FAST PATH PLANNING IN UNCERTAIN ENVIRONMENTS: THEORY AND EXPERIMENTS

BIN XU

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

Doctor of Philosophy
in
Electrical Engineering

Andrew J. Kurdila, Co-Chair
Daniel J. Stilwell, Co-Chair
Douglas K. Lindner
Craig A. Woolsey
Christopher L. Wyatt

Blacksburg, Virginia
November 19th, 2009

Keywords: Autonomous Vehicle Navigation, Path Planning, Receding Horizon Control and Level Set Method

Fast Path Planning in Uncertain Environments: Theory and Experiments

Bin Xu

ABSTRACT

This dissertation addresses path planning for an autonomous vehicle navigating in a two dimensional environment for which an *a priori* map is inaccurate and for which the environment is sensed in real-time. For this class of application, planning decisions must be made in real-time. This work is motivated by the need for fast autonomous vehicles that require planning algorithms to operate as quickly as possible.

In this dissertation, we first study the case in which there are only static obstacles in the environment. We propose a hybrid receding horizon control path planning algorithm that is based on level-set methods. The hybrid method uses global or local level sets in the formulation of the receding horizon control problem. The decision to select a new level set is made based on certain matching conditions that guarantee the optimality of the path. We rigorously prove sufficient conditions that guarantee that the vehicle will converge to the goal as long as a path to the goal exists. We then extend the proposed receding horizon formulation to the case when the environment possesses moving obstacles. Since all of the results in this dissertation are based on level-set methods, we rigorously investigate how level sets change in response to new information locally sensed by a vehicle. The result is a dynamic fast marching algorithm that usually requires significantly less computation that would otherwise be the case. We demonstrate the proposed dynamic fast marching method in a successful field trial for which an autonomous surface vehicle navigated four kilometers through a riverine environment.

To my parents and grandparents,

for their love and support.

Acknowledgement

I owe a great debt of gratitude to many people who have helped me and made the research work in this dissertation possible.

First and foremost, I would like to thank my advisors, Dr. Daniel Stilwell and Dr. Andrew Kurdila, for their guidance, inspiration, and encouragement over the years. Their continuous support, considerable help, and patience have been of outmost importance for finishing my study. I appreciate the efforts of the other three members of my committee, Dr. Douglas Lindner, Dr. Craig Woolsey, and Dr. Christopher Wyatt, for their time reviewing the dissertation and providing me with insightful comments.

I would like to recognize the contributions of my colleagues, in particular, Xiaojin Gong, Darren Maczka and Caleb Reed. Without their support and assistance, the work would have been impossible to complete. I am grateful to the other members of the Autonomous Systems and Controls Lab (ASCL) at Virginia Tech. We are a great team and it has been a great pleasure knowing everyone and working with them over all these years.

Furthermore, I would like to thank my parents and my grandparents for their love and support. It is their encouragement that keeps me going forward. I would like to express my deepest gratitude to Sunkyung Kim, whose patience with and faith in me made the completion of this project possible.

I would like to express my deepest appreciation to Dr. Stilwell and other faculty and staff at Virginia Tech for their selfless assistance after Hurricane Katrina in 2005.

The work leading to this dissertation was supported by the Army Research Office, the Office of Naval Research, and the National Science Foundation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Statement of Goals

In this dissertation, we consider an autonomous vehicle navigating towards a predefined goal. The target location is situated in a two dimensional dynamic environment and both moving and static obstacles are present. We assume that an *a priori* map is available but inaccurate. Due to the inconsistency between the *a priori* map and the actual environment, fast path replanning is required in order to avoid collisions with obstacles not identified in the *a prior* map. Specifically, we aim to devise a control algorithm for path planning

(1) that alters the path as quickly as possible in the presence of newly detected obstacles,

(2) that generates a minimum risk path given the knowledge of the environment, and

(3) for which sufficient conditions can be derived that guarantee that the vehicle reaches the goal.

In this dissertation, the proposed methods do not account for detailed vehicle dynamics and do not address path-following limitations. Thus, intuitively speaking, our approach is

well-suited to vehicles that can follow prescribed paths to a high degree of accuracy. This class of vehicle includes certain autonomous surface marine vehicles, which motivates our work, but also includes classes of ground vehicles and ground hovercrafts. In this dissertation, it is assumed that obstacles in the environment are detected by on-board sensors that have a limited range. For example, in [21] and [22], we reported on a successful vision algorithm which detects navigation hazards for an autonomous surface vehicle with a maximum range of forty to fifty meters.

1.2 Related Work

Path planning for autonomous vehicles has been studied for decades. Excellent references that survey the current literature can be found in [26], [44], and [46]. These methods can be grouped into two categories: local and global replanning. These local planning methods including [7], [33], [39], [43], and [59], among many others are fast, but they often fail in some trap scenarios for which progress toward the desired endpoint is impossible due to the lack of global knowledge about the environment. Global replanning can avoid these problems but can be computationally expensive. In [13], [14], [38], [68] and [69], a group of global replanning methods are introduced. These methods share some common attributes. They are variants of A* search (see e.g. [44], pp.604). The map is modeled by a set of nodes and the traversal cost between two adjacent nodes. Thus, finding an optimal path is treated as a minimal cost path searching problem in graph [74]. When the environment changes, the costs for traversing the corresponding nodes will change. The overall minimal cost and the path to travel from a given node to the goal are consistently updated.

1.3 Contribution

In this dissertation, we propose several path planning strategies that either locally or globally compute the minimal risk path in the presence of both static and moving obstacles. The following list summarizes the primary contributions of this work:

(a) For planning paths in a static environment, we propose a receding horizon control method which employs a hybrid structure. The locally optimal controller is found by solving an Eikonal equation over either a local or a global environmental domain [82]. This approach captures some of the favorable properties of both local and global planning methods: the technique is local and fast for obstacle avoidance, but convergence to the goal can often be guaranteed as in global methods.

(b) We extend the proposed receding horizon controller to the case in which moving obstacles are present in the environment [83]. By accounting for both moving and static obstacles, the proposed method defines suboptimal paths locally. Perhaps most importantly, we rigorously derive a sufficient condition which ensures that the vehicle will converge to the goal by enforcing an end-point matching condition at each planning horizon.

(c) We propose a novel dynamic fast marching method ([81]) to recompute level sets (the solutions of the global Eikonal equations) when static environmental changes are detected. The computational cost is reduced compared to the conventional fast marching method [64].

(d) We successfully implement the proposed dynamic fast marching method for an autonomous surface vehicle navigating in a riverine environment.

In the following few pages, we discuss the relationship of each of these contributions to the existing literature.

Hybrid Receding Horizon Control Method for Path Planning in Static Environment

For the case in which there are only static obstacles in the environment, the success of the proposed method hinges on the careful and judicious selection of terminal costs in a receding horizon formulation. The terminal cost in this dissertation is always based on the level sets of a solution to certain Eikonal equations. If the domain is completely known, it is possible to express the minimum risk path from any point to a target point in terms of the solution to an Eikonal equation ([36] and [51]). The essence of the approach taken in this dissertation is that as we encounter new obstacles, previous solutions of Eikonal equations do not account for the newly discovered obstacles. It is possible to re-calculate a solution to the Eikonal equation over the entire domain whenever a new obstacle is identified, but this can be costly. In this dissertation, we derive and define matching conditions between local and previous global solutions of related Eikonal equations. The approach guarantees that a locally optimal path can be found without a new global solution. In order to reduce the computational expense, the proposed method first searches for a minimal risk path locally when the environment changes. If such a locally derived path violates the matching conditions, a new solution to the Eikonal equation over the global domain is calculated and used as the terminal cost in a receding horizon formulation. The selection of a global or local solution to the Eikonal equation induces a hybrid system in the control formulation. One advantage of the proposed methodology is that it captures the desirable characteristics of both global and local path planning. It is possible to achieve fast updates for local obstacle avoidance and some guarantees of the convergence to the goal.

Our contribution is to formulate path planning problems in terms of a receding horizon control (RHC) policy. Several challenges typically encountered in using RHC for autonomous navigation are addressed. Among these challenges, a major concern is that the study of the stability of the receding horizon control (see, e.g. [23], [29], [30], [48], and [57]) incorporating a final cost can be delicate. The case in which the terminal is selected from the solutions

of Eikonal equations has not yet been well addressed for path planning problems [40]. In addition, the feasible set of states for the vehicle changes as new obstacles are detected. This change of topology often causes so-called "trapping phenomena" whereby the vehicle cannot make progress toward the desired endpoint due to a limited planning horizon. Examples of this pathology are described in papers [56], [79] and [84]. To address the trapping problem, Bellingham et. al. in [5] proposes a cost function generated by a visibility graph that is constructed with respect to the goal. The vehicle travels on the path that minimizes the distance to a node in the visibility graph. In [63], a safe bound is added such that the vehicle will not enter any trap smaller than some fixed amount. Another challenge is that most current RHC methods for autonomous navigation assume that obstacle geometry is simple and can be modeled as polygons or spheres. Closed form expressions for obstacle geometry are required in some RHC methods, for example, [5], [10], [41], [63] and [84]. In practice, these assumptions are difficult to justify, particularly for outdoor natural environments.

With respect to stability issues and the trapping problem, we prove that the RHC formulation with terminal cost associated with the solution of the Eikonal equation, for the case when the map is completely known, is asymptotically stable. For the case where we have an incomplete *a priori* map, we determine a sufficient condition that guarantees that the vehicle will converge to the goal so long as there exists a feasible path in the updated map. Our approach is developed for a cellular decomposition of the environment. Thus, the proposed method does not need to assume that obstacles are polygons, for example.

Path Planning in the Presence of Moving Obstacles

We then extend the proposed RHC method to the case in which there are moving obstacles in the environment. Motion planning in the presence of moving obstacles has been addressed by a variety of authors. A global search in state-time space is developed in [17], [16] and [35]. Speed maneuvering along a predefined trajectory is studied in [18], [27] and [42]. The estimation of an obstacle trajectory cone is proposed in [15], and perturbation method along

a parameterized polynomial trajectory is employed in [24], [58] and [75]. Some approaches assume either that both moving and static obstacles possess a specific geometry, such as spheres, or that moving obstacles travel along piecewise linear trajectories, for example [15], [24], [42], [58], and [75]. The state-time space search methods ([17] and [35]) do not require these assumptions, but incorporate time as an extra dimension and model the time-varying environment as a static environment with one extra dimension. These methods can compute global optimal paths, but impose additional computational requirements.

We propose a sub-optimal controller for motion planning in a two dimensional dynamic environment that possesses both static and moving obstacles. Our general approach is cast in the framework of the level set methods. In order to simplify our analysis, we assume that the positions of both moving and static obstacles are completely known for the entire duration of the mission. Thus, our *a priori* environment map is accurate. This part of the work is inspired by the recent work of Vladimirsky [78], who derives a partial differential equation (PDE) for the time optimal control problem for non-autonomous systems. We propose a very similar PDE whose domain is the original two dimensional environment. The suboptimal controllers maneuver the vehicle along the gradient of the level sets of this PDE's solution to avoid both moving and static obstacles. Note that since the PDE is defined over the original two dimensional environment, the computational expense is much smaller than some other competing methods such as in [17] and [35].

For the sake of implementation in a realistic scenario, we then study the case in which the *a priori* map is inaccurate, and moving and static obstacles are detected during the mission by an on-board sensor with limited range. In order to make provisions for the newly detected moving and static obstacles within the sensor field of view, we propose a new receding horizon control formulation that generates local path segments [83] by incorporating the PDE studied for the case in which moving and static obstacles are *a priori* known. Similar to our results for static environments, we select the globally computed solution of the Eikonal equation as the terminal cost for the receding horizon controller. We show that this choice allows us to choose end points for local paths. Indeed, one significant achievement is that we

derive a sufficient condition, by enforcing the matching condition for the end points, that guarantees convergence to the desired target. In terms of the computational expense, one only computes the solution of the PDE over a small, local domain. The Eikonal equation would be computed over the entire domain only occasionally. Therefore, the computational expense of the proposed receding horizon control is further reduced.

Dynamics Fast Marching Method

The proposed dynamic fast marching method is based on level-set methods, which are used to compute minimum risk paths. The solution of a PDE known as the Eikonal equation is the level set. The value of the level set at any point is the cost to traverse from that point to the goal location. The optimal path is simply gradient descent of the level set ([8], [34], [36], and [37]). The solution of the Eikonal equation can be approximated by the fast marching method (FMM) [65]. This method has been successfully applied to path planning when the environments are known, for example, in [25], [36], [50] and [53]. There is limited literature that discusses the level set for replanning paths in an uncertain environment. In [54] and [55], an E* Lite algorithm is proposed which locally updates level sets at the nodes dependent on changes in the environment. A qualitative comparison between A* search and level set methods can be found in [1].

In the case that the computation of the solution of the global Eikonal equation is required, a high computational cost is incurred due to the creation of level sets over the entire domain. In order to efficiently recompute the level sets, we propose a dynamic fast marching method when the new environmental changes are detected. Our algorithm addresses, separately, the case in which new obstacles are detected, and the case in which new empty areas are detected. In comparison to the conventional fast marching method [64], our method avoids updating level sets over the entire domain. Instead, it updates the level sets only for the regions that are influenced by the newly detected obstacles or empty areas. Thus, the computational expense is reduced. The proposed algorithm draws heavily from the ideas in [54], and is

very similar in implementation. Our contribution is to present rigorous and formal analysis of how changes in the environment produce corresponding changes in the level set, and how these changes can be used to reduce computational burden.

Experimental Results

Our ultimate goal is to implement the proposed method for navigation of an autonomous vehicle in fully realistic and challenging environments such as waterways and harbors. Operations for an autonomous vehicle where existing maps are inaccurate, incomplete or where the environments are uncharted pose a significant challenge. Among the literature for path planning implementations for autonomous navigation, perhaps one of the most celebrated is achieved by the Stanley unmanned ground vehicle [72]. This vehicle won the DARPA grand challenge in 2005. Stanley adopted a local planning method [31] that can vary the lateral offset left and right with respect to the given base trajectory upon the detection of obstacles. Other vehicles that finished the DARPA challenge in 2005, and therefore have been field-tested under stringent conditions, include Sandstorm and H1ghlander [76], KAT-5 [71] and TerraMax. In terms of on-line path replanning algorithms, all of these vehicles employed local planning methods. There are limited experimental results documented in the literature that describe path planning for marine surface vehicles. In [45], a successful field trial has been reported on an SEADOO Challenger 2000 sport boat. Their path planner first finds a global path by using the A* algorithm, then a local path planner reactively chooses corresponding actions such that the vehicle can avoid the obstacles detected in real-time. In [60], an A* algorithm has been successfully implemented to an autonomous surface vehicle (ASV) that plans local paths to some way-points along a predefined global trajectory. Recently, another successful experiment on an ASV implementing the conventional fast marching method has been reported in [19]. In this dissertation, in order to validate the effectiveness of the proposed dynamic fast marching method, we conduct a field trial for an ASV navigating in a riverine environment. The vehicle managed to reach a target four kilometers away in about

45 minutes without any human interactions. To our knowledge, such a field experiment in a large scale riverine environment is among the first that have been accomplished.

1.4 Organization

This dissertation is organized as follows. Chapter 2 consists of basic definitions and results from the calculus of variations and receding horizon control theory. In Chapter 3, we present the hybrid receding horizon control method for path replanning upon the detection of new static obstacles. Further analysis provides a sufficient condition that guarantees the vehicle to converge to the goal. In Chapter 4, we further develop a new receding horizon control method for determination of the optimal trajectory in the presence of moving obstacles. In Chapter 5, we propose a novel dynamic fast marching method for the case in which the computation of the solutions of the global Eikonal equation is required. The method updates the level sets only for a portion of the domain. In Chapter 6, by implementing the proposed dynamic fast marching method, we demonstrate a successful field trial showing that an autonomous surface vehicle reaches a predefined goal four kilometers away with complete autonomy. Chapter 7 contains conclusions and a brief summary for future work.

Chapter 2

Mathematical Preliminaries

In this chapter, we introduce some concepts and tools that are extensively used in this dissertation.

Notation used throughout this dissertation is standard and any special notation is defined before its use. Generally, lowercase bold letters (e.g. $\mathbf{x}$, $\mathbf{u}$) represent vectors, uppercase bold letters (e.g. $\mathbf{A}$) represent matrices, and standard math typeface represents scalars and other quantities. The set of real numbers is denoted by $\mathcal{R}$. The set of $n \times 1$ vectors is denoted by $\mathcal{R}^n$. The set of $n \times m$ matrices is denoted by $\mathcal{R}^{n \times m}$. The Euclidean norm of a vector is denoted by $\| \cdot \|$. The class of continuously differentiable functions mapping from A into B is denoted by $C^1(A; B)$. The class of Lipschitz continuous function mapping from A into B is denoted by $Lip(A; B)$.

2.1 Calculus of Variations

Hamilton-Jacobi-Bellman Equations

Let $\mathbf{x}(t) \in \Omega \subset \mathcal{R}^n$ and $\mathbf{u}(t) \in \mathbb{U} \subset \mathcal{R}^m$. The dynamic system is

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \\ \mathbf{x}(0) = \mathbf{x}_0, \end{cases} \tag{2.1}$$

where $\mathbf{f} \in C^1(\Omega \times \mathbb{U}; \mathcal{R}^n)$. We pose an optimal control problem where we seek to find $\mathbf{u}(\cdot)$ that minimizes the functional

$$J(\mathbf{x}(\cdot)) = \min_{\mathbf{u}(\cdot)} \int_0^T L(t, \mathbf{x}(t), \mathbf{u}(t))dt + V(\mathbf{x}(T)) \tag{2.2}$$

subject to the dynamic constraints in Equation (2.1), where $L: [0,T] \times \Omega \times \mathbb{U} \to \mathcal{R}$ and $V: \Omega \to \mathcal{R}$ are given continuous functions.

Proposition 2.1.1. *([77], pp. 24) Let W be such that*

$$W(t, \mathbf{x}(t)) = \min_{\mathbf{u}(\cdot)} \int_t^T L(\tau, \mathbf{x}(\tau), \mathbf{u}(\tau))d\tau + V(\mathbf{x}(T)). \tag{2.3}$$

If $W(t, \mathbf{x}(t))$ is a continuously differentiable function, and if for each $(t, \mathbf{x}) \in [0,T] \times \mathcal{R}^n$ the optimization problem has a minimizer that is continuously differentiable, the minimizer $W(t, \mathbf{x}(t))$ is a solution to Hamilton-Jacobi-Bellman (HJB) equation satisfying

$$\begin{aligned} &\frac{\partial W}{\partial t}(t, \mathbf{x}) + \min_{\mathbf{u}(t) \in \mathbb{U}} \left(\frac{\partial W}{\partial \mathbf{x}}(t, \mathbf{x}) \cdot \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) + L(t, \mathbf{x}(t), \mathbf{u}(t))\right) = 0, \\ &W(T, \mathbf{x}(T)) = V(\mathbf{x}(T)). \end{aligned} \tag{2.4}$$

Euler-Lagrange Equations

Another means of characterizing the solution of the optimization problem in equations (2.1) and (2.2) is via the Euler-Lagrange equations. We will summarize the necessary condition (see e.g. [6], pp. 48, or [12]) for the following optimal control problem

$$J(\mathbf{x}(\cdot)) = \min_{\mathbf{u}(\cdot)} \int_0^T L(\mathbf{x}(t), \mathbf{u}(t))dt + V(\mathbf{x}(T)) \tag{2.5}$$

over a fixed time interval, where $L \in C^1(\Omega \times \mathbb{U}; \mathcal{R})$, and $V \in C^1(\Omega; \mathcal{R})$. The minimization problem is subject to the constraints in (2.1). We adjoin the system differential equation (2.1) to J with Lagrange multiplier $\boldsymbol{\lambda}(t) \in \mathcal{R}^n$:

$$\begin{aligned} \overline{J}(\mathbf{x}(\cdot)) &= \int_0^T \{L(\mathbf{x}(t), \mathbf{u}(t)) + \boldsymbol{\lambda}^T(t)[\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) - \dot{\mathbf{x}}(t)]\}dt + V(\mathbf{x}(T)) \\ &= \int_0^T \{H(\mathbf{x}(t), \mathbf{u}(t)) + \dot{\boldsymbol{\lambda}}^T(t)\mathbf{x}(t)\}dt + V(\mathbf{x}(T)) - \boldsymbol{\lambda}^T(T)\mathbf{x}(T) + \boldsymbol{\lambda}^T(0)\mathbf{x}(0) \end{aligned} \tag{2.6}$$

where

$$H(\mathbf{x}(t), \mathbf{u}(t)) := L(\mathbf{x}(t), \mathbf{u}(t)) + \boldsymbol{\lambda}^T(t)\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{2.7}$$

is called Hamiltonian. The variation in J due to variations in the control vector $\mathbf{u}(t)$ over a fixed time interval $[0, T]$ is

$$\delta\overline{J} = \left[\left(\frac{\partial V}{\partial \mathbf{x}} - \boldsymbol{\lambda}^T\right)\delta\mathbf{x}\right]_{t=T} + [\boldsymbol{\lambda}^T\delta\mathbf{x}]_{t=0} + \int_0^T \left[\left(\frac{\partial H}{\partial \mathbf{x}} + \dot{\boldsymbol{\lambda}}^T\right)\delta\mathbf{x} + \frac{\partial H}{\partial \mathbf{u}}\delta\mathbf{u}\right]dt. \tag{2.8}$$

We choose $\boldsymbol{\lambda}(t)$ such that the variations that multiply by $\delta\mathbf{x}$ vanish. Thus, $\boldsymbol{\lambda}(t)$ satisfies the following two equations:

$$\dot{\boldsymbol{\lambda}}(t) + \frac{\partial H^T}{\partial \mathbf{x}} = \mathbf{0}, \tag{2.9}$$

with boundary conditions

$$\boldsymbol{\lambda}(T) = \frac{\partial V^T}{\partial \mathbf{x}}(T). \tag{2.10}$$

Equation (2.8) then becomes

$$\delta\overline{J} = [\boldsymbol{\lambda}^T\delta\mathbf{x}]_{t=0} + \int_0^T \frac{\partial H}{\partial \mathbf{u}}\delta\mathbf{u}dt. \tag{2.11}$$

Since $\mathbf{x}(0)$ is a constant, we know $\delta\mathbf{x}(0) = \mathbf{0}$. For an extremum, $\delta\overline{J}$ must be zero for arbitrary $\delta\mathbf{u}(t)$. This can only happen if for all $t \in [0, T]$

$$\frac{\partial H^T}{\partial \mathbf{u}}(t) = \mathbf{0}. \tag{2.12}$$

Together, equations (2.9), (2.10) and (2.12) are known as the Euler-Lagrange equations.

