting from the initial state z_{init} and following the optimal and probabilistically feasible trajectory calculated by the motion planner. That is, the trajectory calculated by the motion planner should have the minimum cost with respect to the specified cost function. Also, letting X_{obs}, and $\tau : [0, 1] \rightarrow X$ again denote the obstacle space and trajectory respectively, $\mathrm{P}(\tau(s) \subset X_{obs}) < \delta, \forall t, (0 \leq s \leq 1)$ must be satisfied for the probabilistic feasibility.

The online slip estimator is essentially a system identification algorithm [32]. A system identification algorithm estimates the unknown parameters given system dynamics and actual

measurements. Let's assume the system dynamics is given as the equation (4.1). p and w can be thought of as the unknown parameter that needs to be estimated. Given state at $t = 0$, z_0, we can predict the state of the system along the trajectory, $z_{pred}(t)$, as the function of p and w using the equation (4.1). Here, the input u is assumed to be known. We are also given measurement along the trajectory, $z_{mea}(t)$. Then, the slip estimator calculates the best estimate for p and w that minimize the cost function of the form $J_{estimator}\big(z_{pred}(t), z_{mea}(t)\big)$, that is, the cost function that takes the predicted and measured trajectory as the arguments. In many cases, this is just the Euclidean norm.

$$\begin{gathered}\text{find the parameter estimate } p^* \text{ and } w^* \text{ subject to} \\ z_{pred}(0) = z_0 \\ z_{pred}(t), \text{the prediction of system state along the trajectory} \\ z_{mea}(t), \text{the measurement of system state along the trajectory} \\ \dot{z}_{pred} = f\big(z_{pred}, u, p, w\big), \text{with given } u \\ p^* \text{ and } w^* = \underset{p \text{ and } w}{\operatorname{argmin}} J_{estimator}\big(z_{pred}(t), z_{mea}(t)\big)\end{gathered} \tag{4.3}$$

Finally, the trajectory pursuer is a trajectory tracking system that follows a given trajectory. It is composed of sensor and closed-loop controller, which have the same dynamics as the ones used in motion planning step.

4.2. System Model

In this chapter, the following simple kinematic differential drive with augmented input is of interest. It is similar to vehicle model from Chapter 2 and 3 with slight modifications.

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}(\beta + w_\beta)\frac{v_r + v_l}{2}cos\theta\\ (\beta + w_\beta)\frac{v_r + v_l}{2}sin\theta\\ (\alpha + w_\alpha)\frac{v_r - v_l}{b}\end{bmatrix} = \begin{bmatrix}\frac{v_{r,a} + v_{l,a}}{2}cos\theta\\ \frac{v_{r,a} + v_{l,a}}{2}sin\theta\\ \frac{v_{r,a} - v_{l,a}}{b}\end{bmatrix} \tag{4.4}$$

where $z = [x, y, \theta]^T$ is the state vector, $u = [v_r, v_l]^T$ is the nominal input vector applied to the system, and $u_p = [v_{r,a}\ \ v_{l,a}]^T$ is the augmented input vector. $p = [\alpha\ \ \beta]^T$ is the parameter vector and $w = [w_\alpha\ \ w_\beta]^T$ is the process noise vector which takes a normal distribution, $w \sim N(0, \Sigma_w)$ where Σ_w is a constant matrix. u and u_a satisfy the following equation:

$$u_a = \begin{bmatrix}\frac{(\alpha + w_\alpha) + (\beta + w_\beta)}{2} & -\frac{(\alpha + w_\alpha) - (\beta + w_\beta)}{2}\\ -\frac{(\alpha + w_\alpha) - (\beta + w_\beta)}{2} & \frac{(\alpha + w_\alpha) + (\beta + w_\beta)}{2}\end{bmatrix} u = T_a u \tag{4.5}$$

The vehicle model can be considered as the effective wheel base model with an additional parameter β.

The cost function to be optimized for the motion planner is as follows:

$$J(X, u) = t_f + \int_0^{t_f} a(t)^T R a(t) dt \tag{4.6}$$

where t_f is the final time and R is the cost parameter matrix.

The system equation and the cost function needs to be linearized and discretized for the motion planner, which is similar to the case in Section 3.2. Here, we need to linearize the equation (4.4) with respect to the process noise term w too [34].

Now, let's explain the question imposed in Section 4.1. Why does modeling the process noise term w as in equation (4.1) not have much difference from the one suggested in equation (3.1)? This is thanks to the linearization step. If we linearize the equation (4.1) with respect to w, the resulting linearized equation has the process noise term as the additive noise, which has the same effect as the case in equation (3.1).

4.3. Online Slip Estimation

In this chapter, the IPEM method is used for the slip estimator [39]. It calculates the parameter estimate of the system that minimizes the deviation between the state prediction, $z_{pred}(t)$, and measurement, $z_{mea}(t)$. Additionally, it calculates the system's stochastic property, the distribution of the process noise assuming that it takes some normal distribution, $w \sim N(0, \Sigma_w)$.

First, let's briefly explain the IPEM algorithm. In IPEM method, $z_{pred}(t)$ is obtained by integrating the system equation from t_0 to t, using the current parameter estimate, p_{est}:

$$z_{pred}(t) = z_0 + \int_{t_0}^{t} f\left(z_{pred}(\tau), u_a(\tau, p_{est}, 0)\right) d\tau \tag{4.7}$$

where u_a is the augmented input as in Equation (17). Note 0 in $u_a(\tau, p_{est}, 0)$, since the process noise has zero mean.