2.2 Receding Horizon Control

The solution of the optimal control problem described in the last section over a long time period $[0, T]$ can be prohibitively expensive. We introduce the principle of receding horizon optimal control as an alternative to the global optimization over the entire interval $[0, T]$. Let $\{t_k\}_{k=0}^{\infty}$ be a sequence of times and $H > 0$ a scalar. Given a dynamic system (2.1), the receding horizon control can be summarized as follows ([23], pp. 86):

(i) At time t_k and for the current state $\mathbf{x}(t_k)$, solve a local optimal control problem over a fixed future interval $[t_k, t_k + H]$ for the optimal control $\mathbf{u}^*(\cdot)$ that minimizes

$$J_k(\mathbf{x}(\cdot)) = \min_{\mathbf{u}(\cdot)} \int_{t_k}^{t_k+H} L(\mathbf{x}(t), \mathbf{u}(t))dt + V(\mathbf{x}(t_k + H)) \tag{2.13}$$

subject to the governing equations of motion

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \\ \mathbf{x}(0) = \mathbf{x}_0, \\ \mathbf{x}(t) \in \overline{\Omega}, \\ \mathbf{u}(t) \in \mathbb{U}. \end{cases} \tag{2.14}$$

(ii) Apply the optimal control input $\mathbf{u}^*$ for only the subinterval $[t_k, t_k + h] \subseteq [t_k, t_k + H]$.

(iii) Measure the state reached at time $t_k + h$.

(iv) Let $t_{k+1} = t_k + h$. Repeat the local horizon optimization embodied in equations (2.13) and (2.14) at time t_{k+1} over the next interval $[t_{k+1}, t_{k+1} + H]$ starting from the current state $\mathbf{x}(t_{k+1})$.

Stability for Receding Horizon Control

The stability of receding horizon controllers has been addressed in numerous articles such as [23], [29], [30], and [48] over the past few years. We state a stability result for the continuous version.

Theorem 2.2.1. *([30]) Consider the receding horizon control problem summarized in equations (2.13) and (2.14). Assume* $L(\mathbf{x}, \mathbf{u}) \geq c(\|\mathbf{x}\| + \|\mathbf{u}\|)$ *for some* $c > 0$ *and* $L(\mathbf{0}, \mathbf{0}) = 0$. *Suppose the terminal cost* $V(\mathbf{x}(t))$ *is a Lyapunov function such that there exists a feedback control input* $\mathbf{u}_f(\cdot)$ *that stabilizes (2.14) while satisfying*

$$\dot{V}(\mathbf{x}(t)) + L(\mathbf{x}, \mathbf{u}_f) \leq 0. \tag{2.15}$$

If the control input $\mathbf{u}^*$ *minimizes* J_k, *then* J_k *is a Lyapunov function satisfying*

$$J_{k+1}(\mathbf{x}(\cdot)) - J_k(\mathbf{x}(\cdot)) \leq -\int_{t_k}^{t_{k+1}} L(\mathbf{x}(\tau), \mathbf{u}^*(\tau))d\tau. \tag{2.16}$$

Moreover, J_k *satisfies*

$$\lim_{h \to 0} \frac{J_{k+1}(\mathbf{x}(\cdot)) - J_k(\mathbf{x}(\cdot))}{h} \leq -L(\mathbf{x}(t_k), \mathbf{u}^*(t_k)) \leq -c\|\mathbf{x}(t_k)\|. \tag{2.17}$$

and the dynamic system (2.14) is asymptotically stabilized given the optimal input $\mathbf{u}^*$.

2.3 Eikonal Equation and Level Sets

Denote a connected and bounded open set by $\Omega \subset \mathcal{R}^n$. An Eikonal equation ([65]) is a particular partial differential equation (PDE) defined over $\overline{\Omega}$ satisfying

$$\begin{aligned} \|\nabla Q(\boldsymbol{\xi})\| &= g(\boldsymbol{\xi}), \\ Q(\mathbf{z}) &= 0, \end{aligned} \tag{2.18}$$

where $\boldsymbol{\xi} \in \overline{\Omega}$ and $\mathbf{z}$ is the origin. The existence and uniqueness of the viscosity solution for the Eikonal equation in $\overline{\Omega}$ is proven in Theorem 5.1, pp. 117, [47]. In both [47] and [28], it is also shown that the viscosity solution of (3.4) is Lipschitz continuous and is bounded over the domain $\overline{\Omega}$.

Chapter 3

A Hybrid Receding Horizon Controller for Path Planning in Static Environments

We propose a hybrid receding horizon control for path planning when new static obstacles are detected by an on-board sensor having limited range. The hybrid method uses the level sets of the solution of either a global or local Eikonal equation. Whenever an obstacle is detected along the path of the autonomous vehicle, a solution to a local Eikonal equation is used to determine whether a new, global Eikonal equation must be solved. The decision to select a new level set is made based on certain matching conditions that guarantee the optimality of the path. The selection of a global or local solution to the Eikonal equation induces the structure of a hybrid system in the control formulation. We rigorously prove sufficient conditions that guarantee that the vehicle will converge to the goal as long as the goal is accessible. The chapter is organized as follows. In Section 3.1, we introduce the vehicle model, the formulation of the level sets method as well as the measurement process used to detect new obstacles as they are encountered. In Section 3.2, we discuss the stability of a RHC controller that uses the solution to an Eikonal equation as the terminal cost when the

environment is completely known. The receding horizon control strategy that is used when new static obstacles are detected is introduced in Section 3.3. In Section 3.4, we propose the hybrid receding horizon control methodology and derive sufficient conditions that guarantee its convergence. The computational cost for the proposed method and simulation results are discussed in Section 3.5.

3.1 Problem Formulation

3.1.1 Vehicle Model

Consider an autonomous vehicle navigating in $\overline{\Omega} \subset \mathcal{R}^2$, where Ω is a connected and bounded open set in $\mathcal{R}^2$. The vehicle is regarded as a point mass since it is small relative to $\overline{\Omega}$. The task for the vehicle is to travel along an obstacle free path such that the vehicle can reach a predefined goal $\mathbf{z} \in \overline{\Omega}$. Letting the vehicle position at any time t be $\mathbf{x}(t) \in \mathcal{R}^2$, we model the motion of an autonomous vehicle as

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{u}(t), \\ \mathbf{x}(t_0) &= \mathbf{x}_0 \in \overline{\Omega}, \end{aligned} \tag{3.1}$$

where $\mathbf{x}(t)$ is the location of the vehicle in $\mathcal{R}^2$ and $\mathbf{u}(t)$ is the input. We assume that the vehicle can turn without forward motion and can make a sudden stop as soon as the vehicle arrives at $\mathbf{z}$. This assumption is reasonable since we are concerned with altering the heading of the vehicle so that it does not collide with obstacles. We model the admissible input as $\mathbf{u}(t) \in \mathbb{U}$, where

$$\mathbb{U} := \left\{ \mathbf{u} \in \mathcal{R}^2 : \ \|\mathbf{u}\| \leq v_{\max} \right\} \tag{3.2}$$

and $v_{\max}$ is a scalar corresponding to the maximum speed of the vehicle.

3.1.2 Optimal Trajectories For Known Geometry and The Eikonal Equation

In this section, we review some of the well-known characterizations of optimal trajectories when the environment is completely known. For specific choices of the risk, these trajectories can be obtained via the solution of the classical Eikonal equation (2.18). The methodology presented later in this chapter will use these solutions to define a receding horizon control policy for unknown geometries.

For each point $\boldsymbol{\xi}$ in $\overline{\Omega} \subset \mathcal{R}^2$, we associate a risk for the vehicle to traverse $\boldsymbol{\xi}$ by a cost function $g \in C^1(\overline{\Omega}; \mathcal{R})$. This function is positive everywhere except at the goal $\mathbf{z}$, where $g(\mathbf{z}) = 0$. The value of $g(\cdot)$ can be chosen according to the map information. For example, for the *a priori* map of a riverine environment shown in Figure 3.1, if the risk traversing the region near the shores is higher than traveling in the middle of the river, one can select larger g values for the domain close to the shorelines in order to penalize the paths traversing this area. Another example of the selection of the value g is to interpret it as occupancy probability ([11]). Since the map is often represented by a set of occupancy grids, we can set the g value proportional to the probability that the corresponding cell in the grid is occupied. In so doing, the path planner will tend to search for a path that has lower overall risk by avoiding collisions with the cells of higher occupied probability. This idea is discussed in detail in Chapter 6.

To compute the minimal cumulative cost to travel from any point $\boldsymbol{\xi} \in \overline{\Omega}$ to $\mathbf{z}$, we define a function $Q(\boldsymbol{\xi})$ satisfying

$$Q(\boldsymbol{\xi}) = \min_{\mathbf{c}} \int_0^1 g(\mathbf{c}(p)) \, \|\mathbf{c}'(p)\| \, dp \tag{3.3}$$

where where $\mathbf{c} \in Lip([0,1]; \overline{\Omega})$ is a Lipschitz continuous parameterized path from the starting point $\mathbf{c}(0) = \boldsymbol{\xi}$ to the goal $\mathbf{c}(1) = \mathbf{z}$ (see e.g. [47], pp. 116). It is well-known (see, e.g. [36]) that the solution of this optimization problem is characterized by the solution of the Eikonal

equation

$$\begin{aligned} \|\nabla Q(\boldsymbol{\xi})\| &= g(\boldsymbol{\xi}), \\ Q(\mathbf{z}) &= 0. \end{aligned} \tag{3.4}$$

The existence and uniqueness of the viscosity solution (see e.g. [47]) in $\overline{\Omega}$ is proved in Theorem 5.1, pp. 117, [47]. In [47], it is also shown that the viscosity solution of (3.4) is Lipschitz continuous and bounded over the domain $\overline{\Omega}$. The solution value $Q(\boldsymbol{\xi})$ is equal to the overall minimal risk to travel from the point $\boldsymbol{\xi}$ to the goal $\mathbf{z}$. A level set of Q is the set of points $\boldsymbol{\xi}$ that have the same Q values

$$\left\{\boldsymbol{\eta} \in \overline{\Omega} : \ Q(\boldsymbol{\eta}) = c\right\} \tag{3.5}$$

for some constant c. The optimal paths are along the gradient of level sets. Figure 3.1 shows an example of *a priori* map, and Figure 3.2 shows the contours of the corresponding level set. Figure 3.3 illustrates that a minimum-risk path progresses in the gradient direction down the level set.

The reader should note that if the risk and geometry are known, one needs only to solve the equation (3.3), or equivalently, (3.4) one time before travel. The optimal policy is to choose $\dot{\mathbf{x}}(t)$ along the gradient descent direction of the level sets that are characterized by (3.4). However, in our control formulation, the geometry and its corresponding risk g vary as new obstacles are detected. In the next section, we discuss the observation process and evolution of estimates of new geometry.

3.1.3 Observation Process and New Eikonal Equation

As noted earlier, there are unmarked obstacles due to the inconsistency between the *a priori* map and the actual environment. Denote the actual environment by Ω and the initial *a priori* map by $\Omega(0)$, which again is a bounded open set in $\mathcal{R}^2$. We assume that $\Omega \subseteq \Omega(0)$.

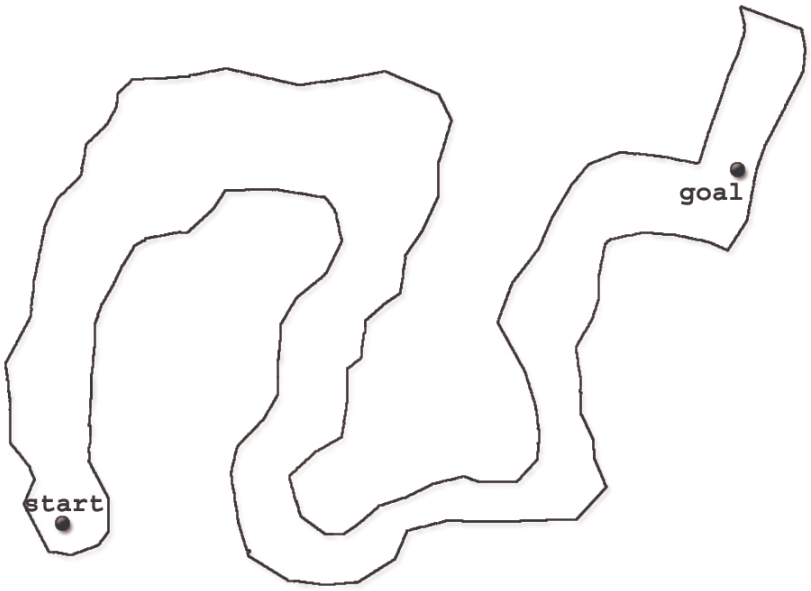

Figure 3.1: An *a priori* map $\overline{\Omega}$.

This is consistent with the requirement that actual environment has unexpected obstacles compared to the initial *a priori* map. We denote the set of unmarked obstacles by the closure of an open set $\mathcal{O}$ in $\mathcal{R}^2$ which satisfies $\mathcal{O} = \Omega(0) \setminus \overline{\Omega}$.

Let the detection range of an onboard sensor be r, and suppose that measurements are made at a period of h. Letting

$$t_k = t_0 + kh, \quad k = 1, 2, 3, \ldots, \tag{3.6}$$

we denote $\Omega(k)$ the updated map at t_k. The new geometry $\Omega(k+1)$ depends on the obstacles that the vehicle can detect during the time period $[t_k, t_k + h]$ and satisfies

$$\Omega(k+1) = \Omega(k) \setminus (\overline{\cup_{t\in[t_k,t_k+h]} B_r(\mathbf{x}(t)) \cap \mathcal{O}}) \tag{3.7}$$

where $B_r(\mathbf{x}(t))$ is an open ball with radius r and center $\mathbf{x}(t)$. For example, Figure 3.4 shows the detection range $B_r(\mathbf{x}(t))$ when the vehicle is at point A and, correspondingly, the shaded area observed at time t. The detected obstacles during the time interval $[t_k, t_k + h]$ is the area that is swept by the dashed circle along the trajectory from A to B.

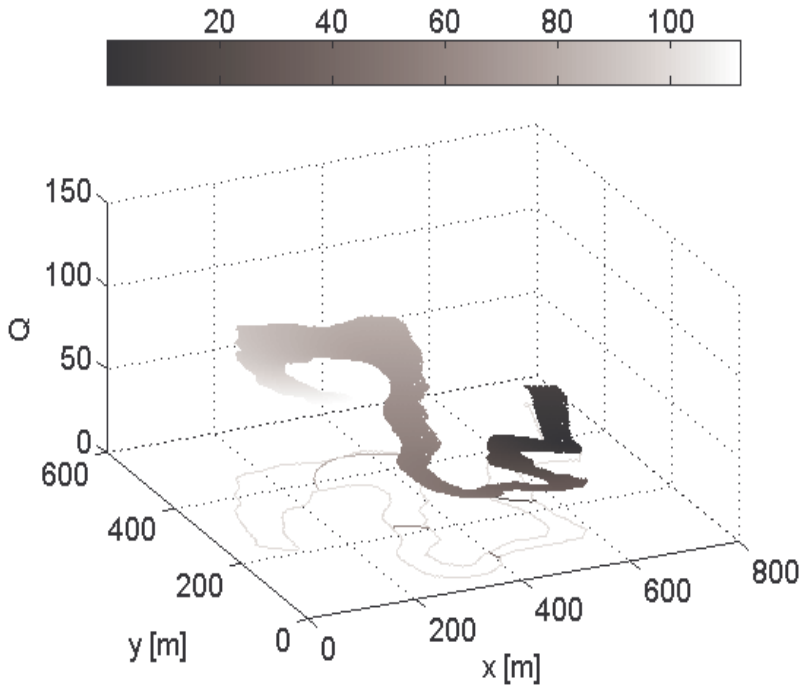

Figure 3.2: The level sets for the *a priori* map.

Note that there are two different approaches to represent the geometry changes. First, we can increase the cost function g for the regions that correspond to the newly detected obstacles. In so doing, the cost to traverse an obstacle is significantly higher than to pass through empty areas. Thus, the path planner will select a trajectory that avoids the obstacles such that the overall risk cost is as low as possible. Such a map representation will be employed in Chapter 4 and Chapter 5. In this chapter, instead, we represent the geometry changes by restricting the feasible domain for the autonomous vehicle as the observation process picks out new obstacles. Thus, the path planner will not plan any paths traversing any obstacles. To do so, we define a new Eikonal equation over a strictly smaller domain. That is, for $\boldsymbol{\xi} \in \overline{\Omega(k)}$, the Eikonal equation for $Q_k(\boldsymbol{\xi})$ becomes

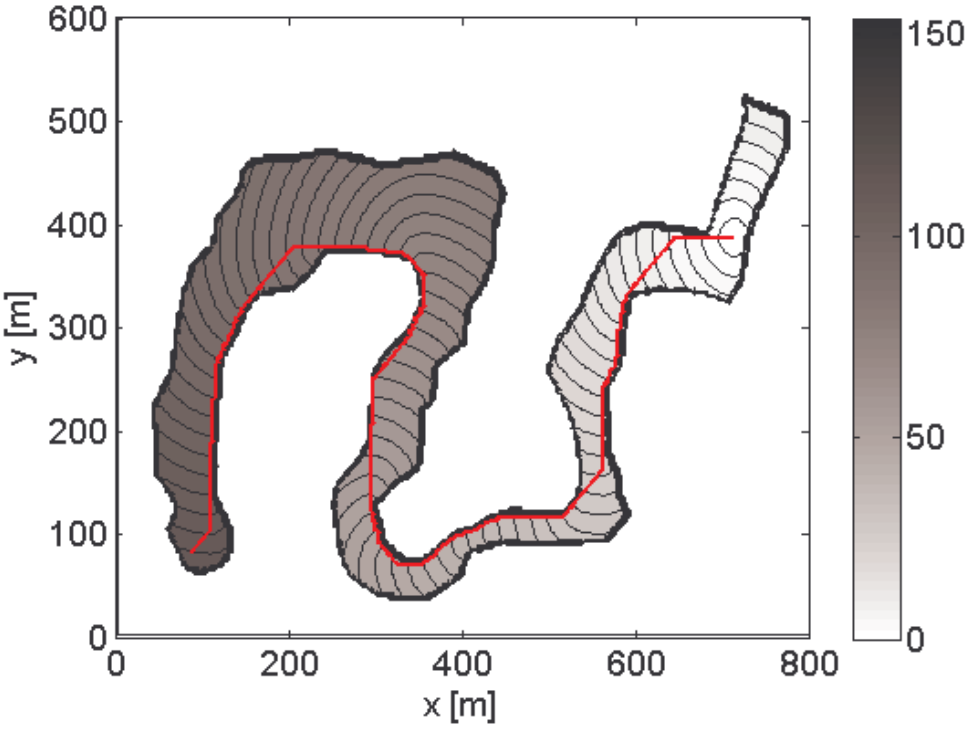

Figure 3.3: The level sets contours and the optimal path for the *a priori* map.

$$\begin{aligned} \|\nabla Q_k(\boldsymbol{\xi})\| &= g(\boldsymbol{\xi})|_{\overline{\Omega(k)}}, \\ Q_k(\mathbf{z}) &= 0. \end{aligned} \tag{3.8}$$

where $\boldsymbol{\xi} \in \overline{\Omega(k)}$, $g \in C^1(\overline{\Omega(0)}; \mathcal{R})$ and $g(\boldsymbol{\xi})|_{\overline{\Omega(k)}}$ denotes the restriction of g to $\overline{\Omega(k)}$. We notice that in contrast to (3.4), the solution of the above Eikonal calculates the minimal cost to travel from $\boldsymbol{\xi}$ to $\mathbf{z}$ in the set $\overline{\Omega(k)}$.

3.2 Asymptotic Stability of RHC: Known Geometry

In this section, we introduce our first receding horizon control formulation that will serve as the model for more challenging variants presented in later sections. We show that if the

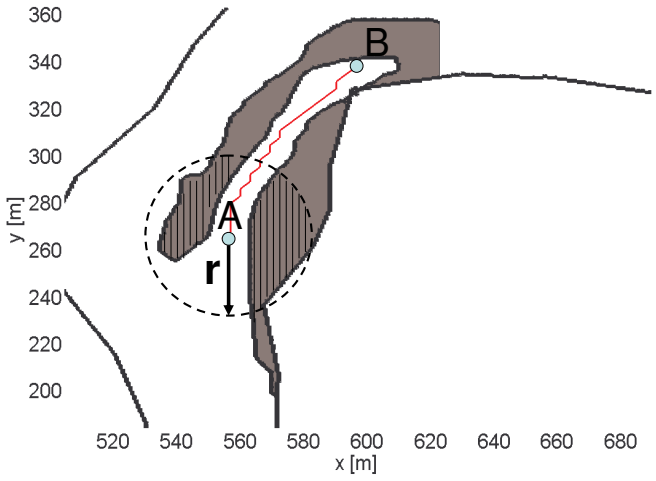

Figure 3.4: A vehicle travels from A to B during $[t_k, t_k + h]$. The dash circle is the detection range. The grey area represents the new obstacles that are not marked in the *a priori* map. The shaded area corresponds to the detected obstacles when the vehicle is at point A.

map is accurately known, we can apply the methodology proposed in [48] and [29] to prove the asymptotical stability of the trajectory.

We assume that the *a priori* map is accurate, so that the *a priori* map $\Omega(0)$ is equal to the actual environment Ω. We let the implementation horizon be h which is identical to the sensor's sampling period, although there are no new obstacles to be detected in this case. We choose H as the planning horizon where $H \geq h$. We aim to minimize the risk for traversal during the horizon H, as well as the expected minimal risk for traversal from the terminal state $\mathbf{x}(t_k + H)$ toward the goal. The receding horizon formulation is then to find the local optimal control and state pair on $[t_k, t_k + H]$ that solves the minimization problem

$$J_k(\mathbf{x}(\cdot)) = \min_{\mathbf{u}(\cdot)\in\mathbb{U}} \int_{t_k}^{t_k+H} g(\mathbf{x}(\tau))|_{\overline{\Omega}} \|\dot{\mathbf{x}}(\tau)\| d\tau + Q(\mathbf{x}(t_k + H)) \tag{3.9}$$

subject to

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{u}(t), \\ \mathbf{x}(t_0) = \mathbf{x}_0, \\ \mathbf{x}(t_k) = \mathbf{x}(t_{k-1} + h), \\ \mathbf{x}(t) \in \overline{\Omega}. \end{cases} \tag{3.10}$$

for $k = 0, 1, 2, \ldots, \infty$. The overall planned trajectory and control on $[t_0, \infty)$ are spliced together from the locally optimal trajectory in the usual way (see, e.g. [23]).