Let us put $z_{pred}(t) = g(z(t_0), u_a(\cdot, p_{est}, 0))$. This equation is usually a nonlinear equation for vehicle models. The above equation can be linearized as follows.

$$z_{pred}(t) \approx \left.\frac{\partial g}{\partial p}\right|_{SE} p + \cdots \tag{4.8}$$

We will use the notation SE, which stands for slip estimate, to denote $p = p_{est}, w = 0$. Now, we can approximate the new parameter estimate based on measurements received using the following equation:

$$\left.\frac{\partial g}{\partial p}\right|_{SE} \Delta p \approx z_{mea} - z_{pred} \tag{4.9}$$

However, obtaining the Jacobian matrix in Equation (4.9), which can be obtained as applying the difference equation $\frac{\partial g}{\partial p_i} \approx \left.\frac{g(z_{pred}(t_0),u_a(\cdot,p+\delta p_i,w))-g(z_{pred}(t_0),u_a(\cdot,p,w))}{\varepsilon}\right|_{SE}$ for all matrix elements, is computationally demanding. We take the detour and first linearize the differential system equation.

$$\begin{aligned}\delta \dot{z}_{pred}(t) &\approx \left.\frac{\partial f}{\partial z_{pred}}\right|_{SE} \delta z_{pred}(t) + \left.\frac{\partial f}{\partial u_a}\right|_{SE} \delta u_a(t,p,w)|_{SE} \\ &= F\delta z_{est}(t) + G\delta u_a(t,p,w)|_{SE}\end{aligned} \tag{4.10}$$

For this linearized differential system equation, the solution is the following vector convolution integral:

$$\delta z_{pred}(t) = \Phi(t,t_0)\delta z(t_0) + \int_{t_0}^{t} \Gamma(t,\tau)\delta u_a(t,p,w)|_{SE} d\tau \tag{4.11}$$

Here, $\Phi(t,\tau)$ and $\Gamma(t,\tau)$ can be obtained in the following way.

$$\begin{aligned}\dot{\Phi}(t,\tau) &= F(t)\Phi(t,\tau) \\ \Psi(t,\tau) &= \int_{\tau}^{t} F(\zeta)d\zeta\end{aligned} \tag{4.12}$$

$$\Phi(t,\tau) = e^{\Psi(t,\tau)}$$

$$\Gamma(t,\tau) = \Phi(t,\tau)G(\tau)$$

We can approximate the Jacobian matrix $\left.\frac{\partial g}{\partial p}\right|_{SE}$ as follows:

$$\left.\frac{\partial g}{\partial p}\right|_{SE} \approx H_{sys} = \int_{t_0}^{t} \Gamma(t,\tau) \left.\frac{du_a(\tau,p,w)}{dp}\right|_{SE} d\tau \tag{4.13}$$

In addition, we need the measurement covariance matrix, the covariance matrix of $r(t) = z_{mea}(t) - z_{pred}(t)$.

$$R_{sys} = \Phi(t,t_0)\Sigma_{x,mea}(t_0)\Phi(t,t_0)^T + \int_{t_0}^{t} \Gamma(t,\tau)Q(\tau)\Gamma(t,\tau)^T d\tau + \Sigma_{x,mea}(t) \tag{4.14}$$

where $\Sigma_{x,mea}(t)$ is the covariance matrix of $z_{mea}(t)$ (it is easy to think of it as the GPS error covariance matrix) and $Q(\tau)$ is the covariance matrix of $\delta u_a(\tau,p,w)$.

We can obtain the parameter value using the following Kalman filter framework iteratively:

$$p_{est,k|k-1} = p_{est,k-1|k-1}$$

$$\Sigma_{p_{est},k|k-1} = \Sigma_{p_{est},k-1|k-1} + \Delta_k$$

$$r_k = z_{pred,k} - z_{mea,k}$$

$$S_k = H_{sys,k}\Sigma_{p_{est},k|k-1}{H_{sys,k}}^T + R_{sys,k} \tag{4.15}$$

$$K_{e,k} = \Sigma_{p_{est},k|k-1}{H_{sys,k}}^T {S_k}^{-1}$$

$$p_{est,k|k} = p_{est,k|k-1} + K_{e,k}r_k$$

$$\Sigma_{p_{est},k|k} = (I - K_{e,k}H_{sys,k})\Sigma_{p_{est},k|k-1}$$

where k stands for k^{th} iteration. $\Sigma_{p_{est}}$ is the covariance matrix of the parameter estimate and Δ_k is the covariance matrix that governs how much weight we put on the most recent measurement. The larger that Δ_k is, the more weight we put on the recent measurement.

We can also estimate for Σ_w, assuming that it is a constant, although it can vary according to terrain type. The overall procedure is similar to the above explanation. More detailed explanation can be found in [39].