Proposition 3.2.1. *Assume that the viscosity solution of $Q \in C^1(\overline{\Omega}; \mathcal{R})$. Define the value function*

$$V(\mathbf{x}(t_k)) = J_k(\mathbf{x}(\cdot)). \tag{3.11}$$

If $V(\mathbf{x}(t_k))$ is continuously differentiable, the goal $\mathbf{z}$ is asymptotically stable.

Remark 3.2.2. *In the general case, $Q(\boldsymbol{\xi})$ is not differentiable everywhere. Interested readers are referred to [2], [3], [4] and [67] for the stability analysis when a Lyapunov function is not a C^1 function.*

Proof of Proposition 3.2.1. Following the procedure in [48] and [29], we denote by $\mathbf{u}_f(t)$ the admissible feedback controller that ensures the vehicle to be asymptotically stable with respect to $\mathbf{z}$. Given the level set values Q, let the controller $\mathbf{u}_f$ satisfy

$$\mathbf{u}_f(t) = -\gamma(t) v_{\max} \frac{\nabla Q(\mathbf{x}(t))}{\|\nabla Q(\mathbf{x}(t))\|}, \text{ if } \mathbf{x}(t) \neq \mathbf{z}, \tag{3.12}$$

for all $t \in [t_0, \infty)$, where $\gamma(t) \in (0, 1]$. Thus, we infer that

$$\dot{Q}(\mathbf{x}(t)) = \nabla Q(\mathbf{x}(t)) \cdot \mathbf{u}_f(t) = -g(\mathbf{x}(t)) \|\mathbf{u}_f(t)\|, \tag{3.13}$$

and $Q(\mathbf{x}(t))$ is a control Lyapunov function. Denote the optimal controller that minimizes $J_k(\mathbf{x}(\cdot))$ by $\mathbf{u}^*(t)$ for $t \in [t_k, t_k + H]$. Since $t_{k+1} = t_k + h$ and since $h \leq H$, we can define the admissible controller $\mathbf{u}^+(t)$ satisfying

$$\mathbf{u}^+(t) = \begin{cases} \mathbf{u}^*(t), \ \forall t \in [t_{k+1}, t_{k+1} + H - h], \\ \mathbf{u}_f(t), \ \forall t \in [t_{k+1} + H - h, t_{k+1} + H]. \end{cases} \tag{3.14}$$

Given the initial state $\mathbf{x}(t_{k+1})$ at t_{k+1}, we denote by $\mathcal{J}(\mathbf{x}(t_{k+1}), \mathbf{u}^+(\cdot))$ the cost function that is evaluated by $\mathbf{u}^+(\cdot)$ satisfying

$$\mathcal{J}(\mathbf{x}(t_{k+1}), \mathbf{u}^+(\cdot)) := \int_{t_{k+1}}^{t_{k+1}+H} g(\mathbf{x}(\tau))\|\mathbf{u}^+(\tau)\|d\tau + Q(\mathbf{x}(t_{k+1}+H)). \tag{3.15}$$

Since $J_{k+1}(\mathbf{x}(\cdot))$ is the minimal cost, we obtain the following inequality

$$\begin{aligned}
&J_{k+1}(\mathbf{x}(\cdot)) \\
\leq&\mathcal{J}(\mathbf{x}(t_{k+1}), \mathbf{u}^+(\cdot)) \\
=&\int_{t_{k+1}}^{t_{k+1}+H-h} g(\mathbf{x}(\tau))\|\mathbf{u}^*(\tau)\|d\tau \\
&+\int_{t_{k+1}+H-h}^{t_{k+1}+H} g(\mathbf{x}(\tau))\|\mathbf{u}_f(\tau)\|d\tau + Q(\mathbf{x}(t_{k+1}+H)) \\
=&\int_{t_k+h}^{t_k+H} g(\mathbf{x}(\tau))\|\mathbf{u}^*(\tau)\|d\tau \\
&+\int_{t_k+H}^{t_k+h+H} g(\mathbf{x}(\tau))\|\mathbf{u}_f(\tau)\|d\tau + Q(\mathbf{x}(t_{k+1}+H)) \\
=&J_k(\mathbf{x}(\cdot)) - \int_{t_k}^{t_k+h} g(\mathbf{x}(\tau))\|\mathbf{u}^*(\tau)\|d\tau - Q(\mathbf{x}(t_k+H)) \\
&+\int_{t_k+H}^{t_k+h+H} g(\mathbf{x}(\tau))\|\mathbf{u}_f(\tau)\|d\tau + Q(\mathbf{x}(t_k+H+h)).
\end{aligned} \tag{3.16}$$

Since $\mathbf{u}_f(t)$ satisfies (3.13), together with the inequality (3.16), we infer that

$$J_{k+1}(\mathbf{x}(\cdot)) - J_k(\mathbf{x}(\cdot)) \leq -\int_{t_k}^{t_k+h} g(\mathbf{x}(\tau))\|\mathbf{u}^*(\tau)\|d\tau. \tag{3.17}$$

According to definition of $V(\mathbf{x}(t_k))$, $J_k(\mathbf{x}(\cdot))$ and $\mathcal{J}(\mathbf{x}(t_k), \mathbf{u}(\cdot))$, we have

$$V(\mathbf{x}(t_k)) = \min_{\mathbf{u}\in\mathbb{U}} \mathcal{J}((\mathbf{x}(t_k), \mathbf{u}(\cdot)) = J_k(\mathbf{x}(\cdot)). \tag{3.18}$$

Note that $V(\mathbf{z}) = 0$ and for any $\mathbf{x} \neq \mathbf{z}$, $V(\mathbf{x}) > 0$. According to the assumption, $V(\mathbf{x})$ is continuously differentiable with respect to $\mathbf{x}$. Due to (3.17), the following inequality holds

$$\lim_{h\to 0} \frac{V(\mathbf{x}(t_{k+1})) - V(\mathbf{x}(t_k))}{h} \leq -g(\mathbf{x}(t_k))\|\mathbf{u}^*(t_k)\| < 0. \tag{3.19}$$

which completes the proof for asymptotical stability according to [29]. □

Note that in the above proof, the closed form of the optimal controller $\mathbf{u}^*$ is not required. The stability is proved by using the optimality property of the cost function $J_k(\mathbf{x}(\cdot))$.

3.3 A Receding Horizon Control Approach with an Incomplete *a priori* Map

In this section, we show the receding horizon control when the *a priori* map is not accurate. Suppose at time $t_k \geq t_0$, we have computed $Q_k(\mathbf{x})$ defined in (3.8) corresponding to the newly updated map $\overline{\Omega(k)}$. A new receding horizon problem seeks to find the state and control pair on the time interval $[t_k, t_k + H]$ that minimizes the functional

$$J_k(\mathbf{x}(\cdot)) = \min_{\mathbf{u}(\cdot)\in\mathbb{U}} \int_{t_k}^{t_k+H} g(\mathbf{x}(\tau))|_{\overline{\Omega(k)}} \|\dot{\mathbf{x}}(\tau)\| d\tau + Q_k(\mathbf{x}(t_k + H)) \tag{3.20}$$

subject to

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{u}(t), \\ \mathbf{x}(t_0) = \mathbf{x}_0, \\ \mathbf{x}(t_k) = \mathbf{x}(t_{k-1} + h), \\ \mathbf{x}(t) \in \overline{\Omega(k)}. \end{cases} \tag{3.21}$$

Just as in equations (3.9) and (3.10), the above cost function aims to minimize both the risk of traversal during the horizon H and the expected minimal risk for traversal from the terminal state $\mathbf{x}(t_k + H)$ toward the goal. The following proposition characterizes the trajectory that minimizes the above cost function.

Proposition 3.3.1. *The optimal trajectory that minimizes the cost function (3.20) satisfies, for* $t \in [t_k, t_k + H]$

$$\mathbf{u}(t) = \begin{cases} -v_{\max} \frac{\nabla Q_k(\mathbf{x}(t))}{\|\nabla Q_k(\mathbf{x}(t))\|}, & \text{if } \mathbf{x}(t) \neq \mathbf{z}, \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{3.22}$$

Moreover, $J_k(\mathbf{x}(\cdot)) = Q_k(\mathbf{x}(t_k))$.

Proof of Proposition 3.3.1. The proof is a direct consequence of the conclusions in [36] and [47]. Substituting (3.22) into the cost function (3.20), we conclude that $J_k(\mathbf{x}(\cdot)) = Q(\mathbf{x}(t_k))$. □

3.4 Hybrid Receding Horizon Control Method

The RHC method proposed in Section 3.3 requires the computation of Q_k whenever an unexpected obstacle is detected. This process can be computationally expensive. Seeking to minimize the frequency to update Q_k, we propose a variant of the RHC proposed in Section 3.3 in which Q_k will be computed less frequently.

3.4.1 Preliminaries

In the hybrid RHC method, our approach to reduce the computational cost by replacing the solutions for Q_k with a smaller, local problem. Recall that $Q_k(\boldsymbol{\xi})$ encodes the cost of travel from point $\boldsymbol{\xi}$ to the goal $\mathbf{z}$ corresponding to the most recent map $\overline{\Omega(k)}$ detected at time t_k. We define $Q_k^*(\boldsymbol{\xi})$ to be the solution of the Eikonal equation

$$\|\nabla Q_k^*(\boldsymbol{\xi})\| = g(\boldsymbol{\xi})|_{\overline{\Omega(k)}},\ Q_k^*(\mathbf{x}(t_k)) = 0. \tag{3.23}$$

where $\overline{\Omega(k)}$ is the updated map at t_k and $\mathbf{x}(t_k)$ is the corresponding location of the vehicle. In contrast to $Q_k(\boldsymbol{\xi})$, $Q_k^*(\boldsymbol{\xi})$ encodes the cost to travel from the current vehicle location $\mathbf{x}(t_k)$ to a point $\boldsymbol{\xi} \in \overline{\Omega(k)}$. We will see in Section 3.4.2 that by calculating the solution $Q_k^*(\boldsymbol{\xi})$ over a relatively small neighborhood of $\mathbf{x}(t_k)$, it is often possible to forego the calculation of the solution $Q_k(\boldsymbol{\xi})$ over the entire domain. We continue to use the old global level set corresponding to the map detected at the previous time as the terminal cost, unless some

matching conditions to be defined later are violated. By doing this, we can reduce the frequency with which we must solve an Eikonal equation over the global domain. In order to indicate which level sets are used at time t_k, we introduce a new index $n(k)$ satisfying $n(k) \leq k$. For example, in some cases, we may use Q_0 corresponding to the *a priori* map $\overline{\Omega(0)}$ as the terminal cost in $J_k(\mathbf{x}(\cdot))$ throughout the entire mission. In this case, $n(k) \equiv 0$, $k = 0, 1, 2, \ldots$.

3.4.2 Steps for the Hybrid Receding Horizon Control

The procedure for the hybrid RHC is summarized in the following steps:

Initialization: Calculate Q_0. Let $n(0) = 0$.

Step 1: At time t_k, we seek the vehicle state and control on $[t_k, t_k + H]$ which minimizes

$$J_k(\mathbf{x}(\cdot)) = \min_{\mathbf{u}(\cdot)\in\mathbb{U}} \int_{t_k}^{t_k+H} g(\mathbf{x}(\tau))|_{\overline{\Omega(k)}} \|\mathbf{u}(\tau)\| d\tau + Q_{n(k)}(\mathbf{x}(t_k + H)) \tag{3.24}$$

subject to (3.21).

Step 2: If there is no obstacle along the path generated by the controller satisfying

$$\mathbf{u}^*(t) = \begin{cases} -v_{\max} \frac{\nabla Q_{n(k)}(\mathbf{x}(t))}{\|\nabla Q_{n(k)}(\mathbf{x}(t))\|}, & \text{if } \mathbf{x}(t) \neq \mathbf{z}, \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{3.25}$$

for $t \in [t_k, t_k + H]$, we let $\mathbf{u}(t) = \mathbf{u}^*(t)$ for $t \in [t_k, t_k + h]$. Then, we let $n(k+1) = n(k)$ and go to *Step 1* for the next optimization interval $[t_{k+1}, t_{k+1} + H]$. We will show that (3.25) is the optimal controller in Theorem 3.4.1.

Step 3: If there are obstacles along the path generated by $\mathbf{u}^*$ in (3.25), we solve the Eikonal equation (3.23) over the set of all points accessible from $\mathbf{x}(t_k)$ during the planning horizon $[t_k, t_k + H]$. Since the maximum speed of the vehicle is $v_{\max}$, the domain that is reachable

for the vehicle over the planning horizon is $\overline{B_{v_{\max}H}(\mathbf{x}(t_k))}$, where $B_{v_{\max}H}(\mathbf{x}(t_k))$ is an open ball with radius $v_{\max}H$ and center $\mathbf{x}(t_k)$. Therefore, Q_k^* is computed over the local domain $\overline{B_{v_{\max}H}(\mathbf{x}(t_k)) \cap \Omega(k)}$.

Step 4: For each trajectory that satisfies

$$\mathbf{u}^*(t) = \begin{cases} v_{\max}\frac{\nabla Q_k^*(\mathbf{x}((t)))}{\|\nabla Q_k^*(\mathbf{x}((t)))\|}, & \text{if } \mathbf{x}(t) \neq \mathbf{z} \\ \mathbf{0}, & \text{otherwise,} \end{cases} \tag{3.26}$$

we check if the following two matching conditions hold for $\mathbf{u}^*(t)$,

Condition 1 (Convergence Condition):

$$\frac{1}{v_{\max}}\mathbf{u}^*(t) \cdot \nabla Q_{n(k)}(\mathbf{x}(t)) \leq -\gamma\|\nabla Q_{n(k)}(\mathbf{x}(t))\|, \ \forall \tau \in [t_k, t_k + h] \tag{3.27}$$

where $\gamma \in (0, 1]$ is a predefined constant.

Condition 2 (Optimality Condition):

$$\nabla Q_k^*(\mathbf{x}(t_k + H)) = -\nabla Q_{n(k)}(\mathbf{x}(t_k + H)) \tag{3.28}$$

In practice, we set some tolerances to check whether the two conditions are satisfied in the discretized domain. For example, we relax *Condition 2* so that if the difference between the two sides of (3.28) is smaller than a given threshold we say that *Condition 2* is satisfied. An example of an optimal trajectory satisfying *Condition 2* is shown in Figure 3.5. Since there are a few cells in the discretized local domain $\overline{B_{v_{\max}H}(\mathbf{x}(t_k)) \cap \Omega(k)}$, we can check the trajectories generated by the controller in (3.26) from every node in this discretized domain to the node corresponding to the vehicle's current location. If there is a trajectory satisfying both matching conditions, we let $\mathbf{u} = \mathbf{u}^*(t)$ and implement $\mathbf{u}(t)$ for $t \in [t_k, t_k + h]$. We let $n(k + 1) = n(k)$ and go to *Step 1*. We continue to use the old $Q_{n(k)}$ value for the next receding horizon control (3.24) for $[t_{k+1}, t_{k+1} + H]$.

Step 5: If there is no such trajectory generated by (3.26) that satisfies both *Condition 1* and *2*, we recompute Q_k over the entire domain $\overline{\Omega(k)}$ and seek $\mathbf{u}(t)$ such that the cost function

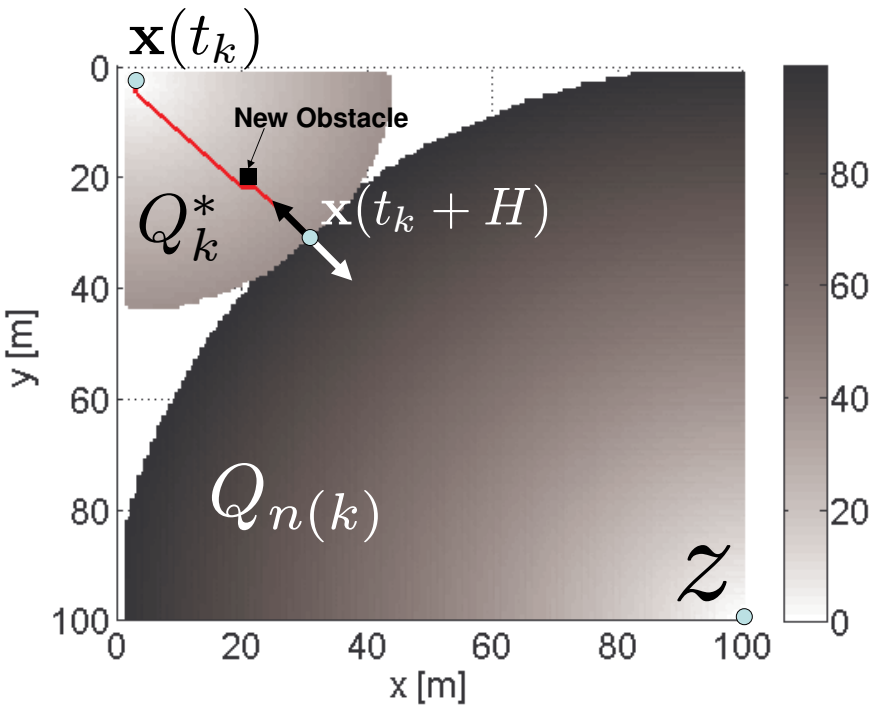

Figure 3.5: An example of an optimal trajectory (the red line) satisfying *Condition 2*. The white and black arrows perpendicular to the border between Q_k^* and $Q_{n(k)}$ are, respectively, ∇Q_k^* and $\nabla Q_{n(k)}$.

(3.20) is minimized. Then, we set $\mathbf{u}$ to be (3.22). In this case, $n(k) = k$. Then, we let $n(k+1) = k$ and go to *Step 1* which indicates that we use Q_k for next receding horizon control (3.24) for $[t_{k+1}, t_{k+1} + H]$.

We stop the process when the vehicle reaches the goal.

We now justify the proposed hybrid receding receding horizon control algorithm.

Theorem 3.4.1. *Consider the hybrid receding horizon control algorithm summarized in* Step 1 *through* Step 5. *We have the following:*

(*a*) *Assume there exists a finite time* $t_N \geq t_0$ *such that*

$$Q_{n(k)}(\boldsymbol{\xi}) = Q_N(\boldsymbol{\xi}), \ \forall \boldsymbol{\xi} \in \Omega(N) \tag{3.29}$$

for all $t_k \geq t_N$. *Assume that* $Q_N(\boldsymbol{\xi})$ *is continuously differentiable with respect to* $\boldsymbol{\xi}$. *Then,* $\mathbf{x}(t) \to \mathbf{z}$ *as* $t \to \infty$.

(*b*) *For* Step 2, *if there is no obstacle along the trajectory generated by the control* $\mathbf{u}^*$ *in (3.25),* $\mathbf{u}^*$ *is the control input that minimizes the cost function (3.24).*

(*c*) *For* Step 4, *if* Condition 2 *holds, the control input that necessarily minimizes the cost function (3.24) is (3.26).*

Remark 3.4.2. *Such finite time* t_N *required in the assumption of Theorem 3.4.1*(*a*) *exists in many practical problems. Since objects have a finite minimum size, there are finite number of obstacles in the bounded domain* $\overline{\Omega(0)}$. *Since* Step 2 *indicates that the global level sets* Q_k *do not need to be recomputed unless there is an obstacle on trajectory (3.25), the global level sets will only be updated a finite number of times. Thus, the time* t_N *exists and level sets are not updated after* t_N. *One may question whether it is possible that the vehicle moves along a periodic trajectory. We preclude the case by contradiction. Note that the optimal controllers in both* Step 2 *and* Step 4 *(see* Condition 1*) are along the gradient descent directions of the global level set* $Q_{n(k)}$. *Therefore, throughout the time interval in which* $Q_{n(k)}$ *is not re-computed, it is guaranteed that the vehicle does not traverse the same location multiple times. Thus, by induction, we conclude that the vehicle traverses the same location an infinite number of times only if the global level sets are updated infinitely many times due to the infinite number of new obstacles. This contradicts our previous conclusion that the number of new obstacles is finite.*

Remark 3.4.3. *For the case that* Q_N *is not differentiable everywhere, readers are referred to [2], [3] and [67].*

Remark 3.4.4. *For* Step 4, *the trajectory generated by* $\mathbf{u}^*$ *in (3.26) is suboptimal and it satisfies Euler-Lagrange equation. Theorem 3.4.1*(*c*) *shows that the optimal trajectories satisfying both* Condition 1 *and* Condition 2 *are included in the set of trajectories generated by* $\mathbf{u}^*$ *in (3.26). However, we need to check whether these trajectories satisfy both* Condition 1 *and* Condition 2.

Proof of Theorem 3.4.1. We first show (a). Given the assumption that the global level sets Q_k are not updated after the time t_N, we infer that the vehicle travels along the trajectories generated either by (3.25) or (3.26). First, if the vehicle is moving along (3.25) for some $t_k \geq t_N$, we can derive the following inequality

$$\frac{1}{v_{\max}}\mathbf{u}(t) \cdot \nabla Q_N(\mathbf{x}(t)) = -\|\nabla Q_N(\mathbf{x}(t))\| \leq -\gamma\|\nabla Q_N(\mathbf{x}(t))\|, \tag{3.30}$$

where $\gamma \in (0, 1]$ is a scalar. Second, if vehicle's optimal controller is (3.26), we can infer that for some time $t_k \geq t_N$, there are some obstacles detected but *Condition 1* is not violated for all $t \in [t_k, t_k + h]$. Both cases indicate that for all $t \geq t_N$ the following inequality holds in either cases

$$\frac{1}{v_{\max}}\mathbf{u}(t) \cdot \nabla Q_N(\mathbf{x}(t)) \leq -\gamma\|\nabla Q_N(\mathbf{x}(t))\|. \tag{3.31}$$

By the properties of Q_N, we take $Q_N(\mathbf{x}(t))$ as a Lyapunov function. Given (3.31), for all $\mathbf{x}(t) \neq \mathbf{z}$, the following inequality holds

$$\dot{Q}_N(\mathbf{x}(t)) = \mathbf{u}(t) \cdot \nabla Q_N(\mathbf{x}(t)) \leq -v_{\max}\gamma\|\nabla Q_N(\mathbf{x}(t))\| < 0. \tag{3.32}$$

Thus, we infer that for any initial condition $\mathbf{x}(t_0) \in \overline{\Omega}$, $\mathbf{x}(t) \to \mathbf{z}$ as $t \to \infty$.