Next, let us apply the above IPEM method to our vehicle model presented in Section 4.2. We need the Jacobian matrices $\left.\frac{\partial f}{\partial z_{pred}}\right|_{SE}$ and $\left.\frac{\partial f}{\partial u_a}\right|_{SE}$.

$$F = \left.\frac{\partial f}{\partial z_{pred}}\right|_{SE} = \begin{bmatrix} 0 & 0 & -\frac{v_{r,a}+v_{l,a}}{2}sin\theta_{pred} \\ 0 & 0 & \frac{v_{r,a}+v_{l,a}}{2}cos\theta_{pred} \\ 0 & 0 & 0 \end{bmatrix}_{SE} = \begin{bmatrix} 0 & 0 & -\dot{y}_{pred} \\ 0 & 0 & \dot{x}_{pred} \\ 0 & 0 & 0 \end{bmatrix}$$

$$G = \left.\frac{\partial f}{\partial u_a}\right|_{SE} = \begin{bmatrix} \frac{1}{2}cos\theta_{pred} & \frac{1}{2}cos\theta_{pred} \\ \frac{1}{2}sin\theta_{pred} & \frac{1}{2}sin\theta_{pred} \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix} \tag{4.16}$$

We can calculate $\Phi(t,\tau)$ easily:

$$\Phi(t,\tau) = I + \Psi(t,\tau) = \begin{bmatrix} 1 & 0 & -\Delta y_{pred} \\ 0 & 1 & \Delta x_{pred} \\ 0 & 0 & 1 \end{bmatrix} \tag{4.17}$$

where $\Delta x_{pred} = x_{pred}(t) - x_{pred}(\tau)$. Using the above equations,

$$\Gamma(t,\tau) = \begin{bmatrix} \frac{1}{2}cos\theta_{pred} - \frac{\Delta y_{pred}}{b} & \frac{1}{2}cos\theta_{pred} + \frac{\Delta y_{pred}}{b} \\ \frac{1}{2}sin\theta_{pred} + \frac{\Delta x_{pred}}{b} & \frac{1}{2}sin\theta_{pred} - \frac{\Delta x_{pred}}{b} \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix} \tag{4.18}$$

Thus,

$$\Gamma(t,\tau)\frac{du_a(\tau,p,w)}{dp}\Big|_{SE} = \begin{bmatrix} -\frac{\Delta y_{pred}}{b}(v_r - v_l) & \frac{1}{2}(v_r + v_l)cos\theta_{pred} \\ \frac{\Delta x_{pred}}{b}(v_r - v_l) & \frac{1}{2}(v_r + v_l)sin\theta_{pred} \\ \frac{(v_r - v_l)}{b} & 0 \end{bmatrix} \tag{4.19}$$

We can obtain $H_{sys} = \int_{t_0}^{t} \Gamma(t,\tau)\frac{du_a(\tau,p,w)}{dp}\Big|_{SE} d\tau$ by integrating Equation (24). Since we assume piecewise constant input, that is, $u = [v_r \;\; v_l]^T$ is piecewise constant, the calculation is simplified.

When we obtain R_{sys}, we use the assumption that Σ_w is a constant matrix. The following holds in that case:

$$Q(\tau) = T_u \Sigma_w {T_u}^T \tag{4.20}$$

where $T_u = \begin{bmatrix} \frac{v_r - v_l}{2} & \frac{v_r + v_l}{2} \\ -\frac{v_r - v_l}{2} & \frac{v_r + v_l}{2} \end{bmatrix}$.

4.4. Simulation Results

Simulations were conducted in an artificially created environment with several properties. Figure 4.2 shows the simulated map. There are numerous polygonal obstacles (black polygons) such as sand dunes, ponds, buildings, etc. In addition, the environment is assumed to consist of two terrain

types, represented by the white and green (gray) regions in Figure 4.2. The white region represents a more slippery and unstable terrain (such as an icy region), whereas the green region represents a less slippery and more stable terrain (such as turf). The map size is 200 by 100 meters. The vehicle's starting pose is (180 m, 85 m, π rad) and its goal region is the 10 m × 10 m box around the point (20 m, 20 m).

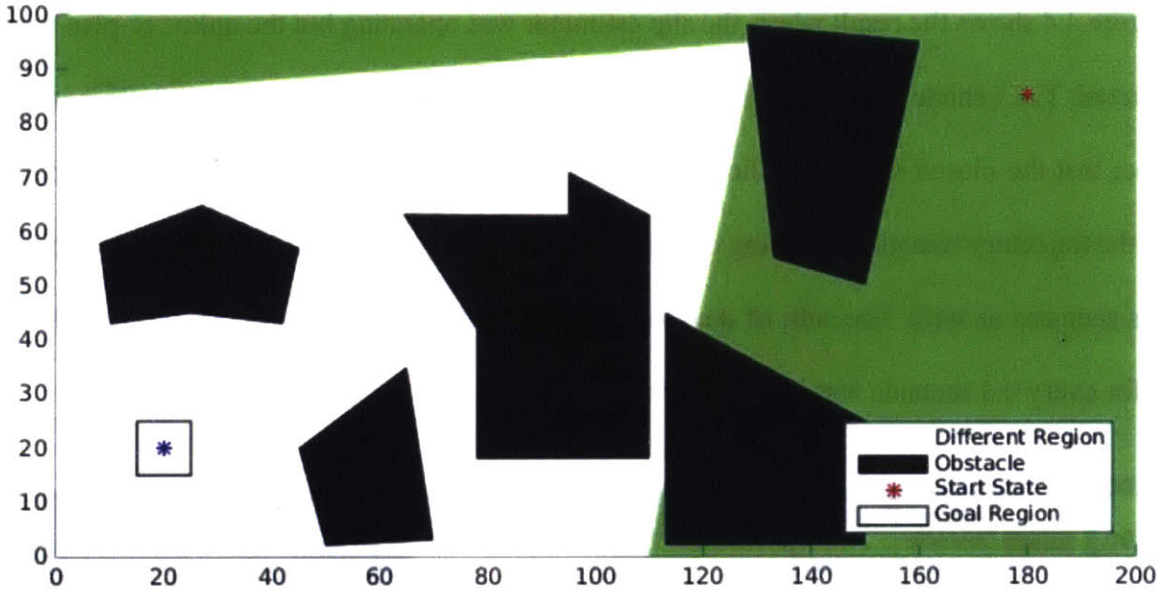


Figure 4.2. The map scenario for simulation [29]

The vehicle's trajectory pursuit is also simulated using a closed loop controller. The same controller that was used for the motion planner must be used, which is shown in equation (3.16). The vehicle slip is also simulated during the pursuit. The slip was simulated so that each terrain has different true p and Σ_w values. The slip estimator would have to estimate the values for p and Σ_w.