We now show (b). By the result of Proposition 3.3.1, we know that the cost function (3.24) satisfies

$$J_k(\mathbf{x}(\cdot)) \geq Q_{n(k)}(\mathbf{x}(t_k)). \tag{3.33}$$

By the assumption of (b), we know the trajectory generated by (3.25) is obstacle-free. Thus, by taking $\mathbf{u}(t)$ to be (3.25) and substituting $\mathbf{u}(t)$ into (3.24), we obtain $J_k(\mathbf{x}(\cdot)) = Q_{n(k)}(\mathbf{x}(t_k))$, which completes the proof.

The last step is to show (c). We need to show that the trajectory generated by $\mathbf{u}^*$ in (3.26) satisfies the Euler-Lagrange equations. We first discuss the case when $\mathbf{x}(t) \neq \mathbf{z}$. We adjoin the system differential equation (3.1) to $J_k(\mathbf{x}(\cdot))$ with multiplier $\boldsymbol{\lambda}(t) \in \mathcal{R}^2$ (see e.g. [6], pp. 48):

$$\overline{J}_k(\mathbf{x}(\cdot)) = \min_{\mathbf{u}(\cdot)\in\mathbb{U}}(\int_{t_k}^{t_k+H} g(\mathbf{x}(\tau))\|\mathbf{u}(\tau)\| + \boldsymbol{\lambda}^T(\tau)(\mathbf{u}(\tau) - \dot{\mathbf{x}}(\tau))d\tau + Q_{n(k)}(\mathbf{x}(t_k+H))). \tag{3.34}$$

Denoting the Hamiltonian by

$$\mathcal{H}(\mathbf{x}(t), \mathbf{u}(t)) = g(\mathbf{x}(t))\|\mathbf{u}(t)\| + \boldsymbol{\lambda}^T(t)\mathbf{u}(t), \tag{3.35}$$

we obtain the necessary condition for the optimality:

$$\boldsymbol{\lambda}(t_k+H) = \frac{\partial Q_{n(k)}^T}{\partial \mathbf{x}}(t_k+H), \tag{3.36}$$

$$\dot{\boldsymbol{\lambda}}(t) = -\frac{\partial \mathcal{H}^T}{\partial \mathbf{x}}(t) = -\frac{\partial g^T}{\partial \mathbf{x}}(t)\|\mathbf{u}(t)\|, \tag{3.37}$$

If $\|\mathbf{u}\| \neq 0$,

$$\frac{\partial \mathcal{H}^T}{\partial \mathbf{u}}(t) = \frac{g(\mathbf{x}(t))\mathbf{u}(t)}{\sqrt{\mathbf{u}^T(t)\mathbf{u}(t)}} + \boldsymbol{\lambda}(t) = \mathbf{0}. \tag{3.38}$$

Substituting (3.23) and (3.26) into (3.37) yields

$$\begin{aligned}\dot{\boldsymbol{\lambda}}(t) &= -\frac{\partial g^T(\mathbf{x}(t))}{\partial \mathbf{x}}(t)\|\mathbf{u}(t)\| = -\frac{\partial \|\nabla Q_k^*(\mathbf{x}(t))\|}{\partial \mathbf{x}(t)} v_{\max} \\ &= -\frac{\partial^2 Q_k^*}{\partial \mathbf{x}^2}\frac{v_{\max}\nabla Q_k^*(\mathbf{x}(t))}{\|\nabla Q_k^*(\mathbf{x}(t))\|} = -\frac{\partial^2 Q_k^*}{\partial \mathbf{x}^2}\mathbf{u}(t).\end{aligned} \tag{3.39}$$

Given the boundary condition (3.36) and given *Condition 2*, integrating the above equation with respect to t yields

$$\boldsymbol{\lambda}(t) = -\frac{\partial Q_k^{*T}}{\partial \mathbf{x}}(t) \tag{3.40}$$

for $t \in [t_k, t_k+H]$. Substituting (3.40) and (3.26) into the left side of (3.38) yields

$$\begin{aligned}&\frac{g(\mathbf{x}(t))\mathbf{u}(t)}{\sqrt{\mathbf{u}^T(t)\mathbf{u}(t)}} + \boldsymbol{\lambda}(t) = \frac{v_{\max} g(\mathbf{x}(t))\nabla Q_k^*(\mathbf{x}(t))}{v_{\max}\|\nabla Q_k^*(\mathbf{x}(t))\|} - \frac{\partial Q_k^{*T}}{\partial \mathbf{x}}(t) \\ =&\frac{\partial Q_k^{*T}}{\partial \mathbf{x}}(t) - \frac{\partial Q_k^{*T}}{\partial \mathbf{x}}(t) = \mathbf{0}.\end{aligned} \tag{3.41}$$

Therefore, we can conclude that the equality of (3.38) holds.

If $\mathbf{z}$ is on the trajectory (3.26), since $\mathbf{u}(t)$ is discontinuous, we treat $\mathbf{z}$ as a corner (see e.g. [6] pp. 125 and [20] pp. 61). Assume that $\mathbf{x}$ reaches $\mathbf{z}$ at time $t_k + c$, where $c \leq H$. We adjoin the system differential equation (3.1) to $J_k(\mathbf{x}(\cdot))$ with multiplier $\boldsymbol{\lambda}(t) \in \mathcal{R}^2$:

$$\begin{aligned}\overline{J}_k(\mathbf{x}(\cdot)) = \min_{\mathbf{u}(\cdot)\in\mathbb{U}} (&\int_{t_k}^{t_k+c-0} g(\mathbf{x}(\tau))\|\mathbf{u}(\tau)\| + \boldsymbol{\lambda}^T(\tau)(\mathbf{u}(\tau) - \dot{\mathbf{x}}(\tau))d\tau \\ &+ \int_{t_k+c+0}^{t_k+H} g(\mathbf{x}(\tau))\|\mathbf{u}(\tau)\| + \boldsymbol{\lambda}^T(\tau)(\mathbf{u}(\tau) - \dot{\mathbf{x}}(\tau))d\tau \\ &+ Q_{n(k)}(\mathbf{x}(t_k + H))).\end{aligned} \tag{3.42}$$

Since the second integrand and $Q_{n(k)}(\mathbf{x}(t + H))$ are both zero, the above equation becomes

$$\overline{J}_k(\mathbf{x}(\cdot)) = \min_{\mathbf{u}(\cdot)\in\mathbb{U}} \int_{t_k}^{t_k+c-0} g_k(\mathbf{x}(\tau))\|\mathbf{u}(\tau)\| + \boldsymbol{\lambda}^T(\tau)(\mathbf{u}(\tau) - \dot{\mathbf{x}}(\tau))d\tau. \tag{3.43}$$

Thus, $\delta\overline{J}_k$ becomes

$$\begin{aligned}\delta\overline{J}_k =&[(\mathcal{H} - \boldsymbol{\lambda}^T\dot{\mathbf{x}})\delta c]_{t=t_k+c-0} + \int_{t_k}^{t_k+c-0} \frac{\partial\mathcal{H}}{\partial\mathbf{u}}\delta\mathbf{u} + \frac{\partial\mathcal{H}}{\partial\mathbf{x}}\delta\mathbf{x} - \boldsymbol{\lambda}^T\delta\dot{\mathbf{x}}d\tau \\ =&[(\mathcal{H} - \boldsymbol{\lambda}^T\dot{\mathbf{x}})\delta c - \boldsymbol{\lambda}^T\delta\mathbf{x}]_{t=t_k+c-0} + [\boldsymbol{\lambda}^T\delta\mathbf{x}]_{t=t_k} + \int_{t_k}^{t_k+c-0} \frac{\partial\mathcal{H}}{\partial\mathbf{u}}\delta\mathbf{u} + (\frac{\partial\mathcal{H}}{\partial\mathbf{x}} + \dot{\boldsymbol{\lambda}}^T)\delta\mathbf{x}d\tau.\end{aligned} \tag{3.44}$$

Thus, the necessary condition for optimality is

$$\begin{aligned}&\boldsymbol{\lambda}|_{t_k+c-0} = \mathbf{0}, \\ &\frac{\partial\mathcal{H}^T}{\partial\mathbf{x}}(t) + \dot{\boldsymbol{\lambda}}(t) = \mathbf{0}, \\ &(\mathcal{H} - \boldsymbol{\lambda}^T\dot{\mathbf{x}})|_{t_k+c-0} = 0, \\ &\frac{\partial\mathcal{H}^T}{\partial\mathbf{u}}(t) = \mathbf{0}.\end{aligned} \tag{3.45}$$

Let us show that each equation in (3.45) holds. Given the first two equations in (3.45), we solve for $\dot{\boldsymbol{\lambda}}(t)$ and obtain

$$\boldsymbol{\lambda}(t) = -\frac{\partial Q_k^{*T}}{\partial\mathbf{x}}(t). \tag{3.46}$$

We need to check if the last two equations in (3.45) hold. The left side of the third equation in (3.45) becomes

$$\begin{aligned}
&\lim_{t\to t_k+c-0}(\mathcal{H}(t)-\boldsymbol{\lambda}^T\dot{\mathbf{x}}(t))\\
=&\lim_{t\to t_k+c-0}(g(\mathbf{x}(t))\|\mathbf{u}(t)\|+\boldsymbol{\lambda}^T(t)\mathbf{u}(t)-\boldsymbol{\lambda}^T\mathbf{u}(t))\\
=&\lim_{t\to t_k+c-0}g(\mathbf{x}(t))\|\mathbf{u}(t)\|\\
\leq&\lim_{t\to t_k+c-0}g(\mathbf{x}(t))v_{\max}.
\end{aligned} \tag{3.47}$$

Since $g(\mathbf{x}(t))\geq 0$ and is continuous for $t\in[t_k,t_k+c)$, and given that $\lim_{t\to t_k+c-0}\mathbf{x}(t)=\mathbf{z}$, we infer the following inequality

$$0\leq\lim_{t\to t_k+c-0}g(\mathbf{x}(t))\|\mathbf{u}(t)\|\leq\lim_{t\to t_k+c-0}g(\mathbf{x}(t))v_{\max}=0. \tag{3.48}$$

Using the above inequality and (3.47), we can conclude that the third equation in (3.45) holds. Since $\mathbf{u}(t)\neq\mathbf{0}$ on $t\in[t_k,t_k+c)$, substituting (3.26) and (3.46) into the left side of the 4th equation in (3.45) yields

$$\begin{aligned}
&\frac{\partial\mathcal{H}^T}{\partial\mathbf{u}}(t)\\
=&\frac{g(\mathbf{x}(t))\mathbf{u}(t)}{\sqrt{\mathbf{u}^T(t)\mathbf{u}(t)}}+\boldsymbol{\lambda}(t)\\
=&\frac{v_{\max}g(\mathbf{x}(t))\nabla Q_k^*(\mathbf{x}(t))}{v_{\max}\|\nabla Q_k^*(\mathbf{x}(t))\|}-\frac{\partial Q_k^{*T}}{\partial\mathbf{x}}(t)\\
=&\frac{\partial Q_k^{*T}}{\partial\mathbf{x}}(t)-\frac{\partial Q_k^{*T}}{\partial\mathbf{x}}(t)\\
=&\mathbf{0}.
\end{aligned} \tag{3.49}$$

Thus, we can conclude that each equation in (3.45) holds. □

3.5 Computational Expense and Simulation

The computational expense of the proposed methodology is reduced in two important regards in comparison to the algorithm proposed in Section 3.3. First, Theorem 3.4.1(b) indicates

that we can simply identify the optimal trajectory by following the gradient of $Q_{n(k)}$ if there are no obstacles on the path generated by the controller in Equation (3.25). Second, even if there are obstacles on the path generated by (3.25), at first we need only to compute Q_k^* in a local domain. Note that computing the solution of this local problem can be many times less costly than computing the solution of the Eikonal equation over the full domain. In contrast, when either *Condition 1* or *Condition 2* is violated, updating Q_k can be expensive since we have to recompute the level sets for Q_k over the entire domain $\overline{\Omega(k)}$.

To illustrate the principal conclusions in this chapter, we study an example of an autonomous surface vehicle (ASV) navigating in a riverine environment. Figure 3.1 and 3.6 respectively represent the *a priori* map and actual environment, both of which span an area of $800m \times 600m$. To ensure that an ASV can move freely between empty nodes, the map is discretized with grids size of $3m \times 3m$, which is about twice as large as the specific ASV ($3m \times 1.5m$). The vehicle's detection range is $r = 30m$. We set the fixed parameters that characterize the problem to be $h = 4$ *sec*, $H = 6$ *sec*, $v_{\max} = 3$ m/sec and $\gamma = 0.01$. During the entire mission, there are a total number of fourteen instances where new solutions of either global or local Eikonal Equations were calculated. The location at which each RHC optimization problem is solved is marked by small circles, and they are numbered in order as seen in Figure 3.6. Among these calculations, there are thirteen times for which the ASV manages to plan trajectories by computing Q_k^* locally. The only global update occurs at the 14th update event when the ASV enters and detects a U-shaped trap shown in the upper right corner of Figure 3.6. In the end, Figure 3.6 shows that the vehicle eventually reaches the goal.

To focus on the process as for how the update of map information evolves and how the corresponding trajectory changes, we show a sequence of RHC calculations between the 7th and the last instances of RHC calculations in Figure 3.7-3.9. In these figures, the shaded areas are the corresponding unexpected obstacles that are detected, and the black dash circles represent the planning horizon H. For the 7th to 13th updates, since the algorithm finds paths that satisfy both matching conditions, the solutions to the global Eikonal equations

do not need to be updated as shown in Figure 3.10. In Figure 3.9, since the vehicle detects the dead end, and since the algorithm can not find a path that satisfies one of the matching conditions (*Condition 1*), the new level set Q_k is updated globally. A closeup of the new global level set Q_k for the corresponding area is shown in Figure 3.11. After globally updating Q_k, the vehicle manages to find paths over the next several horizons such that it escapes the dead end.

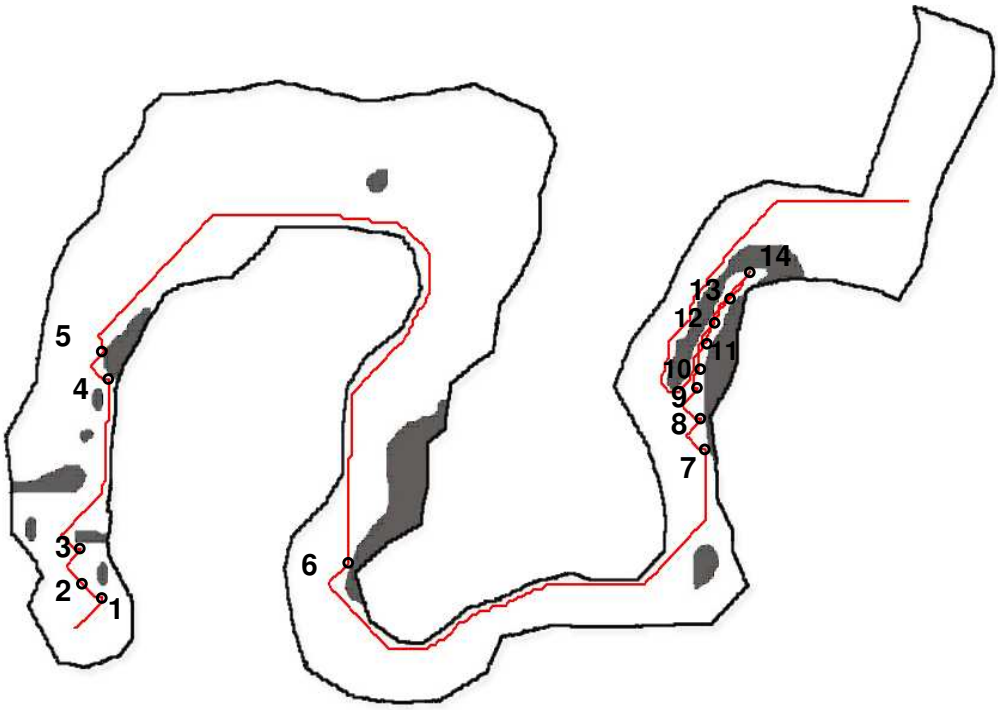

Figure 3.6: The gray areas are some unexpect obstacles due to the inaccuracy and incompleteness of the *a priori* map in Figure 3.1. The red line is the actual course of the ASV.

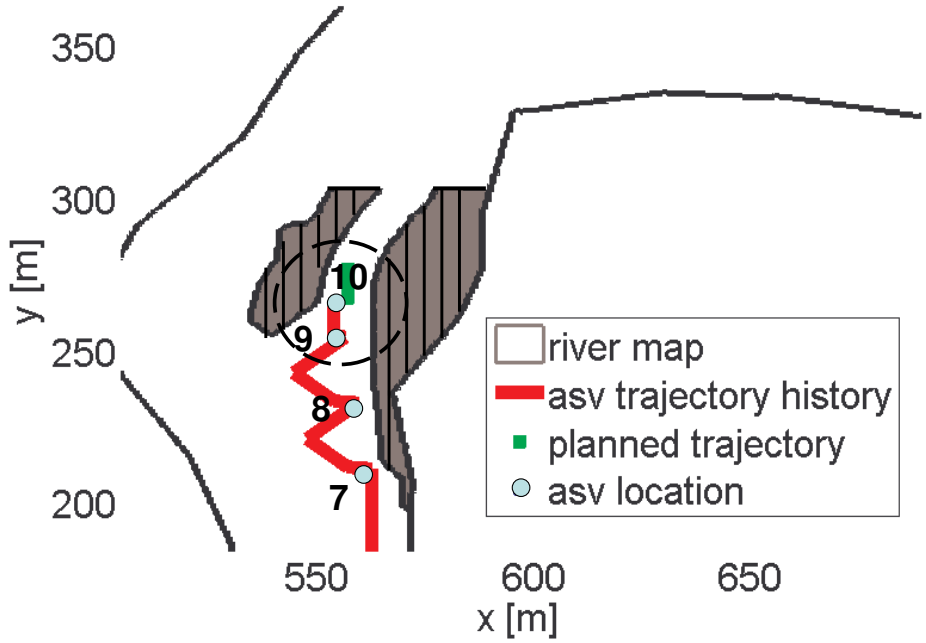

Figure 3.7: The ASV's trajectory history and planned trajectory after the 10th RHC replanning. The shaded area is the detected unexpected obstacle between the 7th and the 10th replannings.

3.6 Concluding Remarks

We have proposed a hybrid RHC method that incorporates the solutions of certain Eikonal equations as a terminal cost. We show that the RHC formulation with this selection of the terminal cost is asymptotically stable using the results of [48] in the case that the domain is completely known. In addition, we derive sufficient conditions for the convergence of the method to the target. The convergence proof relies on the definition of the matching conditions that determine when and if a new global solution of the Eikonal equations should be incorporated as a terminal cost in the RHC formulation. The computational expense is reduced by searching for paths locally if possible.

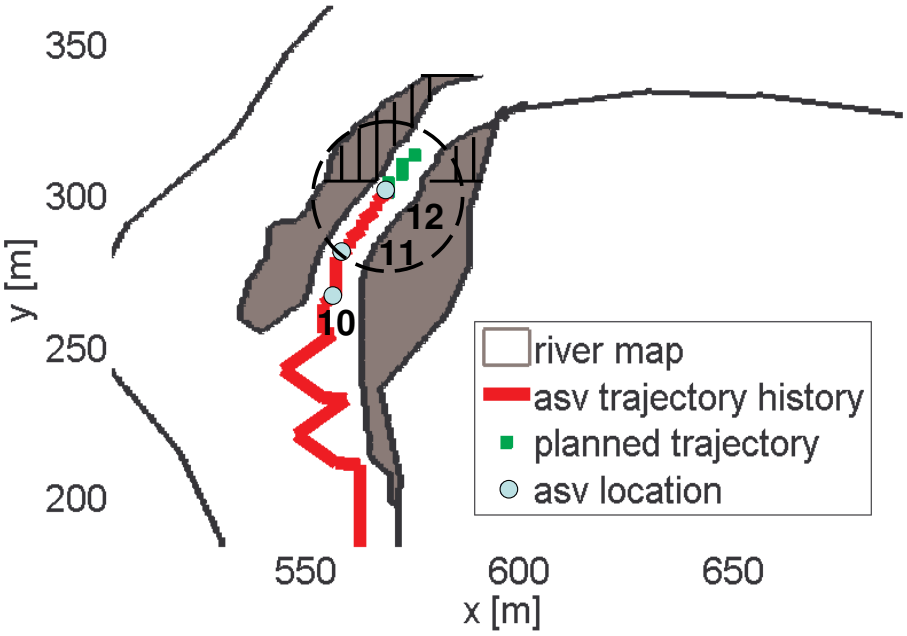

Figure 3.8: The ASV's trajectory history and planned trajectory after the 12th RHC replanning. The shaded area is the detected unexpected obstacle between the 11th and the 12th replannings.

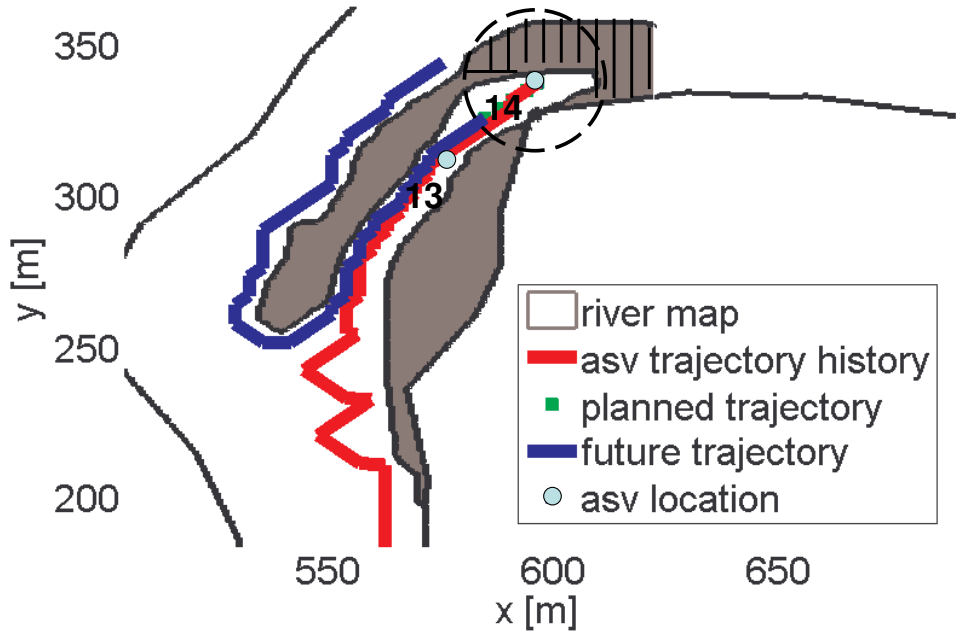

Figure 3.9: The ASV's trajectory history and planned trajectory in the next few horizons. The shaded area is the detected unexpected obstacle between the 13th and the 14th replannings.

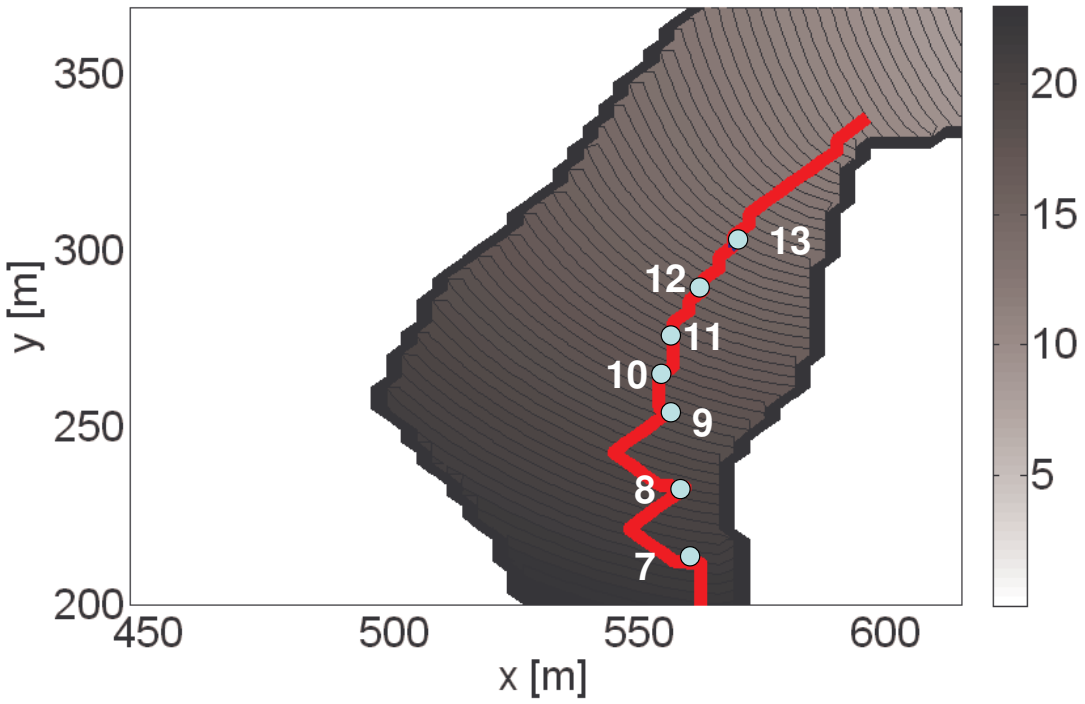

Figure 3.10: The closeup of the old global level set used as the terminal cost between the 7th and the 13th RHC replannings.

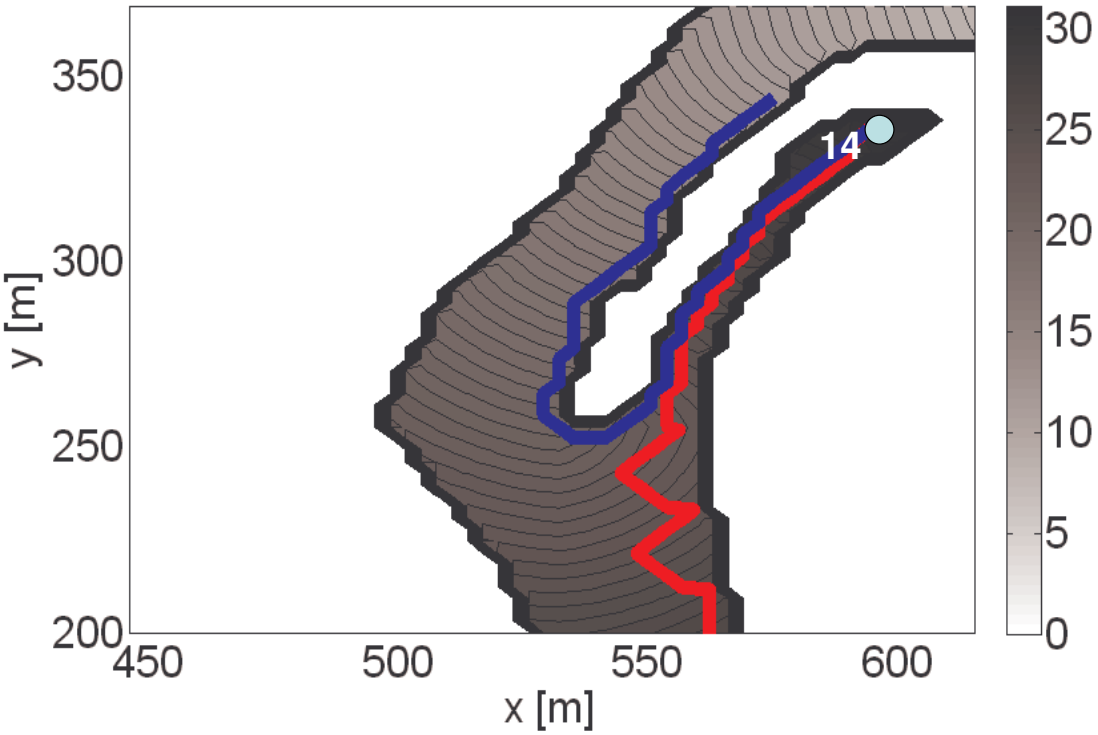

Figure 3.11: The closeup of the new global level set recalculated at the 14th RHC replanning.

Chapter 4

Receding Horizon Control Method in the Presence of Moving Obstacles

In this chapter, we address the minimal risk motion planning problem in a two dimensional environment in the presence of both moving and static obstacles. Our approach is inspired by recent results due to Vladimirsky [78] in which path planning on time-varying maps is addressed using a new level-set approach, and for which computational costs are remarkably low. Toward practical implementation of these results for path planning in unstructured environments, we develop a receding-horizon formulation in which path planning for moving and static obstacles is addressed locally, while path planning for static obstacles is addressed globally. This formulation reduces the overall computational burden of path planning and makes it suitable for very large domains. The result is a suboptimal receding horizon planner and a matching condition that connects local planning with global planning. We present a rigorous analysis from which convergence to a desired endpoint is guaranteed.

The chapter is organized as follows. In Section 4.1, we introduce the formulation of the minimal risk path planning problem in the presence of moving obstacles. In Section 4.2, we employ the Euler-Lagrange equation to derive the governing PDE and find a suboptimal trajectory for the minimal risk problem. Then, we propose, in Section 4.3, a receding horizon

controller that plans sub-optimal paths locally when moving obstacles are detected during the mission. We show simulation results in Section 4.4, and the last section is left for conclusions.

4.1 Problem Formulation

The same as in Section 3.1, we denote the environment by $\overline{\Omega}$, where Ω is a connected and bounded open set in $\mathcal{R}^2$ and $\overline{\Omega}$ is the closure of Ω. The task for the vehicle is to avoid collisions with both moving and static obstacles as well as to reach a predefined goal $\mathbf{z} \in \overline{\Omega}$. To model both static and moving obstacles in $\overline{\Omega}$, we associate a risk for the vehicle to traverse a point at any time by a cost function $g^* \in C^1(\overline{\Omega} \times [t_0, \infty); \mathcal{R}^1)$. For a point $\boldsymbol{\xi} \in \overline{\Omega}$, $g^*(\boldsymbol{\xi}, t)$ remains constant when there are no moving obstacles close to $\boldsymbol{\xi}$ at time t. It increases when $\boldsymbol{\xi}$ is occupied by or very close to a moving obstacle and decreases to the constant value after the moving obstacle leaves. In addition, we assume that there exist two positive scalars G_1 and G_2 such that

$$0 < G_1 \leq g^*(\boldsymbol{\xi}, t) \leq G_2 < \infty, \ \forall \boldsymbol{\xi} \in \overline{\Omega}, \ \forall t \in [t_0, \infty). \tag{4.1}$$

To ensure that the minimal risk problem is well-posed, we only search for optimal controllers in a given time interval $[t_0, t_1]$ where $t_0 \leq t_1 < \infty$. Our goal is to find $\mathbf{u}(\cdot)$ that minimizes the cumulative minimal cost from $\mathbf{x}_0$ to $\boldsymbol{\xi} \in \overline{\Omega}$ satisfying,

$$J(\boldsymbol{\xi}) = \min_{\mathbf{u}(\cdot) \in \mathbb{U}} \int_{t_0}^{T} g^*(\mathbf{x}(\tau), \tau) \|\dot{\mathbf{x}}(\tau)\| d\tau \tag{4.2}$$

subject to

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{u}(t), \\ \mathbf{x}(t) \in \overline{\Omega}, \\ \mathbf{x}(t_0) = \mathbf{x}_0, \\ \mathbf{x}(T) = \boldsymbol{\xi}, \\ \mathbf{u}(t) \in \mathbb{U} := \left\{ \mathbf{u} \in \mathcal{R}^2 : \ \|\mathbf{u}\| \leq v_{\max} \right\}, \\ T \in [t_0, t_1], \end{cases} \tag{4.3}$$

where $v_{\max}$ is the maximum speed of the autonomous vehicle. In particular, when $\boldsymbol{\xi} = \mathbf{z}$, we obtain the optimal controller $\mathbf{u}(\cdot)$ to maneuver the vehicle from the initial location $\mathbf{x}_0$ to the goal $\mathbf{z}$.

4.2 Suboptimal Solution

Proposition 4.2.1 defines a necessary condition that minimizes the cost function (4.2). Without loss of generality, we let $t_0 = 0$.

Proposition 4.2.1. *Consider the optimization problem (4.2) that is subject to the dynamics (4.3). Suppose that function $Q^* \in C^2(\overline{\Omega}; \mathcal{R}^1)$ satisfies*

$$\begin{aligned} &\|\nabla Q^*(\boldsymbol{\xi})\| = g^* \left(\boldsymbol{\xi}, \frac{Q^*(\boldsymbol{\xi})}{\gamma G_1 v_{\max}} \right), \\ &Q^*(\mathbf{x}(0)) = 0, \end{aligned} \tag{4.4}$$

where $\gamma \in (0, 1]$ is a constant scalar. If $t_1 \geq \max_{\boldsymbol{\xi} \in \overline{\Omega}} \frac{Q^*(\boldsymbol{\xi})}{\gamma G_1 v_{\max}}$, the controller that locally minimizes (4.2) is characterized by

$$\mathbf{u}(t) = \gamma \frac{G_1 v_{\max}}{g^* \left(\mathbf{x}(t), \frac{Q^*(\mathbf{x}(t))}{\gamma G_1 v_{\max}} \right)} \frac{\nabla Q^*(\mathbf{x}(t))}{\|\nabla Q^*(\mathbf{x}(t))\|}. \tag{4.5}$$

Along the trajectory generated by the controller (4.5), the following equation holds

$$g^* \left(\mathbf{x}(t), \frac{Q^*(\mathbf{x}(t))}{\gamma G_1 v_{\max}} \right) = g^*(\mathbf{x}(t), t). \tag{4.6}$$

In addition,

$$J(\mathbf{z}) = Q^*(\mathbf{z}) \tag{4.7}$$

is the suboptimal cost that is achieved using the control defined by (4.5).

The partial differential equation (4.4) is the same as that derived in [78]. In [78], the author employs the Hamilton-Jacobi-Bellman equation and finds sufficient conditions for the global minimal-time problem. The author also establishes existence and uniqueness of the viscosity solution of the partial differential equation (4.4).

Remark 4.2.2. *Note that unlike the Eikonal equation (3.4), which computes the minimal risk paths from a point $\boldsymbol{\xi} \in \overline{\Omega}$ to the goal $\mathbf{z}$, the PDE (4.4) proposed herein encodes the minimal paths to travel from the initial position of the vehicle $\mathbf{x}(0)$ to a point $\boldsymbol{\xi} \in \overline{\Omega}$. This condition is required because the minimal risk path in the presence of moving obstacles is dependent on the time when the vehicle crosses a point in the environment. Thus, we have to start computing the trajectory of the autonomous vehicle from its initial position so that we know exactly when the vehicle will traverse a point in the environment.*

Remark 4.2.3. *Equations (4.4) and (4.5) indicate that the suboptimal controller can be found by going along the gradient of value $Q^*(\boldsymbol{\xi})$ whose domain is the original two dimensional environment $\overline{\Omega}$. Therefore, to search the suboptimal controller, we need only to compute $Q^*(\boldsymbol{\xi})$ over the original environmental domain $\overline{\Omega}$.*

Remark 4.2.4. *For the case in which there are no moving obstacles, the cost function $g^*(\boldsymbol{\xi}, t)$ depends only on its first argument. Thus, the partial differential equation (4.4) reduces to the well-known Eikonal equation [64]. Then, the resulting optimal controller by (4.5) coincides with that proposed in [36].*

Remark 4.2.5. *Equations (4.5) and (4.6) show that the speed of the vehicle is $\gamma \frac{G_1 v_{\max}}{g(\mathbf{x}(t),t)}$. This indicates that the higher the risk for traversal is, the slower is the speed of the vehicle. This property captures a desirable characteristic of the vehicle motion. That is, the vehicle moves slower over locations determined to have higher risk.*

Proof of Proposition 4.2.1. In this proof, we will use the fact that along the trajectory generated by the controller (4.5), the equation (4.6) holds. Therefore, the first step is to establish a connection between $Q^*(\mathbf{x}(t))$ and t. Since $Q^*(\boldsymbol{\xi})$ is a conservative vector field, we have

$$\begin{aligned}
&Q^*(\mathbf{x}(t)) \\
=&\int_{\mathbf{x}(0)}^{\mathbf{x}(t)} \frac{\partial Q^*}{\partial \mathbf{x}} d\mathbf{x} \\
=&\int_0^t \frac{\partial Q^*(\mathbf{x}(\tau))}{\partial \mathbf{x}(\tau)} \dot{\mathbf{x}}(\tau) d\tau \\
=&\int_0^t \frac{\partial Q^*(\mathbf{x}(\tau))}{\partial \mathbf{x}(\tau)} \frac{\gamma G_1 v_{\max}}{g^*\left(\mathbf{x}(\tau), \frac{Q^*(\mathbf{x}(\tau))}{\gamma G_1 v_{\max}}\right)} \frac{\nabla Q^*(\mathbf{x}(\tau))}{g^*\left(\mathbf{x}(\tau), \frac{Q^*(\mathbf{x}(\tau))}{\gamma G_1 v_{\max}}\right)} d\tau \\
=&\int_0^t \gamma G_1 v_{\max} d\tau \\
=&\gamma G_1 v_{\max} t
\end{aligned} \tag{4.8}$$

Thus, we infer that

$$t = \frac{Q^*(\mathbf{x}(t))}{\gamma G_1 v_{\max}}, \tag{4.9}$$

which indicates (4.6).

We use the Euler-Lagrange method to find the necessary condition for the optimal path. We adjoin the system differential equation (4.3) to $J(\boldsymbol{\xi})$ with multiplier $\boldsymbol{\lambda}(\tau) \in \mathcal{R}^2$ (see e.g. [6], pp. 72):

$$\overline{J} = \int_{t_0}^{T} g^*(\mathbf{x}(\tau), \tau) \|\dot{\mathbf{x}}(\tau)\| + \boldsymbol{\lambda}^T(\tau)(\mathbf{u} - \dot{\mathbf{x}}(\tau)) d\tau. \tag{4.10}$$

Since $\|\mathbf{u}\| \neq 0$, we perturb $\overline{J}$ with variations to $\delta\mathbf{x}$, $\dot{\delta}\mathbf{x}$, $\delta\mathbf{u}$ and δT. The variations in $\overline{J}$

satisfy

$$\begin{aligned}
\delta\overline{J} &= \int_{t_0}^{T} \left[\frac{\partial g^*}{\partial \mathbf{x}} \delta\mathbf{x} \|\mathbf{u}\| + g^* \frac{\mathbf{u}^T}{\|\mathbf{u}\|} \delta\mathbf{u} + \boldsymbol{\lambda}^T \delta\mathbf{u} - \boldsymbol{\lambda}^T \delta\dot{\mathbf{x}} \right] d\tau \\
&\quad + g^*(\mathbf{x}(T), T) \|\mathbf{u}(T)\| \delta T \\
&= \int_{t_0}^{T} \left[\frac{\partial g^*}{\partial \mathbf{x}} \delta\mathbf{x} \|\mathbf{u}\| + g^* \frac{\mathbf{u}^T}{\|\mathbf{u}\|} \delta\mathbf{u} + \boldsymbol{\lambda}^T \delta\mathbf{u} + \dot{\boldsymbol{\lambda}}^T \delta\mathbf{x} \right] d\tau \\
&\quad - \boldsymbol{\lambda}^T \delta\mathbf{x}|_{t_0}^{T} + g^*(\mathbf{x}(T), T) \|\mathbf{u}(T)\| \delta T \\
&= \int_{t_0}^{T} \left[(\frac{\partial g^*}{\partial \mathbf{x}} \|\mathbf{u}\| + \dot{\boldsymbol{\lambda}}^T) \delta\mathbf{x} + (g^* \frac{\mathbf{u}^T}{\|\mathbf{u}\|} + \boldsymbol{\lambda}^T) \delta\mathbf{u} \right] d\tau \\
&\quad - \boldsymbol{\lambda}^T \underbrace{\delta\mathbf{x}}_{\delta\mathbf{x} = d\mathbf{x} - \dot{\mathbf{x}} dt} |_T + g^*(\mathbf{x}(T), T) \|\mathbf{u}(T)\| \delta T \\
&= \int_{t_0}^{T} \left[(\frac{\partial g^*}{\partial \mathbf{x}} \|\mathbf{u}\| + \dot{\boldsymbol{\lambda}}^T) \delta\mathbf{x} + (g^* \frac{\mathbf{u}^T}{\|\mathbf{u}\|} + \boldsymbol{\lambda}^T) \delta\mathbf{u} \right] d\tau \\
&\quad - \boldsymbol{\lambda}^T (d\mathbf{x} - \dot{\mathbf{x}} \delta T)|_T + g^*(\mathbf{x}(T), T) \|\mathbf{u}(T)\| \delta T.
\end{aligned} \tag{4.11}$$

Each of the coefficients multiplying the variations must vanish. Thus, we have

$$\begin{aligned}
\delta\overline{J} = \int_{t_0}^{T} \Bigg[&(\underbrace{\frac{\partial g^*}{\partial \mathbf{x}} \|\mathbf{u}\| + \dot{\boldsymbol{\lambda}}^T}_{=\mathbf{0}^T}) \delta\mathbf{x} + (\underbrace{g^* \frac{\mathbf{u}^T}{\|\mathbf{u}\|} + \boldsymbol{\lambda}^T}_{=\mathbf{0}^T}) \delta\mathbf{u} \Bigg] d\tau \\
&- \underbrace{\boldsymbol{\lambda}^T d\mathbf{x}|_T}_{=0} + (\underbrace{g^*(\mathbf{x}(T), T) \|\mathbf{u}(T)\| + \boldsymbol{\lambda}^T(T) \dot{\mathbf{x}}(T)}_{=0}) \delta T.
\end{aligned} \tag{4.12}$$

Note that the terminal constraint is that $\mathbf{x}(T) = \boldsymbol{\xi}$. Thus, we infer that

$$d\mathbf{x}|_T = [0, 0]^T. \tag{4.13}$$

Therefore, from (4.12), we obtain a set of Euler-Lagrange equations and the boundary condition as the following for all $t \in [0, T]$:

$$\frac{\partial g^*(\mathbf{x}(t,t)}{\partial \mathbf{x}(t)} \|\mathbf{u}(t)\| + \dot{\boldsymbol{\lambda}}^T(t) = \mathbf{0}^T, \tag{4.14}$$

$$g^*(\mathbf{x}(t), t) \frac{\mathbf{u}^T(t)}{\|\mathbf{u}(t)\|} + \boldsymbol{\lambda}^T(t) = \mathbf{0}^T, \tag{4.15}$$

and

$$g^*(\mathbf{x}(T), T)\|\mathbf{u}(T)\| + \boldsymbol{\lambda}^T(T)\dot{\mathbf{x}}(T) = 0. \tag{4.16}$$

We now show that the solution of $\boldsymbol{\lambda}(t)$ satisfies

$$\boldsymbol{\lambda}(t) = -\nabla Q^*(\mathbf{x}(t)). \tag{4.17}$$

To do so, together with (4.5), we show that the equations (4.14), (4.15), and (4.16) hold. Substituting (4.5), (4.4), (4.6), and (4.17) into the left side of (4.15), we obtain

$$\begin{aligned} & g^*(\mathbf{x}(t), t)\frac{\mathbf{u}(t)}{\|\mathbf{u}(t)\|} + \boldsymbol{\lambda}(t) \\ =& g^*(\mathbf{x}(t), t)\frac{\nabla Q^*(\mathbf{x})}{\|\nabla Q^*(\mathbf{x})\|} - \nabla Q^*(\mathbf{x}(t)) \\ =& [0, 0]^T. \end{aligned} \tag{4.18}$$

Thus, Equation (4.15) holds. Substituting (4.5), (4.4), (4.6), and (4.17) into the left side of (4.16), we derive

$$\begin{aligned} & g^*(\mathbf{x}(T), T)\|\mathbf{u}(T)\| + \boldsymbol{\lambda}^T(T)\dot{\mathbf{x}}(T) \\ =& g^*(\mathbf{x}(T), T)\frac{\gamma G_1 v_{\max}}{g^*\left(\mathbf{x}(T), \frac{Q^*(\mathbf{x}(T))}{\gamma G_1 v_{\max}}\right)} \\ & - \nabla Q^*(\mathbf{x}(T))^T\frac{\gamma G_1 v_{\max}}{g^*\left(\mathbf{x}(T), \frac{Q^*(\mathbf{x}(T))}{\gamma G_1 v_{\max}}\right)}\frac{\nabla Q^*(\mathbf{x}(T))}{\|\nabla Q^*(\mathbf{x}(T))\|} \\ =& \gamma G_1 v_{\max} - \gamma G_1 v_{\max} \\ =& 0, \end{aligned} \tag{4.19}$$

which indicates that (4.16) holds. The equation (4.14) is left to be proven. Given (4.4) and (4.6), we have

$$\frac{\partial g^*(\mathbf{x}(t), t)^T}{\partial \mathbf{x}} = \frac{\partial \|\nabla Q^*(\mathbf{x}(t))\|^T}{\partial \mathbf{x}} = \frac{1}{g^*}\frac{\partial^2 Q^*}{\partial \mathbf{x}^2}\frac{\partial Q^{*T}}{\partial \mathbf{x}}. \tag{4.20}$$

From (4.5) and (4.17), $\dot{\boldsymbol{\lambda}}$ satisfies

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial^2 Q^*}{\partial \mathbf{x}^2}\dot{\mathbf{x}} = -\frac{\partial^2 Q^*}{\partial \mathbf{x}^2}\frac{\gamma G_1 v_{\max}}{g^*\left(\mathbf{x}, \frac{Q^*(\mathbf{x})}{\gamma G_1 v_{\max}}\right)}\frac{\nabla Q^*(\mathbf{x})}{\|\nabla Q^*(\mathbf{x})\|}. \tag{4.21}$$

Substituting (4.5), (4.4), (4.20), and (4.21) into the left side of (4.14), we obtain

$$\begin{aligned}
&\frac{\partial g^*(\mathbf{x}(t),t)^T}{\partial \mathbf{x}(t)}\|\mathbf{u}(t)\| + \dot{\boldsymbol{\lambda}}(t) \\
=&\frac{1}{g^*\left(\mathbf{x}(t), \frac{Q^*(\mathbf{x}(t))}{\gamma G_1 v_{\max}}\right)}\frac{\partial^2 Q^*}{\partial \mathbf{x}^2}\frac{\partial Q^{*T}}{\partial \mathbf{x}}\frac{\gamma G_1 v_{\max}}{g^*\left(\mathbf{x}(t), \frac{Q^*(\mathbf{x}(t))}{\gamma G_1 v_{\max}}\right)} \\
&-\frac{\partial^2 Q^*}{\partial \mathbf{x}^2}\frac{\gamma G_1 v_{\max}}{g^*\left(\mathbf{x}, \frac{Q^*(\mathbf{x})}{\gamma G_1 v_{\max}}\right)}\frac{\nabla Q^*(\mathbf{x})}{\|\nabla Q^*(\mathbf{x})\|} \\
=&[0,0]^T.
\end{aligned} \tag{4.22}$$

which shows that Equation (4.14) holds. Since all of the three Euler-Lagrange equations (4.14), (4.15), and (4.16) hold, we prove that $\boldsymbol{\lambda}(t)$ satisfies (4.17) and that (4.5) satisfies the necessary condition to minimize (4.2).