Figure 4.3 shows the simulation result when the slip estimator was not operating. The optimal trajectory was planned using initial guesses of p and Σ_w. Then, the vehicle tried to follow the trajectory assuming that the initial guesses were correct. The guesses were chosen so that they were

close to true p and Σ_w of the green region, and the vehicle thus followed the trajectory well within that region. However, when the terrain changed, the vehicle deviated much from the planned trajectory and collided with an obstacle. The sum of deviation between reference trajectory and the vehicle's real state for every 0.3 seconds was 53.7946m in green region and 869.2009m in white region.

Figure 4.4 shows the result where the slip estimator was operating but the online re-plan was not conducted. The vehicle follows the trajectory much better than it did in Figure 4.3. This is due to the fact that the closed loop controller could utilize the newly calculated slip estimate. However, since the trajectory was planned using only the initial guesses, the vehicle collided with an obstacle in this scenario as well. The sum of deviation between reference trajectory and the vehicle's real state for every 0.3 seconds was 69.8949m in green region and 601.3396m in white region.

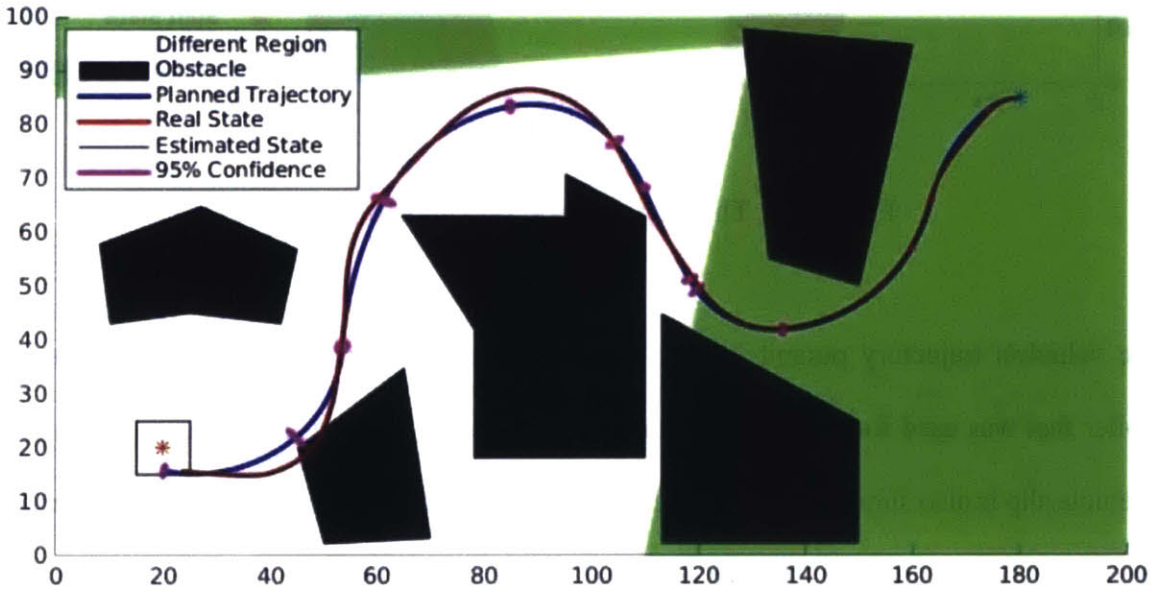


Figure 4.3. The simulation without slip estimation [29]