Finally, to show (4.7), we substitute (4.5) and (4.6) into (4.2) and obtain

$$\begin{aligned}
J(\mathbf{z}) &= \int_0^T g^*(\mathbf{x}(\tau),\tau)\|\dot{\mathbf{x}}(\tau)\|d\tau \\
&= \int_0^T g^*(\mathbf{x}(\tau),\tau)\frac{\gamma G_1 v_{\max}}{g^*\left(\mathbf{x}(\tau), \frac{Q^*(\mathbf{x}(\tau))}{\gamma G_1 v_{\max}}\right)}d\tau \\
&= \gamma G_1 v_{\max} T.
\end{aligned} \tag{4.23}$$

From (4.9), we conclude

$$J(\mathbf{z}) = \gamma G_1 v_{\max}\frac{Q^*(\mathbf{z})}{\gamma G_1 v_{\max}} = Q^*(\mathbf{z}), \tag{4.24}$$

which completes our proof. □

4.3 Receding Horizon Control Formulation when the State of the Moving Obstacles Detected in Mission

In more realistic scenarios, the vehicle often detects the moving obstacles in real-time by an on-board sensor with limited range. Note that a static obstacle can be viewed as a

moving obstacle with speed zero. In this section, we introduce a receding horizon control formulation (see, e.g. [23]) that addresses the case in which moving and static obstacles within the vehicle's field of view are detected in real-time. We first establish the necessary condition to optimize the cost of the proposed receding horizon controller. We show that by solving a PDE over a local domain centered at the vehicle's current position, we can find a suboptimal controller minimizing the proposed cost functional over the planning horizon. Thus, we can forgo the computation of the solution for the PDE defined in (4.4) at each iteration over the entire domain, even if we detect new moving obstacles. Then, we propose a sufficient condition that guarantees that the vehicle converges to the goal using a sequence of local plans. We show that we can apply the methodology proposed in [48] and [29] to justify this sufficient condition.

To make provision for the new moving and static obstacles detected in the mission, we employ $g^*(\cdot,\cdot)$ as defined in Section 4.1. We assume that between any two consecutive planning horizons $[t_k, t_k + H]$ and $[t_{k+1}, t_{k+1} + H]$, the detection range of the on-board sensors is sufficiently large with respect to the possible maximum travel distance for the autonomous vehicle over the horizon H. The new environmental changes only occur over the non-overlapped duration $[t_k + H, t_{k+1} + H]$. Therefore, for any two consecutive planning horizons, the $g^*(\boldsymbol{\xi}, t)$ values for the overlapped interval $[t_{k+1}, t_k + H]$ is the same whereas the newly detected environmental change information is incorporated in $g^*(\boldsymbol{\xi}, t)$ for the duration of $[t_k + H, t_{k+1} + H]$. In addition, the same as in Section 4.1, we assume that $g^*(\boldsymbol{\xi}, t) \in \overline{\Omega} \times [t_0, \infty) \rightarrow \mathcal{R}^+$ despite the fact that the changes of the environment are detected discretely with respect to time.

Before we introduce the receding horizon control formulation, we introduce the Eikonal equation satisfying

$$\|\nabla Q(\boldsymbol{\xi})\| = g(\boldsymbol{\xi}),\ Q(\mathbf{z}) = 0. \tag{4.25}$$

where $g \in C^1(\overline{\Omega}; \mathcal{R}^+)$ corresponds to the risk of traversal for the *a priori* static environment. In our approach, the solution Q is computed over the entire domain $\overline{\Omega}$. As we have discussed

in Chapter 3, the solution $Q(\boldsymbol{\xi})$, in contrast to $Q^*(\cdot)$ defined in (4.4), encodes the minimal risk path from $\boldsymbol{\xi}$ to the goal $\mathbf{z}$ in a static environment. Since for a given location, the risk to traverse will increase if a moving obstacle is passing through or if a new static obstacle is detected, we assume

$$0 < G_1 \leq g(\boldsymbol{\xi}) \leq g^*(\boldsymbol{\xi}, t) \leq G_2, \ \forall \ (\boldsymbol{\xi}, t) \in \overline{\Omega} \times [t_0, \infty).$$

In the receding horizon control formulation that is proposed herein, the function Q serves as the terminal cost in the local cost functional. Indeed, in Proposition 4.3.1, we compute a path in a local region around the vehicle using the results in Section 4.2. This local path will account for moving obstacles. We also compute a global path as in (4.25) that accounts for only static obstacles. When computing local paths, it is not immediately obvious how to choose the end-point of the path since a local region may not include the desired goal location. It is also important to show that a sequence of local paths does eventually lead to the desired goal location. The level set Q that is computed for the global path is used to address both of these challenges.

If the sensor's sampling period is h, we often let the implementation horizon be h as well. We choose H as the planning horizon where $H \geq h$. Note that the receding horizon control formulation must make provision for any newly detected moving obstacles. We aim to minimize the risk for traversal over the planning horizon as well as the expected minimal risk for traversal from the terminal state $\mathbf{x}(t_k + H)$ toward the goal. The receding horizon formulation then seeks to find the local optimal control and state pair on $[t_k, t_k + H]$ that solves the minimization problem

$$J(\mathbf{x}(\cdot), t_k) = \min_{\mathbf{u}(\cdot) \in \mathbb{U}} \int_{t_k}^{t_k + H} g^*(\mathbf{x}(\tau), \tau) \|\dot{\mathbf{x}}(\tau)\| d\tau + Q(\mathbf{x}(t_k + H)) \tag{4.26}$$

subject to

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{u}(t), \\ \mathbf{x}(t_0) = \mathbf{x}_0, \\ \mathbf{x}(t_k) = \mathbf{x}(t_{k-1} + h), \\ \mathbf{x}(t) \in \overline{\Omega}. \end{cases} \tag{4.27}$$

for $k = 0, 1, 2, \ldots, \infty$. The overall planned trajectory and control on $[t_0, \infty)$ are spliced together from the locally optimal trajectory in the usual way (see, e.g. [23]). The following proposition defines a necessary condition to minimize (4.26).

Proposition 4.3.1. *Consider the optimization problem (4.26) that is subject to the dynamics (4.27), and a function $Q_k \in C^2(\overline{\Omega}; \mathcal{R}^1)$ satisfying*

$$\begin{aligned} &\|\nabla Q_k^*(\boldsymbol{\xi})\| = g^* \left(\boldsymbol{\xi}, \frac{Q_k^*(\boldsymbol{\xi})}{\gamma G_1 v_{\max}} + t_k \right), \\ &Q_k^*(\mathbf{x}(t_k)) = 0. \end{aligned} \tag{4.28}$$

where $\gamma \in (0, 1]$ is a constant scalar. If the matching condition

$$\nabla Q_k^*(\mathbf{x}(t_k + H)) = -\nabla Q(\mathbf{x}(t_k + H)), \quad \text{if } \mathbf{x}(t_k + H) \neq \mathbf{z}, \tag{4.29}$$

is satisfied, the controller that locally minimizes (4.26) satisfies

$$\mathbf{u}(t) = \begin{cases} \gamma \frac{G_1 v_{\max}}{g^* \left(\mathbf{x}(t), \frac{Q_k^*(\mathbf{x}(t))}{\gamma G_1 v_{\max}} + t_k \right)} \frac{\nabla Q_k^*(\mathbf{x}(t))}{\|\nabla Q_k^*(\mathbf{x}(t))\|}, & \text{if } \mathbf{x}(t) \neq \mathbf{z}, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{4.30}$$

Remark 4.3.2. *The condition (4.29) indeed shows how to choose the end-point of the path over each planning horizon H.*

Remark 4.3.3. *In contrast to $Q^*(\boldsymbol{\xi})$ defined in (4.4), $Q_k^*(\boldsymbol{\xi})$ in (4.28) represents the minimal risk path to travel from the vehicle's current position $\mathbf{x}(t_k)$ to the point $\boldsymbol{\xi}$.*

Remark 4.3.4. *Note that from (4.9), the value of the Q_k^* indeed corresponds to the relative time that the autonomous vehicle travels from its current position to any point in the environment. Since the planning horizon is H, using the ordered upwind method ([66]), we*

can terminate the computation of Q_k^* as soon as the solution is larger than the constant $\gamma G_1 v_{\max} H$. Thus, Q_k^* will be computed over a small, local domain.

Proof of Proposition 4.3.1. The proof is very much similar to that of Proposition 4.2.1, given the fact that the matching condition (4.29) holds.

We adjoin the system differential equation (4.27) to $J(\mathbf{x}(\cdot), t_k)$ with multiplier $\boldsymbol{\lambda}(t) \in \mathcal{R}^2$ (see e.g. [6], pp. 48):

$$\begin{aligned}\overline{J}(\mathbf{x}(\cdot), t_k) = \min_{\mathbf{u}(\cdot)\in\mathbb{U}} (& \int_{t_k}^{t_k+H} g^*(\mathbf{x}(\tau), \tau)\|\mathbf{u}(\tau)\| + \boldsymbol{\lambda}^T(\tau)(\mathbf{u}(\tau) - \dot{\mathbf{x}}(\tau))d\tau \\ & + Q(\mathbf{x}(t_k + H)))\end{aligned} \tag{4.31}$$

Denoting the Hamiltonian by

$$\mathcal{H}(\mathbf{x}(t), \mathbf{u}(t)) = g^*(\mathbf{x}(t), t)\|\mathbf{u}(t)\| + \boldsymbol{\lambda}^T(t)\mathbf{u}(t) \tag{4.32}$$

we obtain the necessary condition for the optimality

$$\boldsymbol{\lambda}(t_k + H) = \frac{\partial Q^T}{\partial \mathbf{x}}(t_k + H), \tag{4.33}$$

$$\dot{\boldsymbol{\lambda}}(t) = -\frac{\partial \mathcal{H}^T}{\partial \mathbf{x}}(t) = -\frac{\partial g^{*T}(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)}\|\mathbf{u}(t)\|, \tag{4.34}$$

and

$$\frac{\partial \mathcal{H}^T}{\partial \mathbf{u}}(t) = \frac{g^*(\mathbf{x}(t), t)\mathbf{u}(t)}{\sqrt{\mathbf{u}^T(t)\mathbf{u}(t)}} + \boldsymbol{\lambda}(t) = \mathbf{0}. \tag{4.35}$$

It is obvious that from (4.8), the following equation holds for Q_k^*

$$t = \frac{1}{\gamma G_1 v_{\max}} Q_k^*(\mathbf{x}(t)) + t_k. \tag{4.36}$$

Thus, substituting (4.28), (4.30), and (4.36) into (4.34) yields

$$\begin{aligned}\dot{\boldsymbol{\lambda}}(t) &= -\frac{\partial g^{*T}(\mathbf{x}(t),t)}{\partial \mathbf{x}}(t)\|\mathbf{u}(t)\| \\ &= -\frac{\partial \|\nabla Q_k^*(\mathbf{x}(t))\|}{\partial \mathbf{x}(t)}\gamma\frac{G_1 v_{\max}}{g^*\left(\mathbf{x}(t), \frac{Q_k^*(\mathbf{x}(t))}{\gamma G_1 v_{\max}} + t_k\right)} \\ &= -\frac{\partial^2 Q_k^*}{\partial \mathbf{x}^2}\frac{\nabla Q_k^*(\mathbf{x}(t))}{\|\nabla Q_k^*(\mathbf{x}(t))\|}\gamma\frac{G_1 v_{\max}}{g^*\left(\mathbf{x}(t), \frac{Q_k^*(\mathbf{x}(t))}{\gamma G_1 v_{\max}} + t_k\right)} \\ &= -\frac{\partial^2 Q_k^*}{\partial \mathbf{x}^2}\dot{\mathbf{x}}(t)\end{aligned} \tag{4.37}$$

From (4.33) and given (4.29), integrating the above equation with respect to t yields for $t \in [t_k, t_k + H]$

$$\boldsymbol{\lambda}(t) = -\frac{\partial Q_k^{*T}}{\partial \mathbf{x}}(t) \tag{4.38}$$

Substituting (4.29), (4.38) and (4.30) into the left side of (4.35) yields

$$\begin{aligned}&\frac{g^*(\mathbf{x}(t),t)\mathbf{u}(t)}{\sqrt{\mathbf{u}^T(t)\mathbf{u}(t)}} + \boldsymbol{\lambda}(t) = \frac{g^*(\mathbf{x}(t),t)\nabla Q_k^*(\mathbf{x}(t))}{\|\nabla Q_k^*(\mathbf{x}(t))\|} - \frac{\partial Q_k^{*T}}{\partial \mathbf{x}}(t) \\ =&\frac{\partial Q_k^{*T}}{\partial \mathbf{x}}(t) - \frac{\partial Q_k^{*T}}{\partial \mathbf{x}}(t) = \mathbf{0}.\end{aligned} \tag{4.39}$$

Therefore, we can conclude that the equality of (4.35) holds which completes the proof. □

We now present a sufficient condition such that the vehicle will converge to the goal under the receding horizon control formulation (4.26).

Proposition 4.3.5. *Assume that the function $Q \in C^1(\overline{\Omega}; \mathcal{R})$. Define the value function*

$$V(\mathbf{x}(t_k), t_k) = J(\mathbf{x}(\cdot), t_k). \tag{4.40}$$

We assume that $V(\mathbf{x}, t)$ is a continuously differentiable function such that

$$W_1(\mathbf{x} - \mathbf{z}) \leq V(\mathbf{x}, t) \leq W_2(\mathbf{x} - \mathbf{z}),$$

where $W_1(\cdot)$ and $W_2(\cdot)$ are continuous positive definition functions on $\overline{\Omega}$. If the matching condition (4.29) holds for any time t_k, $V(\mathbf{x}(t_k), t_k)$ is a Lyapunov function. The vehicle state $\mathbf{x}(t) \to \mathbf{z}$ as $t \to \infty$.

Remark 4.3.6. *In the general case, $Q(\boldsymbol{\xi})$ is not differentiable everywhere. Interested readers are referred to [3] for the stability analysis when a Lyapunov function is not C^1.*

Remark 4.3.7. *The proof of Proposition 4.3.5 is very similar to the proof of asymptotic stability given for the receding horizon controller proposed in [48] and [29]. The difference is that since the risk for traversal $g^*(\mathbf{x}(t), t)$ depends on time, the optimal cost $J(\mathbf{x}(\cdot), t_k)$ depends on time as well. Thus, the Lyapunov function $V(\mathbf{x}, t)$ to be used in the following proof is for the asymptotical stability analysis for a non-autonomous system.*

Remark 4.3.8. *In the proposed receding horizon control formulation (4.26), the global level set Q which accounts for the static environment is assumed to be computed only once at the beginning of the mission. In practice, it would be recomputed, based on whether the matching condition (4.29) holds. The convergence result in Proposition 4.3.1 holds as long as there exists a $t_N > 0$ after which the global level set value Q does not need to be recomputed either because Q represents the complete map of the static environment or because the matching condition (4.29) holds for every planning horizon. The idea to globally update the level set Q will be clarified and discussed in detail in Chapter 5.*

Proof of Proposition 4.3.5. Following the procedure in [48] and [29], we denote by $\mathbf{u}_f(t)$ the admissible feedback controller that ensures that the vehicle will converge to the goal $\mathbf{z}$. Given the level set values Q, let the controller $\mathbf{u}_f$ satisfy

$$\mathbf{u}_f(t) = -\alpha v_{\max} \frac{\nabla Q(\mathbf{x}(t))}{\|\nabla Q(\mathbf{x}(t))\|}, \text{ if } \mathbf{x}(t) \neq \mathbf{z}, \tag{4.41}$$

for all $t \in [t_0, \infty)$, where

$$0 < \alpha \leq \gamma \frac{G_1}{G_2} \leq 1. \tag{4.42}$$

Thus, from (4.25) we infer that

$$\dot{Q}(\mathbf{x}(t)) = \nabla Q(\mathbf{x}(t)) \cdot \mathbf{u}_f(t) = -g(\mathbf{x}(t)) \alpha v_{\max}. \tag{4.43}$$

Therefore, $Q(\mathbf{x}(t))$ is a control Lyapunov function. Denote the optimal controller that minimizes $J(\mathbf{x}(\cdot), t_k)$ by $\mathbf{u}^*(t)$ for $t \in [t_k, t_k + H]$. Since $t_{k+1} = t_k + h$ and since $h \leq H$, we can

define the admissible controller $\mathbf{u}^+(t)$ satisfying

$$\mathbf{u}^+(t) = \begin{cases} \mathbf{u}^*(t), \ \forall t \in [t_{k+1}, t_{k+1} + H - h], \\ \mathbf{u}_f(t), \ \forall t \in [t_{k+1} + H - h, t_{k+1} + H]. \end{cases} \tag{4.44}$$

Given the initial state $\mathbf{x}(t_{k+1})$ at t_{k+1}, we define $\mathcal{J}(\mathbf{x}(t_{k+1}), \mathbf{u}^+(\cdot))$ the cost function that is evaluated by $\mathbf{u}^+(\cdot)$ satisfying

$$\mathcal{J}(\mathbf{x}(t_{k+1}), \mathbf{u}^+(\cdot)) := \int_{t_{k+1}}^{t_{k+1}+H} g^*(\mathbf{x}(\tau), \tau) \|\mathbf{u}^+(\tau)\| d\tau + Q(\mathbf{x}(t_{k+1} + H)). \tag{4.45}$$

Since $J(\mathbf{x}(\cdot), t_{k+1})$ is the minimal cost, we obtain the following inequality

$$\begin{aligned} & J(\mathbf{x}(\cdot), t_{k+1}) \\ \leq & \mathcal{J}(\mathbf{x}(t_{k+1}), \mathbf{u}^+(\cdot)) \\ = & \int_{t_{k+1}}^{t_{k+1}+H-h} g^*(\mathbf{x}(\tau), \tau) \|\mathbf{u}^+(\tau)\| d\tau \\ & + \int_{t_{k+1}+H-h}^{t_{k+1}+H} g^*(\mathbf{x}(\tau), \tau) \|\mathbf{u}^+(\tau)\| d\tau + Q(\mathbf{x}(t_{k+1} + H)) \\ = & \int_{t_{k+1}}^{t_{k+1}+H-h} g^*(\mathbf{x}(\tau), \tau) \|\mathbf{u}^*(\tau)\| d\tau \\ & + \int_{t_{k+1}+H-h}^{t_{k+1}+H} g^*(\mathbf{x}(\tau), \tau) \|\mathbf{u}_f(\tau)\| d\tau + Q(\mathbf{x}(t_{k+1} + H)) \\ = & \int_{t_k+h}^{t_k+H} g^*(\mathbf{x}(\tau), \tau) \|\mathbf{u}^*(\tau)\| d\tau \\ & + \int_{t_k+H}^{t_k+h+H} g^*(\mathbf{x}(\tau), \tau) \|\mathbf{u}_f(\tau)\| d\tau + Q(\mathbf{x}(t_{k+1} + H)). \end{aligned}$$

$$
\begin{aligned}
&= \underbrace{\int_{t_k}^{t_k+H} g^*(\mathbf{x}(\tau),\tau)\|\mathbf{u}^*(\tau)\|d\tau + Q(\mathbf{x}(t_k+H))}_{J(\mathbf{x}(\cdot),t_k)} \\
&\quad - \int_{t_k}^{t_k+h} g^*(\mathbf{x}(\tau),\tau)\|\mathbf{u}^*(\tau)\|d\tau - Q(\mathbf{x}(t_k+H)) \\
&\quad + \int_{t_k+H}^{t_k+h+H} g^*(\mathbf{x}(\tau),\tau)\|\mathbf{u}_f(\tau)\|d\tau + Q(\mathbf{x}(t_{k+1}+H)) \\
&= J(\mathbf{x}(\cdot),t_k) - \int_{t_k}^{t_k+h} g^*(\mathbf{x}(\tau),\tau)\|\mathbf{u}^*(\tau)\|d\tau - Q(\mathbf{x}(t_k+H)) \\
&\quad + \int_{t_k+H}^{t_k+h+H} g^*(\mathbf{x}(\tau),\tau)\|\mathbf{u}_f(\tau)\|d\tau + Q(\mathbf{x}(t_k+H+h)).
\end{aligned}
\tag{4.46}
$$