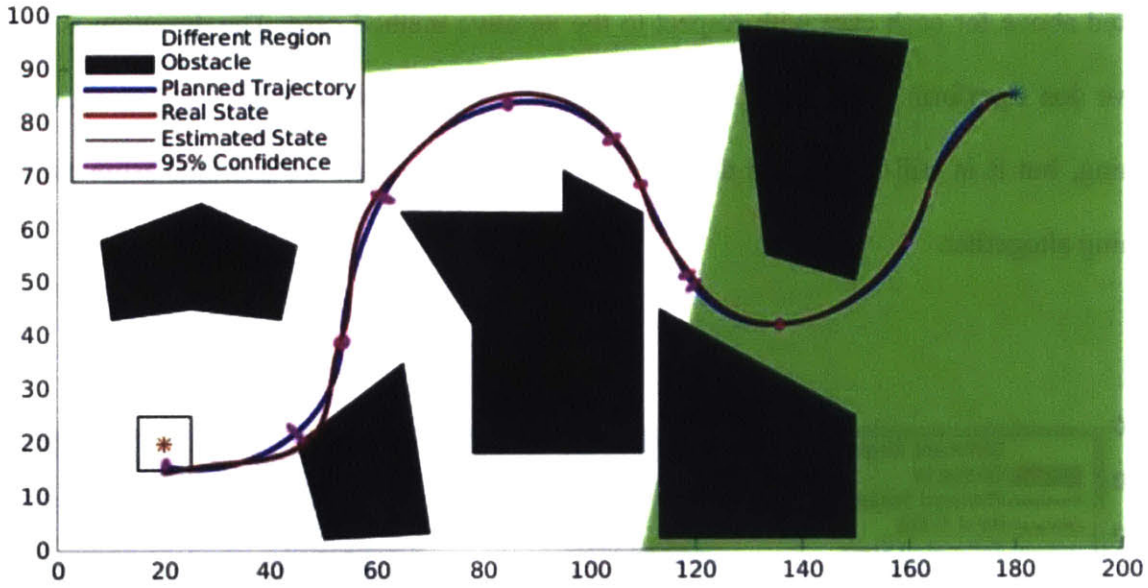


Figure 4.4. The simulation with slip estimation, but without re-planning [29]

Figure 4.5 shows the result with the entire framework operating. It shows re-planned trajectories for all iterations. In this case, the vehicle not only followed the planned trajectory successfully, but it also avoided all the obstacles. This indicates that the re-planning plays a crucial role in the overall vehicle operating scheme. The sum of deviation between reference trajectory and the vehicle's real state for every 0.3 seconds was 71.8236m in green region and 90.2926m in white region. This means the vehicle followed the reference trajectory well even in white region, unlike the cases in Figure 4.3 and 4.4. Figure 4.6 shows the calculated parameter values (p) for the Figure 4.4 case (left) and the Figure 4.5 case (right). The true parameters used were $p_1 = [0.9\ 1.0]^T$ for the first region (white) and $p_2 = [0.5\ 0.8]^T$ for the second (green) region, which are indicated as dashed horizontal lines. The true parameters were calculated correctly and the terrain change was also successfully detected. The terrain change is indicated with vertical lines. The high frequency noise is due to the assumed measurement noise and Σ_w. Table 4.1 shows the ratio of sums of deviation

calculated above for each case with respect to the iterative method case. The deviation is largest when we don't perform estimation. The deviation shrinks when we perform estimation without replanning, but it is still large. The deviation is the smallest when we perform estimation and replanning altogether.

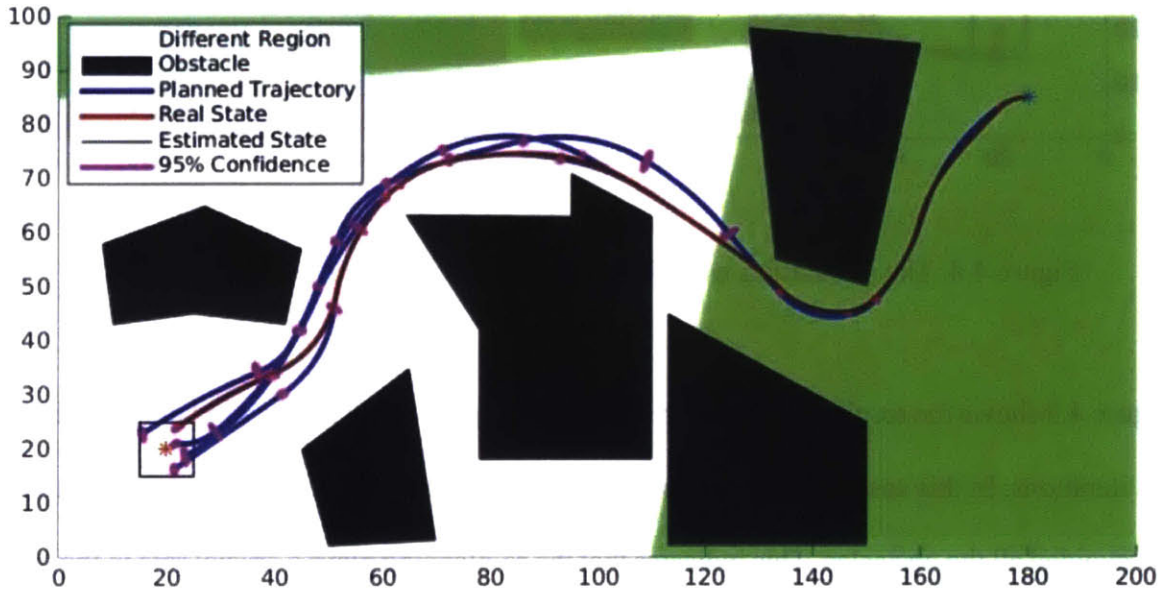


Figure 4.5. The simulation with slip estimation and re-planning [29]

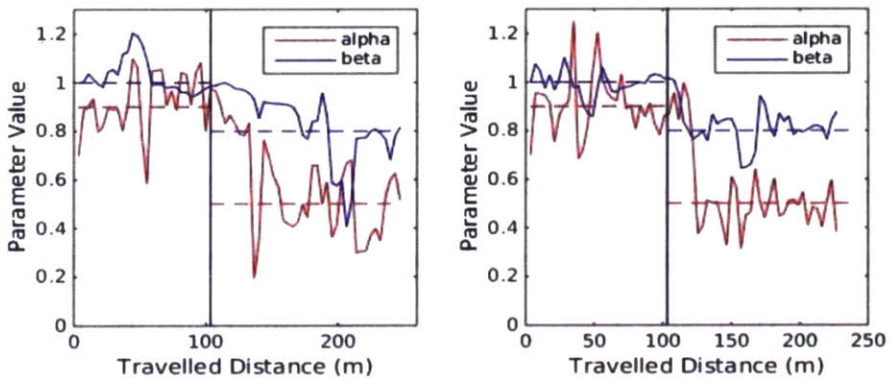


Figure 4.6. The parameter estimate for Figure 4.4 (left) and Figure 4.5 (right) [29]