From (4.41), (4.43) and (4.44), together with the inequality (4.46), we infer that

$$
\begin{aligned}
&J(\mathbf{x}(\cdot),t_{k+1}) - J(\mathbf{x}(\cdot),t_k) \\
\leq &- \int_{t_k}^{t_k+h} g^*(\mathbf{x}(\tau),\tau)\|\mathbf{u}^*(\tau)\|d\tau \\
&+ \int_{t_k+H}^{t_k+h+H} g^*(\mathbf{x}(\tau),\tau)\|\mathbf{u}_f(\tau)\|d\tau + \int_{t_k+H}^{t_k+h+H} \frac{\partial Q}{\partial \mathbf{x}}(\tau)\mathbf{u}_f(\tau)d\tau \\
= &- \int_{t_k}^{t_k+h} g^*(\mathbf{x}(\tau),\tau)\|\mathbf{u}^*(\tau)\|d\tau + \int_{t_k+H}^{t_k+h+H} g^*(\mathbf{x}(\tau),\tau)\alpha v_{\max}d\tau \\
&- \int_{t_k+H}^{t_k+h+H} g(\mathbf{x}(\tau))\alpha v_{\max}d\tau.
\end{aligned}
\tag{4.47}
$$

From the fact that the matching condition (4.29) holds, we conclude that $\mathbf{u}^*$ satisfies (4.30). Thus (4.47) further becomes

$$
\begin{aligned}
&J(\mathbf{x}(\cdot),t_{k+1}) - J(\mathbf{x}(\cdot),t_k) \\
\leq &- \int_{t_k}^{t_k+h} \gamma v_{\max} G_1 d\tau + \int_{t_k+H}^{t_k+h+H} g^*(\mathbf{x}(\tau),\tau)\alpha v_{\max}d\tau - \int_{t_k+H}^{t_k+h+H} g(\mathbf{x}(\tau))\alpha v_{\max}d\tau \\
= &\int_{t_k+H}^{t_k+h+H} (g^*(\mathbf{x}(\tau),\tau)\alpha - \gamma G_1)v_{\max}d\tau - \int_{t_k+H}^{t_k+h+H} g(\mathbf{x}(\tau))\alpha v_{\max}d\tau.
\end{aligned}
\tag{4.48}
$$

From (4.42), the inequality (4.48) becomes

$$\begin{aligned}
&J(\mathbf{x}(\cdot), t_{k+1}) - J(\mathbf{x}(\cdot), t_k) \\
&\leq \int_{t_k+H}^{t_k+h+H} \left(g^*(\mathbf{x}(\tau), \tau)\gamma \frac{G_1}{G2} - \gamma G_1 \right) v_{\max} d\tau - \int_{t_k+H}^{t_k+h+H} g(\mathbf{x}(\tau))\alpha v_{\max} d\tau \\
&= + \int_{t_k+H}^{t_k+h+H} g^*(\mathbf{x}(\tau), \tau) G_1 \gamma v_{\max} \left(\frac{1}{G_2} - \frac{1}{g^*(\mathbf{x}(\tau), \tau)} \right) d\tau - \int_{t_k+H}^{t_k+h+H} g(\mathbf{x}(\tau))\alpha v_{\max} d\tau \\
&= \int_{t_k+H}^{t_k+h+H} g^*(\mathbf{x}(\tau), \tau) G_1 \gamma v_{\max} \underbrace{\left(\frac{g^*(\mathbf{x}(\tau), \tau) - G_2}{G_2 g^*(\mathbf{x}(\tau), \tau)} \right)}_{\leq 0} d\tau - \int_{t_k+H}^{t_k+h+H} g(\mathbf{x}(\tau))\alpha v_{\max} d\tau \\
&\leq - \int_{t_k+H}^{t_k+h+H} g(\mathbf{x}(\tau))\alpha v_{\max} d\tau.
\end{aligned} \tag{4.49}$$

According to (4.40), we have for all $\mathbf{x}(t_k) \neq \mathbf{z}$,

$$\begin{aligned}
&\dot{V}(\mathbf{x}(t_k), t_k) \\
&= \lim_{h \to 0} \frac{V(\mathbf{x}(t_{k+1}), t_{k+1}) - V(\mathbf{x}(t_k), t_k)}{h} \\
&\leq - g(\mathbf{x}(t_k + H))\alpha v_{\max} \\
&\leq - G_1 \alpha v_{\max}.
\end{aligned} \tag{4.50}$$

Since the environmental geometry $\overline{\Omega}$ is a closed and bounded set, there exists a continuous positive definite function $W_3 : \overline{\Omega} \to \mathcal{R}$ such that

$$W_3(\mathbf{x} - \mathbf{z}) \leq G_1 \alpha v_{\max}.$$

Thus, the inequality (4.50) becomes

$$\dot{V}(\mathbf{x}(t_k), t_k) \leq -W_3(\mathbf{x}(t_k) - \mathbf{z}).$$

By the hypothesis on $V(\mathbf{x}, t)$, $V(\mathbf{x}, t)$ is a Lyapunov function for a non-autonomous system. Thus, we conclude that the vehicle converges to the goal $\mathbf{z}$ (see e.g. Theorem 4.9 of [32]). □

4.4 Simulation Results

To validate the proposed receding horizon controller, we study an example of an autonomous surface vehicle (ASV) navigating in a riverine environment. We employ the ordered upwind method proposed in [66] and [78] for computing the solutions of Eikonal equation and the PDE (4.28). This method has a computational complexity of $O(N \ln N)$ where N is the total number of the cells in the domain $\overline{\Omega}$. Figure 4.1 represents the river map which spans an area of $800m \times 600m$. We assume that no new static obstacles are present in the environment but there are three moving obstacles traveling in the river as seen in Figure 4.1. The trajectories of these moving obstacles are chosen so that a collision will occur if the vehicle does not account for these obstacles. The on-board sensor can detect them within 60 meters range. We assume that as soon as the moving obstacles enter the detection range, the ASV immediately knows their trajectories. For a point $\boldsymbol{\xi} \in \overline{\Omega}$, depending on the distance between $\boldsymbol{\xi}$ and the moving obstacles at time t, we define

$$g^*(\boldsymbol{\xi}, t) = \begin{cases} 7, & \text{if } \operatorname{dist}(\boldsymbol{\xi}, \text{obstacle}) \leq 9, \\ 0.2, & \text{otherwise.} \end{cases}$$

Thus, we set the lower bound $G_1 = 0.2$. We choose the planning horizon $H = 50$ *secs*, the maximum speed $v_{\max} = 5$ m/s and $\gamma = 1$. Since the matching condition (4.29) is satisfied at each horizon, as shown in Figure 4.2, the ASV eventually reaches the target. We mark two rectangles corresponding to the locations where the ASV encounters the moving obstacles. The details of the motion of the ASV and the moving obstacles motions are shown respectively in Figure 4.3 and 4.4.

In order to illustrate how to choose the end point at each horizon, as well as how the ASV manages to avoid the moving obstacles, we discuss the receding horizon planning process corresponding to the ASV's motion shown in Figure 4.3. The solution of the Eikonal equation (4.25) over the global domain, as shown in Figure 3.2, serves as the terminal cost in the receding horizon control formulation (4.26). When we replan the path over each horizon, we compute the solution of the PDE (4.28) over a small, local domain as shown in Figure

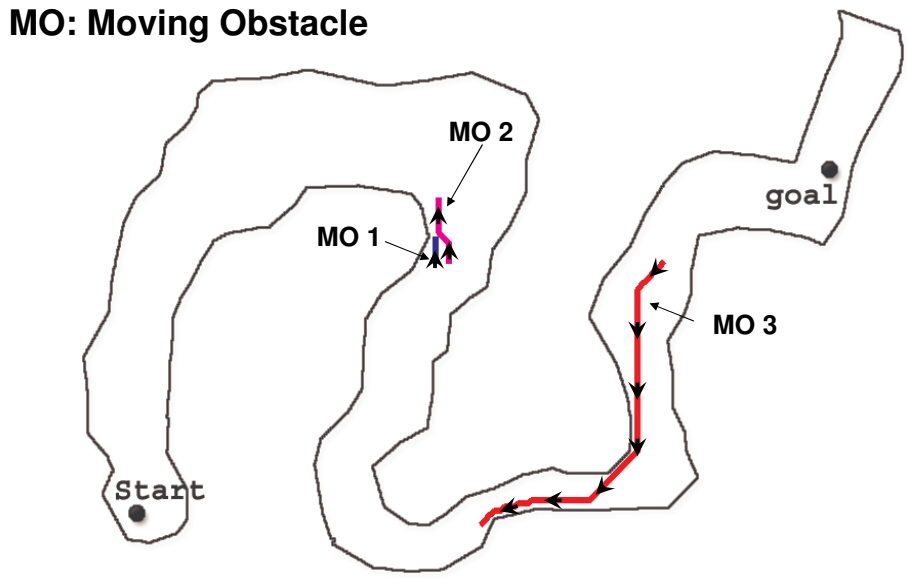

Figure 4.1: The riverine map and moving obstacles.

4.5. Following the discussion in Remark 4.3.4, we terminate the computation of the local PDE as soon as its solution exceeds $\gamma G_1 v_{\max} H$. Thus, as shown in Figure 4.5, the solutions of the PDE in the vicinity of the moving obstacles will not have been computed before the termination. These values will be higher than $\gamma G_1 v_{\max} H$. When the ASV searches for the new path, it automatically avoids these areas. To choose the end point for the planning horizon, we check whether the condition (4.29) holds at the end of the planning horizon, as shown in Figure 4.5. Note that the errors in the numerical approximation are evident. Thus, we relax the condition (4.29) so that if the difference between the two sides of (4.29) is smaller than a given threshold, we say that the condition (4.29) holds. In the example, the global level set Q is only computed once at the beginning of the mission. As we have discussed in Remark 4.3.8, we would recompute Q as needed to ensure that the global level-set adequately represents the static environment. This idea has been reported in [81] and

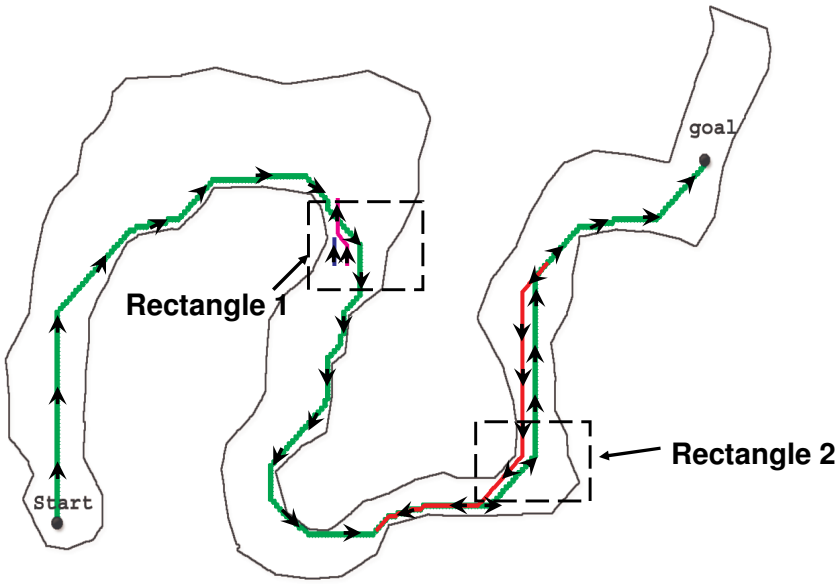

Figure 4.2: The planned trajectory of the ASV plotted onto the map.

will be clarified and discussed in detail in Chapter 5.

4.5 Concluding Remarks

In this chapter, we address the minimal risk planning problem in the presence of both moving and static obstacles. In the case that the position of both the moving and static obstacles are known in advance, we employ the method proposed in [78] that solves a PDE over the global environmental domain. To plan paths in more realistic scenarios, we discuss the case when the moving obstacles are detected in the mission by a sensor with limited range. We propose a receding horizon controller for planning an obstacle-free path. Compared to solving a PDE over the global domain, the proposed receding horizon control method solves a PDE over

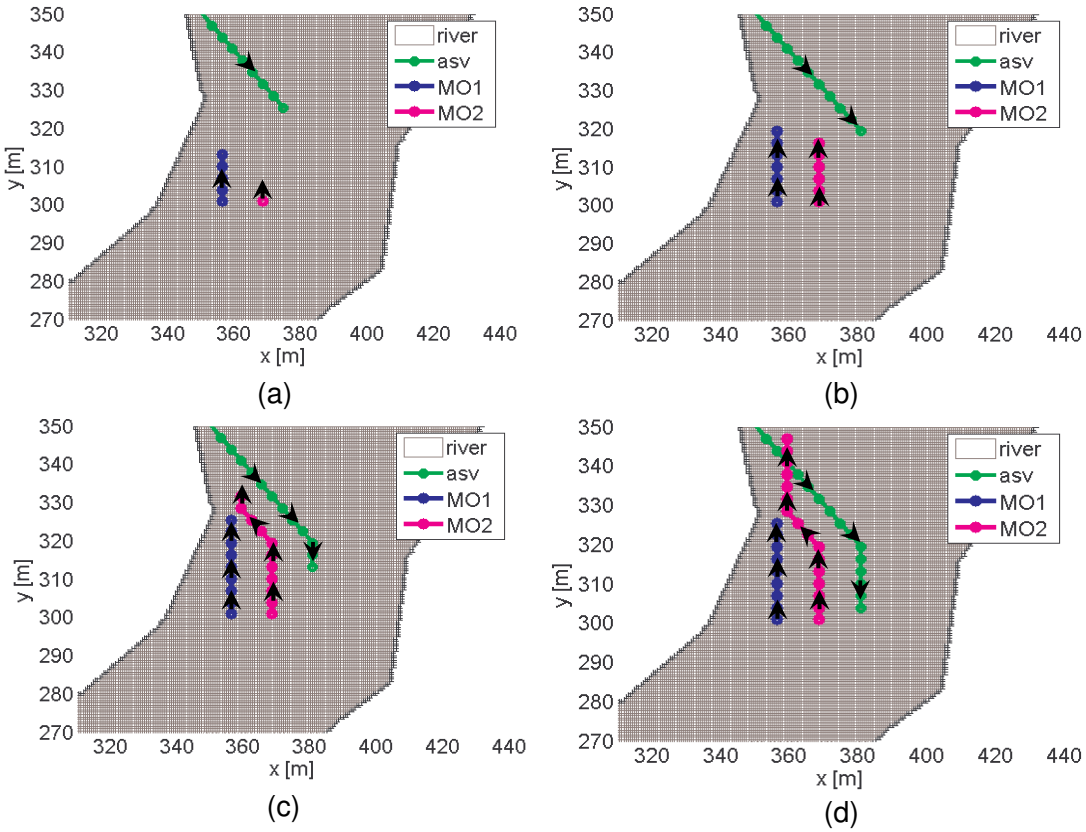

Figure 4.3: The closeup of Rectangle 1 where the ASV encounters both the moving obstacles MO1 and MO2. The motions of the ASV and the moving obstacles MO1 and MO2 are shown in (a)-(d) respectively.

a small, local domain. Thus the computational cost is reduced. In addition, we derive a sufficient condition for the proposed receding horizon controller that guarantees that the vehicle will converge to the goal. The simulation results show that the proposed receding horizon control method is effective in planning paths for a complex environment.

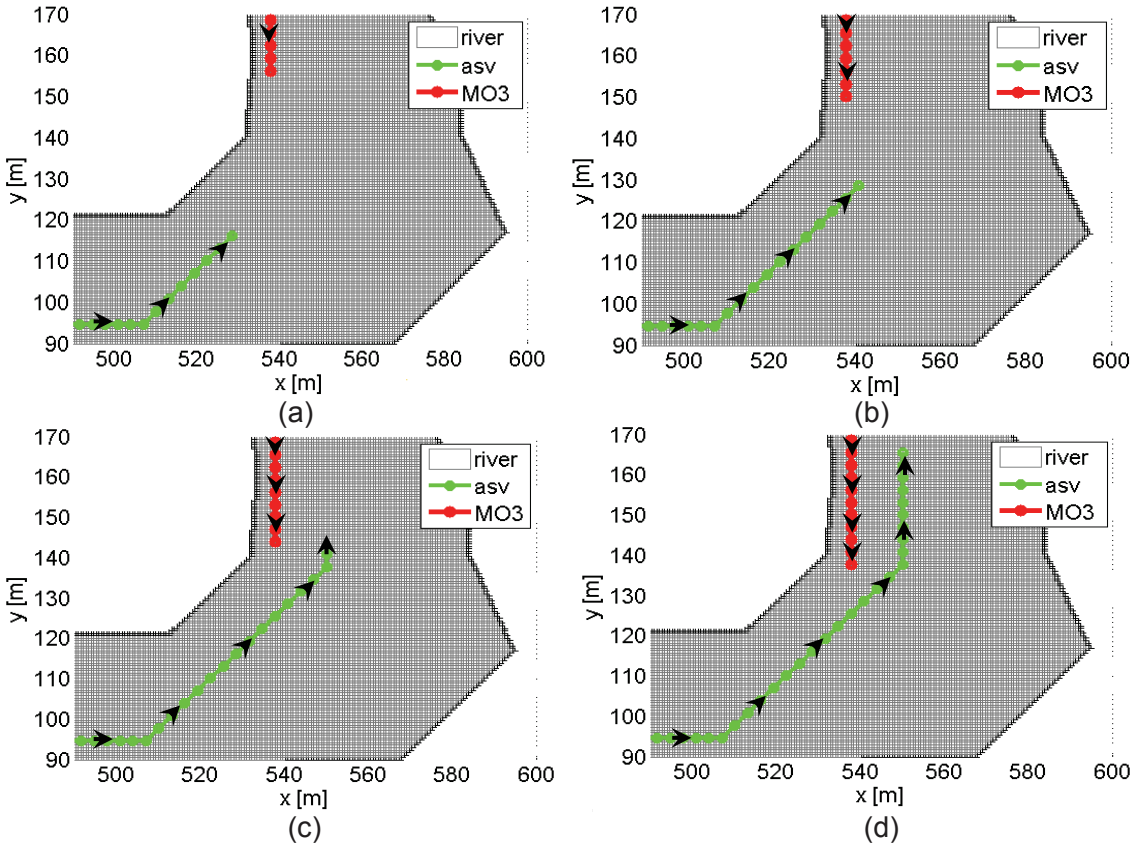

Figure 4.4: The closeup of Rectangle 2 where the ASV encounters the moving obstacle MO3. The motions of the ASV and the moving obstacle MO3 are shown in (a)-(d) respectively.

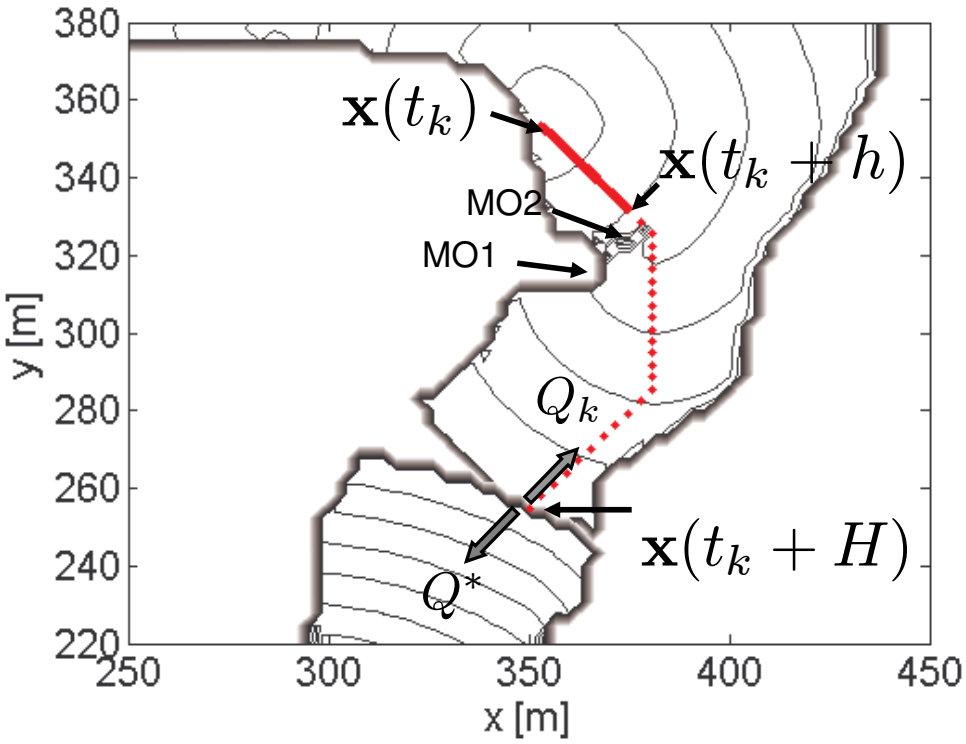

Figure 4.5: The solution of the local PDE, the partial solution of the global Eikonal equation and the planned trajectory. Two gray arrows correspond to $\nabla Q_k(\mathbf{x}(t_k+H))$ and $\nabla Q^*(\mathbf{x}(t_k+H))$ respectively. The solid and dash red lines are, respectively, the planned ASV trajectory for $[t_k, t_k + h]$ and $[t_k + h, t_k + H]$.