Table 4.1. Ratio of sum of vehicle's deviation (with respect to the iterative method) [29]

Scenario	No estimation	No re-plan	Iterative method
Ratio of Sum of Dev.	5.6934	4.1405	1

There is a slight delay in terrain change detection, as we can observe from Figure 4.6. That is, there is a delay for the parameter alpha to converge to the true value after the terrain change. The reason for this observation is that the parameter alpha converges to the true value only when it meets curved trajectory. It doesn't converge on the straight line trajectory. This is due to the fact that the parameter alpha is only related to turning rate of the vehicle as in equation (4.4). This can be seen well from the following result.

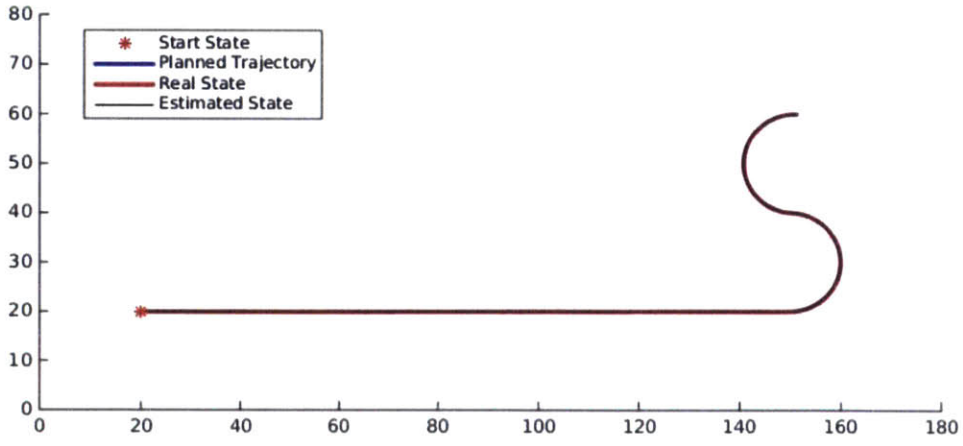


Figure 4.7. Pursuit of the specified reference trajectory consists of long straight line followed by two half circles

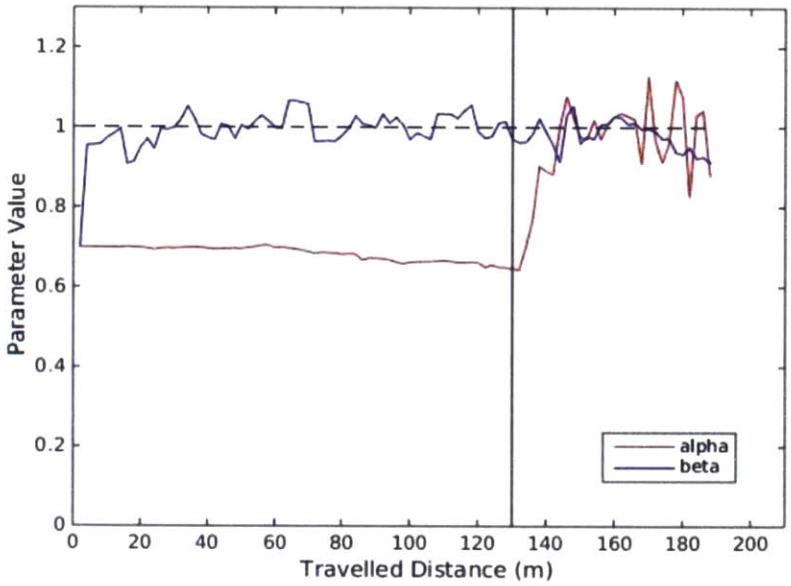


Figure 4.8. Parameter estimate for Figure 4.7

In Figure 4.7, the vehicle followed the specified reference trajectory, which consists of long straight line followed by two half circles. Figure 4.8 shows the parameter estimate along the trajectory. The true parameters, $p = [1.0\ 1.0]^T$, are indicated as the dashed horizontal line. The change from straight line to half circle is indicated with vertical line. The true value of parameter alpha is not detected until the vehicle enters circular trajectory as explained.

Chapter 5

Conclusion

5.1. Results and Contributions

This thesis presents a robust motion planning architecture for autonomous tracked vehicles operating on off-road condition. The slip of the vehicle due to vehicle-terrain interaction becomes a major source of uncertainty. In order to compensate for the effect of slip during motion planning phase, a robust motion planning strategy is introduced.

Before actively discussing robustness, a computationally efficient optimal motion planning method for non-holonomic system, such as tracked vehicles, is presented in Chapter 2. Instead of trying to achieve the exact connection for non-holonomic steering function, which is computationally expensive, an approximate method with "repropagation" step can be used to meet computational efficiency [15].

In Chapter 3, a robust motion planning scheme is introduced by combining two state-of-the-art algorithms, CC-RRT* [35] and LQG-MP [4]. CC-RRT* calculates the probability of collision and

ensures robustness in quantitative manner. LQG-MP propagates uncertainty in closed-loop fashion. The combined algorithm yields robust yet non-conservative trajectory.

The algorithm is further improved by introducing the online slip estimator in Chapter 4. The robust motion planning algorithm in Chapter 3 requires much prior knowledge about the environment. This issue can be resolved by combining the robust motion planner with online slip estimator and re-planning the trajectory iteratively. The online slip estimation plays a crucial role when there is not much information about the environment or when there is an abrupt change in terrain condition.

To the author's knowledge, this work is the first to demonstrate a practical methodology for robust optimal motion planning of autonomous tracked vehicles operating in rough outdoor scenarios. The overall algorithm has been tested on realistic simulated environment. An experimental result is also provided for the partial validation of the algorithm. The work in this thesis has many practical applications. It can be applied to exploration problems, such as planetary rovers, where detecting and estimating different terrains in unknown environments and avoiding obstacles at the same time is the major goal. In addition, though it is only applied to tracked vehicles, the methodology is versatile, which is another advantage of this work. It can be applied to many other robotic systems that requires robust motion planning.

5.2. Future Work

There might be several extensions to improve this research:

The current work is aimed for robust motion planning of tracked vehicle on flat deformable terrain. This scenario can be extended to a more complex one, where the vehicle is traveling on 3-

D terrain where the slope of the terrain cannot be ignored. In this case, an accurate way to model and quantify the slip uncertainty should be developed.

More experimental data can be provided to validate the result. The experimental data presented in the thesis is only for concrete surface. An experiment on several other terrain types can be provided to enrich the result. In addition, the motion planning algorithm can be tested with an actual tracked vehicle operating in environment where it is composed of multiple soil types and the full information about the environment is not known.

The computational efficiency of the algorithm can be improved. For example, a more smart random node sampling and probabilistic feasibility checking for CC-RRT* can improve the computational efficiency.

Bibliography

[1] A. Angelova, L. Matthies, D. Helmick, and P. Perona, "Learning and Prediction of Slip from Visual Information," *Journal of Field Robotics*, Vol. 24, Issue 3, March, 2007, pp. 205-231.

[2] A. Angelova, L. Matthies, D. Helmick, G. Sibley, and P. Perona, "Learning to Predict Slip for Ground Robots," *IEEE International Conference on Robotics and Automation (ICRA)*, May, 2006, Orlando, Florida, U.S.A., pp. 3324-3331.

[3] J. Barraquand and J. C. Latombe, "Robot Motion Planning: A Distributed Representation Approach," *International Journal of Robotics Research*, Vol. 10, No. 6, 1991, pp. 628-649.

[4] J. van den Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized Path Planning for Robots with Motion Uncertainty and Imperfect State Information," *International Journal of Robotics Research*, Vol. 30 No. 7, June, 2011, pp. 895-913.

[5] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, Athena Scientific, 1995, Boston, MA.

[6] A. E. Bryson Jr. and Y. C. Ho, *Applied Optimal Control: Optimization, Estimation, and Control*, CRC Press, 1975.

[7] B. Chazelle, "Approximation and decomposition of shapes," In J. T. Schwartz and C. K. Yap, editors, *Algorithmic and Geometric Aspects of Robotics*, Lawrence Erlbaum Associates, 1987, pp. 145–185, Hillsdale, NJ.

[8] M. Farber, S. Tabachnikoz, and S. Yuzvinsky, "Topological Robotics: Motion Planning in Projective Spaces," *International Mathematics Research Notices*, 2003, pp. 1853-1870.

[9] J. R. Fink and E. A. Stump, "Experimental Analysis of Models for Trajectory Generation on Tracked Vehicles," *IEEE International Conference on Intelligent Robotics and Systems (IROS)*, September 2014, pp. 1970-1977.

[10] G. F. Frankin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 7th Edition, Pearson, 2015.

[11] R. Gonzalez, F. Rodriguez, and J. L. Guzman, *Autonomous Tracked Robots in Planar Off-road Conditions – Modelling, Localization, and Motion Control*, International Publishing Switzerland, Springer, 2014.

[12] L. J. Guibas, D. Hsu, H. Kurniawati, and E. Rehman, "Bounded uncertainty roadmaps for path planning," *Algorithmic Foundation of Robotics VIII*, Springer Berlin Heidelberg, 2009, pp. 199-215.

[13] J. S. Ha, J. J. Lee, and H. L. Choi, "A Successive Approximation-Based Approach for Optimal Kinodynamic Motion Planning with Nonlinear Differential Constraints," *IEEE Conference on Decision and Control (CDC)*, December 2013, Florence, Italy, pp. 3623-3628.

[14] D. Helmick, A. Angelova, and L. Matthies, "Terrain Adaptive Navigation for Planetary Rovers," *Journal of Field Robotics*, Vol. 26, Issue 4, April, 2009, pp. 391-410.

[15] J. H. Jeon, S. Karaman, and E. Frazzoli, "Anytime Computation of Time-Optimal Off-road Vehicle Maneuvers using the RRT*," *IEEE Conference on Decision and Control (CDC)*, December 2011, Orlando, FL, USA, pp. 3276-3282.

[16] Y. Kanayama, Y. Kimura, F. Miyazaki, T. Noguchi, "A Stable Tracking Control Method for an Autonomous Mobile Robot," *IEEE International Conference on Robotics and Automation (ICRA)*, May 1990, Cincinnati, OH, USA, pp. 384-389.

[17] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *International Journal of Robotics Research*, Vol. 30 No. 7, June, 2011, pp. 846-894.

[18] S. Karaman and E. Frazzoli, "Optimal Kinodynamic Motion Planning using Incremental Sampling-based Methods," *IEEE Conference on Decision and Control (CDC)*, December 2010, Atlanta, GA, USA, pp. 7681-7687.

[19] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime Motion Planning Using the RRT*," *IEEE International Conference on Robotics and Automation (ICRA)*, May, 2011, Shanghai, China, pp. 1478-1483.

[20] L. E. Kavraki, J. C. Latombe, R. Motwani, and P. Raghavan, "Randomized Query Processing in Robot Path Planning," *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, ACM, 1995.

[21] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics & Automation*, Vol. 12, No. 4, June 1996, pp. 566–580.

[22] A. Kelly, "Linearized Error Propagation in Odometry," *International Journal of Robotics Research*, Vol. 23, No. 2, February, 2004, pp. 179-218.

[23] D. E. Kirk, *Optimal Control Theory An Introduction*, Dover Publications, Inc., 1970, Mineola, NY.

[24] J.-C. Latombe, *Robot Motion Planning*, Kluwer, 1991, Boston, MA.

[25] J. P. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in Nonholonomic Motion Planning for Mobile Robots," *Robot Motion Planning and Control*, Springer, 1998, pp. 1-53, Berlin Heidelberg.

[26] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006.

[27] S. M. LaValle and J. J. Kuffner. "Rapidly-exploring random trees: Progress and prospects," In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, A K Peters, 2001, pp. 293–308, Wellesley, MA.

[28] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *IEEE International Conference on Robotics and Automation (ICRA)*, 1999, pp. 473–479.

[29] S. U. Lee and K. Iagnemma, "Robust Motion Planning Methodology for Autonomous Tracked Vehicles in Rough Environment Using Online Slip Estimation," *IEEE International Conference on Intelligent Robots and Systems (IROS)*, October, 2016, Daejeon, Korea, Accepted.

[30] S. U. Lee, R. Gonzalez, and K. Iagnemma, "Robust Sampling-based Motion Planning for Autonomous Tracked Vehicles in Deformable High Slip Terrain," *IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, Stockholm, Sweden, pp. 2569-2574.

[31] J. Leonard, J. P. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, 0. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. Antone, R. Galejs, S. Krishnamurthy, and J. Williams, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, Vol. 25, No. 10, October, 2008, pp. 727-774.

[32] L. Ljung, *System Identification: Theory for User*, 2nd Edition, Prentice Hall, 1999.

[33] T. Lozano-perez, "Spatial Planning: A Configuration Space Approach," *IEEE Transaction on Computers*, Vol. 100, No. 2, 1983, pp. 108-120.

[34] B. D. Luders and J. P. How, "Probabilistic Feasibility for Nonlinear Systems with Non-Gaussian Uncertainty using RRT," *AIAA Infotech @ Aerospace Conference*, March, 2011, St. Louis, MO, USA.

[35] B. D. Luders, S. K. Karaman, and J. P. How, "Robust Sampling-based Motion Planning with Asymptotic Optimality Guarantees," *AIAA Guidance, Navigation, and Control Conference (GNC)*, August 2013, Boston, MA, USA.

[36] B. D. Luders, M. Kothari, and J. P. How, "Chance Constrained RRT for Probabilistic Robustness to Environmental Uncertainty," *AIAA Guidance, Navigation, and Control Conference (GNC)*, August 2010, Toronto, Ontario, Canada.

[37] J. H. Reif, "Complexity of the mover's problem and generalizations," *IEEE Symposium on Foundations of Computer Science*, 1979, pp. 421–427.

[38] J. T. Schwartz and M. Sharir, "On the "Piano movers'" problem: 1. The Case of a Two-dimensional Rigid Polygonal Body Moving Amidst Polygonal Barriers," *Communications on Pure and Applied Mathematics*, Vol. 36, No. 3, 1983, pp. 345-398.

[39] N. Seegmiller, F. Rogers-Marcovitz, G. Miller, and A. Kelly, "Vehicle Model Idenfitication by Integrated Prediction Error Minimization," *International Journal of Robotics Research*, Vol. 32, No. 8, July, 2013, pp. 912-931.

[40] R. F. Stengel, *Optimal Control and Estimation*, Courier Corporation, 2012.

[41] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk,

E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, Vol. 23, No. 9, September, 2006, pp. 661-692.

[42] A. Tuan Le, D. C. Rye, and H. F. Durrant-Whyte, "Estimation of Track-soil Interaction for Autonomous Tracked Vehicles," *IEEE International Conference on Robotics and Automation (ICRA)*, April, 1997, Albuquerque, New Mexico, U.S.A., pp. 1388-1393.

[43] D. J. Webb and J. van den Berg, "Kinodynamic RRT* : Asymptotically Optimal Motion Planning for Robots with Linear Dynamics," *IEEE International Conference on Robotics and Automation (ICRA)*, May 2013, Karlsruhe, Germany, pp. 5054-5061.

[44] J. Y. Wong, *Theory of Ground Vehicles*, 4th Edition, Wiley, 2008.

[45] B. Yamauchi, "Packbot: A versatile platform for military robotics," in SPIE, 2004, pp. 228-237.

[46] W. Yu, O. Y. Chuy, E. G. Collins, and P. Hollis, "Analysis and Experimental Verification for Dynamic Modeling of a Skid-steered Wheeled Vehicle," *IEEE Transactions on Robotics*, Vol. 26, No. 2, April, 2010, pp. 4212-4219.