Chapter 5

Dynamic Fast Marching Method for Global Replanning

In this chapter, we propose a dynamic fast marching method that globally replans paths in the presence of environmental changes including both new empty areas and new static obstacles. Computational costs for path planning using level-set methods are due to creation of the level set. Once the level set has been computed, the optimal path is simply gradient descent down the level set. Our algorithm addresses, separately, the case in which new obstacles are detected and the case in which new empty areas are detected. For the case where new obstacles are detected, the proposed method reduces the computational expenses in two aspects. First, we show that if obstacles are not on the vehicle's current optimal trajectory and if there are no unexpected empty areas, the trajectory remains optimal. This observation allows us to delay the computation of the level-set update if the original trajectory is still feasible. Second, if an unexpected obstacle intersects the current path, we show that only a portion of the level set needs to be recomputed. For the case where new empty areas are detected, we again show that only a portion of the level set usually needs to be computed. As illustrated by numerical simulations in Section 5.4, the proposed algorithm can find new paths with low computational cost, especially for environments similar to riverine systems,

which motivates our work.

While the algorithm proposed in the chapter functions as an independent global path planner, it also applies to the case in which one needs to solve the global Eikonal equation in the hybrid receding horizon controller proposed in Chapter 3. We elaborate this point in Section 5.2.2.

The chapter is organized as follows. In Section 5.1, we introduce the conventional fast marching method. In Section 5.2, we show the development of the proposed replanning strategy when new obstacles are detected. In Section 5.3, we propose the algorithm that updates the level sets when new empty areas are detected. Simulation results are illustrated in Section 5.4. The concluding remarks are discussed in the last section.

5.1 Preliminaries: Fast Marching Method and Problem Statement

In this section, we introduce the fast marching method that is originally developed in [65]. We also present the model of the environments that is different from Chapter 3 and enables us to associate the cost function g with a stochastic map. In the end of the section, we formulate our goal in this chapter.

5.1.1 Fast Marching Method and Finite Difference Scheme

The Eikonal equation

$$\begin{aligned} \|\nabla Q(\boldsymbol{\xi})\| &= g(\boldsymbol{\xi}), \\ Q(\mathbf{z}) &= 0, \end{aligned} \tag{5.1}$$

often cannot be solved analytically. In [65], a first order update scheme is proposed which approximates the viscosity solution of (5.1).

Finite Difference Scheme

Let the goal location be $\mathbf{z} = [z_1, z_2]^T$. In order to discretize $\overline{\Omega}$, we define a set $\Psi \in \mathbb{Z} \times \mathbb{Z}$ that is composed of grids with mesh size Δx, where $\mathbb{Z}$ is the set of integers. We denote the approximate value of Q by $\mathcal{Q}: \Psi \to \mathbb{R}^1$ satisfying

$$\mathcal{Q}(i,j) \simeq Q(i\Delta x + z_1, j\Delta x + z_2). \tag{5.2}$$

Correspondingly, we approximate the function g with $\mathbf{g}: \Psi \to \mathbb{R}^1$ satisfying

$$\mathbf{g}(i,j) = g(i\Delta x + z_1, j\Delta x + z_2). \tag{5.3}$$

We define the neighbors of a grid element $(i,j) \in \Psi$ to be the set of grid elements $(i+1,j)$, $(i-1,j)$, $(i,j+1)$ and $(i,j-1)$. If grid element (i,j) satisfies $[i\Delta x + z_1, j\Delta x + z_2]^T \notin \overline{\Omega}$, we set value $\mathcal{Q}(i,j)$ to be a very large number. Otherwise, if grid element (i,j) satisfies $[i\Delta x + z_1, j\Delta x + z_2]^T \in \overline{\Omega}$, the numerical approximation $\mathcal{Q}(i,j)$ satisfies the following conditions

$$\mathcal{Q}(0,0) = 0 \tag{5.4}$$

$$\begin{aligned} &\max\left(\frac{\mathcal{Q}(i,j) - \min(\mathcal{Q}(i-1,j), \mathcal{Q}(i+1,j))}{\Delta x}, 0\right)^2 \\ &+ \max\left(\frac{\mathcal{Q}(i,j) - \min(\mathcal{Q}(i,j+1), \mathcal{Q}(i,j-1))}{\Delta x}, 0\right)^2 \\ &- \mathbf{g}^2(i,j) = 0, \quad \forall (i,j) \neq (0,0) \end{aligned} \tag{5.5}$$

which converges to the continuous solution as $\Delta x \to 0$. As remarked in [62], $\mathcal{Q}$ in (5.4) and (5.5) exhibits a first order accuracy of order Δx.

The discrete approximation solution $\mathcal{Q}$ is found as follows. We define

$$\begin{aligned} a &:= \min(\mathcal{Q}(i+1,j), \mathcal{Q}(i-1,j)), \\ b &:= \min(\mathcal{Q}(i,j+1), \mathcal{Q}(i,j-1)). \end{aligned} \tag{5.6}$$

The approximate solution $\mathcal{Q}(i,j)$ is computed by identifying two cases,

Case 1: If $|a-b| \geq \mathbf{g}(i,j)\Delta x$, then

$$\mathcal{Q}(i,j) = \min(a,b) + \mathbf{g}(i,j)\Delta x. \tag{5.7}$$

Case 2: If $|a-b| < \mathbf{g}(i,j)\Delta x$, then $\mathcal{Q}(i,j)$ is selected as the larger solution of the quadratic equation

$$(\mathcal{Q}(i,j)-a)^2 + (\mathcal{Q}(i,j)-b)^2 - \mathbf{g}^2(i,j)\Delta x^2 = 0, \tag{5.8}$$

that is

$$\mathcal{Q}(i,j) = \left(a+b+\sqrt{2\mathbf{g}^2(i,j)\Delta x^2 - (a-b)^2}\right)/2. \tag{5.9}$$

Both (5.7) and (5.9) show that for each grid element (i,j), the value $\mathcal{Q}(i,j)$ depends on the smaller values of the neighbors. This is called upwind property indicating that the values of $\mathcal{Q}$ propagate from smaller values to larger ones. In addition, the scheme admits no local minima. Indeed, if $\mathcal{Q}(i,j)$ would be lower than its neighbors, and given that $\mathbf{g}(i,j) > 0$ for all $(i,j) \neq (0,0)$, the left-hand side of (5.5) would be negative.

Fast Marching Method Algorithm

The fast marching method proposed in [65] can efficiently compute the solution for (5.5). Indeed, the fast marching method solves (5.4) and (5.5) in $O(N \log N)$ where N is the number of grids in $\overline{\Omega}$. Making use of the upwind property, the fast marching method builds the solution outward from smaller values of $\mathcal{Q}$ to larger values. Readers are referred to [65] for details.

Directed Graph

As proposed in [54], when computing the approximate solution $\mathcal{Q}$ of the Eikonal equation, one can use a directed graph to explicitly represent which neighbor grid elements the value

of $\mathcal{Q}(i,j)$ depends upon. We denote by Σ the graph whose nodes correspond to the grid elements in Ψ and whose directed edges represent the computational dependance between the value of the level set at each node. As an example depicted in Figure 5.1, we illustrate the construction of Σ with respect to the value dependence of the grid element in the center. For notational simplicity, we define the center grid element by E and its four neighbors by A, B, C and D. For *Case 1*, since the value $\mathcal{Q}(E)$ is determined by its smallest neighbor, say node A, Σ would contain an a directed edge from A to E as in Figure 5.1 (b). For *Case 2*, $\mathcal{Q}(E)$ depends on two nodes, say A and B. Then Σ would contain directed edges from A to E and from B to E, as in Figure 5.1 (c). If there is a directed edge from a node (i,j) to another node (k,m), then (k,m) is said to be a direct child of (i,j), and (i,j) is said to be a direct parent of (k,m). In general, if a path leads from (i,j) to (p,q), then (p,q) is said to be a child of (i,j) and (i,j) is said to be a parent of (p,q).

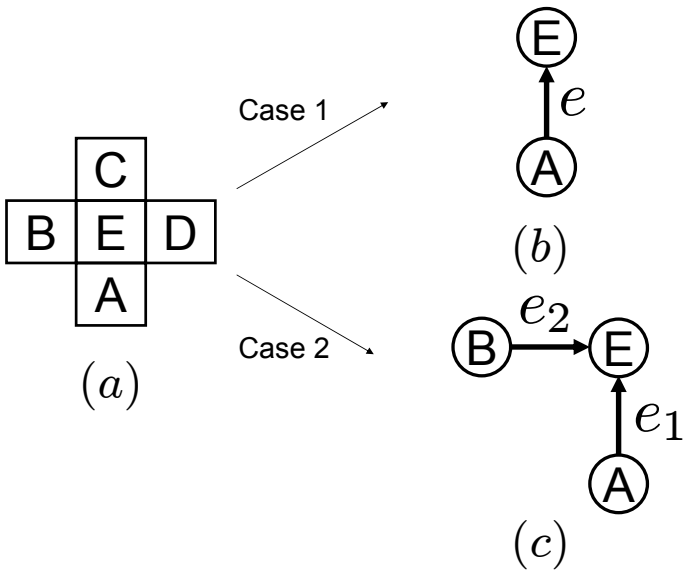

Figure 5.1: (a) Grid E and its neighbors; (b) Graph Σ for *Case 1*; and (c) Graph Σ for *Case 2*

5.1.2 Cost Function and Observation Process of the Environment

Since the onboard sensor has finite range, changes to the map are local to the vehicle. We assume that the sensor obtains measurements periodically with period h. Let the sensor sampling time be $t_k = t_0 + kh$, where $k = 0, 1, \cdots$. In order to model the environmental changes that are induced by detection of obstacles, we denote the cost functions at time instant t_k by g_k. If the sensor detects no changes in the map at time t_{k+1}, then we write $g_k = g_{k+1}$ to denote that $g_k(\boldsymbol{\xi}) = g_{k+1}(\boldsymbol{\xi})$ for any point $\boldsymbol{\xi} \in \overline{\Omega}$. If a point $\boldsymbol{\xi} \in \overline{\Omega}$ is detected in an obstacle domain at time t_{k+1}, then $g_k(\boldsymbol{\xi}) < g_{k+1}(\boldsymbol{\xi})$. Conversely, if an obstacle on the map is determined not to exist by the local sensor at a point $\boldsymbol{\xi} \in \overline{\Omega}$ at time t_k, then $g_k(\boldsymbol{\xi}) > g_{k+1}(\boldsymbol{\xi})$. Thus, the new Eikonal equation that is induced by new cost functions g_{k+1} at time t_{k+1} becomes

$$\|\nabla Q_{k+1}(\boldsymbol{\xi})\| = g_{k+1}(\boldsymbol{\xi}),\ Q_{k+1}(\mathbf{z}) = 0. \tag{5.10}$$

where $\boldsymbol{\xi} \in \overline{\Omega}$.

One way to update the solution of Eikonal equation (5.10) is to recalculate level sets over the entire domain $\overline{\Omega}$ with respect to the new cost function g_k. For the purpose of reducing the computation cost, we propose a new path replanning method such that the solutions of Eikonal equations can be more efficiently updated upon the detection of new environmental changes.

In the following sections, to facilitate the analysis, we separately address the case for which new obstacles are detected and the case for which *a priori* known obstacles are found to not exist. When we address the path replanning in the presence of new obstacles, we assume that g_k monotonically increases with the time sequence t_k, and vice versa. We find, in our analysis, that the two cases are reversible in some situations.

5.2 Level Set Updates When New Obstacles Are Detected

In this section, we discuss the case that only new obstacles are detected. Thus, we assume that g_k monotonically increases with the time sequence t_k. We show that an optimal path remains optimal when a new obstacle is detected so long as the obstacle does not intersect the path. Due to this fact, we are required to update the level set only when a new obstacle intersects the path. For notational simplicity, without further specification, $g_{k+1} \geq g_k$ means that, for any $\xi \in \overline{\Omega}$, $g_{k+1}(\boldsymbol{\xi}) \geq g_k(\boldsymbol{\xi})$. Correspondingly, similar meaning can be deduced for $\mathbf{g}_{k+1} \geq \mathbf{g}_k$ in the discrete case.

Proposition 5.2.1. *Suppose Q_k and Q_{k+1} are the solutions of (5.10) for the cost functions g_k and g_{k+1}, respectively. Denote the obstacle detected at time t_{k+1} by a bounded open set $\mathcal{O}(k+1) \subset \overline{\Omega}$. Assume $g_{k+1} \geq g_k$. Then,*

(a) For all $\boldsymbol{\xi} \in \overline{\Omega}$, $Q_{k+1}(\boldsymbol{\xi}) \geq Q_k(\boldsymbol{\xi})$.

(b) If $\mathbf{c}^*$ is the optimal path associated to g_k and if $\mathcal{O}(k+1)$ does not intersect $\mathbf{c}^*$, then $\mathbf{c}^*$ is still one of the optimal paths associated to the new cost function g_{k+1}.

Proof of Proposition 5.2.1. We prove (a) first and then (b) as an immediate implication of (a).

Let $I = [0, 1]$. Given the cost function g_k, denote the cumulative cost along an arbitrary differentiable parameterized path $\mathbf{c}(p) \in Lip(I; \overline{\Omega})$ by

$$J_k(\mathbf{c}) = \int_I g_k(\mathbf{c}(p)) \, \|\mathbf{c}'(p)\| \, dp, \tag{5.11}$$

where $\mathbf{c}(0) = \boldsymbol{\xi}$ and $\mathbf{c}(1) = \mathbf{z}$. Thus, by definition,

$$Q_k = \min_{\mathbf{c} \in Lip(I; \overline{\Omega})} J_k(\mathbf{c}), \tag{5.12}$$

and

$$Q_{k+1} = \min_{\mathbf{c} \in Lip(I;\overline{\Omega})} J_{k+1}(\mathbf{c}). \tag{5.13}$$

Denote with L the intersection of curve $\mathbf{c}(p)$ and obstacle $\mathcal{O}(k+1)$. Thus, L satisfies

$$L = \{p \in I : \ \mathbf{c}(p) \subseteq \overline{\mathcal{O}(k+1)}\}. \tag{5.14}$$

Since for all $\boldsymbol{\xi} \in \overline{\Omega}$, $g_{k+1}(\boldsymbol{\xi}) \geq g_k(\boldsymbol{\xi})$, subtracting $J_k(\mathbf{c})$ from $J_{k+1}(\mathbf{c})$ yields

$$\begin{aligned} \Delta J(\mathbf{c}) =& J_{k+1}(\mathbf{c}) - J_k(\mathbf{c}) \\ =& \int_{I \setminus L} (g_{k+1}(\mathbf{c}(p)) - g_k(\mathbf{c}(p))) \, \|\mathbf{c}'(p)\| \, dp \\ &+ \int_L (g_{k+1}(\mathbf{c}(p)) - g_k(\mathbf{c}(p))) \, \|\mathbf{c}'(p)\| \, dp \\ =& \int_L (g_{k+1}(\mathbf{c}(p)) - g_k(\mathbf{c}(p))) \, \|\mathbf{c}'(p)\| \, dp. \end{aligned} \tag{5.15}$$

By inspecting the above equation, we conclude that (i) $\Delta J(\mathbf{c}) = 0$ when $L = \varnothing$, and (ii) $\Delta J(\mathbf{c}) \geq 0$ when $L \neq \varnothing$. This implies

$$J_{k+1}(\mathbf{c}) \geq J_k(\mathbf{c}) \tag{5.16}$$

for any differentiable parameterized path $\mathbf{c}$ connecting $\boldsymbol{\xi}$ and $\mathbf{z}$. Since Q_k and Q_{k+1} are the optimal values for $J_k(\mathbf{c})$ and $J_{k+1}(\mathbf{c})$, respectively, we conclude that $Q_{k+1} \geq Q_k$.

We now prove (b). Considering (5.15), since $L = \varnothing$, for the path along $\mathbf{c}^*$

$$J_{k+1}(\mathbf{c}^*) = J_k(\mathbf{c}^*). \tag{5.17}$$

Since $\mathbf{c}^*$ is the optimal path given g_k, $J_k(\mathbf{c}^*) \leq J_k(\mathbf{c})$. Together with (5.16), we conclude that for an arbitrary curve $\mathbf{c}$,

$$J_{k+1}(\mathbf{c}^*) = J_k(\mathbf{c}^*) \leq J_k(\mathbf{c}) \leq J_{k+1}(\mathbf{c}). \tag{5.18}$$

The inequality (5.18) indicates that the path $\mathbf{c}^*$ is the optimal for the cost function $\mathbf{g}_{k+1}$. □

We analyze changes in the discretized solutions of the Eikonal equation that result when the value of the cost function $\mathbf{g}_k$ increases. Using the result of our analysis and employing the directed graph introduced in Section 5.1.1, we show that level sets do not necessarily need to be updated everywhere, and we identify the group of grids whose level sets should be updated.

5.2.1 The Approximation of Level Sets in Discrete Space

Proposition 5.2.1 indicates that the values of level set monotonically increase if the cost function monotonically increases. We now investigate the corresponding property for the discrete approximation of the level set. Although the update scheme (5.5) introduces approximation errors, the following proposition ensures monotonicity of the approximation $\mathcal{Q}_k$ with respect to the risk $\mathbf{g}_k$. This proposition will be used, in part, to show that if the sequence of discrete cost functions satisfy $\mathbf{g}_k \leq \mathbf{g}_{k+1}$, then discrete approximation of the level set satisfies $\mathcal{Q}_k \leq \mathcal{Q}_{k+1}$.

Proposition 5.2.2. *Given cost functions $\mathbf{g}_k$ and $\mathbf{g}_{k+1}$, let $\mathcal{Q}_k$ and $\mathcal{Q}_{k+1}$ be the discrete solutions to (5.10). If $\mathbf{g}_{k+1} \geq \mathbf{g}_k$, the approximation satisfies $\mathcal{Q}_{k+1}(i,j) \geq \mathcal{Q}_k(i,j)$ for all grid element $(i,j) \in \Psi$.*

Proof of Proposition 5.2.2. Using the monotonicity of the scheme (5.5), we show the proof by contradiction. From the scheme (5.5), we define a function

$$\begin{aligned} & F(\mathcal{Q}(i,j), \Phi_{ij}(\mathcal{V}), \mathbf{g}(i,j)) \\ :=& \max\left(\frac{\mathcal{Q}(i,j) - \min(\mathcal{V}(i-1,j), \mathcal{V}(i+1,j))}{\Delta x}, 0\right)^2 \\ & + \max\left(\frac{\mathcal{Q}(i,j) - \min(\mathcal{V}(i,j+1), \mathcal{V}(i,j-1))}{\Delta x}, 0\right)^2 \\ & - \mathbf{g}^2(i,j) \end{aligned} \tag{5.19}$$

where the second argument in $F(\cdot,\cdot,\cdot)$ is defined as the set

$$\Phi_{ij}(\mathcal{V}) := \{\mathcal{V}(i-1,j), \mathcal{V}(i+1,j), \mathcal{V}(i,j+1), \mathcal{V}(i,j-1)\}.$$

Note that when the scheme (5.5) holds, we have, for example,

$$0 = F(\mathcal{Q}_k(i,j), \Phi_{ij}(\mathcal{Q}_k), \mathbf{g}_k(i,j)). \tag{5.20}$$

We denote by Σ_{k+1} the graph corresponding to the value dependence of the solution $\mathcal{Q}_{k+1}$. Assume that there exists a node (i,j) where

$$\mathcal{Q}_{k+1}(i,j) < \mathcal{Q}_k(i,j). \tag{5.21}$$

Given that $\mathbf{g}_k \leq \mathbf{g}_{k+1}$, from (5.20) and (5.21), we have

$$\begin{aligned} 0 =& F(\mathcal{Q}_k(i,j), \Phi_{ij}(\mathcal{Q}_k), \mathbf{g}_k(i,j)) \\ \geq& F(\mathcal{Q}_k(i,j), \Phi_{ij}(\mathcal{Q}_k), \mathbf{g}_{k+1}(i,j)) \\ >& F(\mathcal{Q}_{k+1}(i,j), \Phi_{ij}(\mathcal{Q}_k), \mathbf{g}_{k+1}(i,j)) \end{aligned} \tag{5.22}$$

From the definition of (5.19), the value $F(\mathcal{Q}_{k+1}(i,j), \Phi_{ij}(\mathcal{V}), \mathbf{g}_{k+1})$ monotonically decreases as the value of a neighbor of (i,j), say $\mathcal{V}(i+1,j)$, increases. Thus, from the inequality (5.22), we infer that there is at least one direct parent of the node (i,j) in the graph Σ_{k+1}, say $(i+1,j)$, satisfying

$$\mathcal{Q}_{k+1}(i+1,j) < \mathcal{Q}_k(i+1,j). \tag{5.23}$$

Repeating the same analysis, we always infer that (5.23) holds for the parents of (i,j) by tracing backward along the graph Σ_{k+1}, which eventually leads us to the conclusion that

$$\mathcal{Q}_{k+1}(0,0) < \mathcal{Q}_k(0,0) = 0.$$

The contradiction completes the proof. □

By using the directed graph, we identify the nodes that need to be updated. Assume that at time t_{k+1}, the vehicle detects new obstacles. Then, the cost functions of these grids are

such that $\mathbf{g}_{k+1} > \mathbf{g}_k$. Consider a graph Σ_k that indicates the value dependence of the level set on other neighbor nodes at time t_k. Using Σ_k, Proposition 5.2.3, and Corollary 5.2.4, we show that when the cost function $\mathbf{g}_k$ increases to $\mathbf{g}_{k+1}$, the level set does not necessarily need to be updated at all grid elements, and we can identify the nodes for which the level set update is necessary.

Proposition 5.2.3. *Denote the computational dependence between nodes for fast marching algorithm by the graphs Σ_k and Σ_{k+1} for times t_k and t_{k+1}, respectively. Assume that for every node $\mathbf{g}_{k+1} \geq \mathbf{g}_k$. For a grid element (i,j), if $\mathbf{g}_{k+1}(i,j) = \mathbf{g}_k(i,j)$ and, for all direct parents nodes of (i,j) denoted by (l,m), $\mathcal{Q}_k(l,m) = \mathcal{Q}_{k+1}(l,m)$, then $\mathcal{Q}_{k+1}(i,j) = \mathcal{Q}_k(i,j)$ and the direct parents of (i,j) in Σ_k are direct parents of (i,j) in Σ_{k+1}.*

Proof of Proposition 5.2.3. We only detail the proof for *Case 1*. The proof for *Case 2* follows the similar procedure. For notational simplicity, we define $E := (i,j)$ and neighbors of E by A, B, C and D as shown in Figure 5.1 (a). We assume that A is the only direct parent of E. Thus from (5.5),

$$\mathcal{Q}_k(A) \leq \mathcal{Q}_k(C). \tag{5.24}$$

From (5.5) and the assumption that node A is the only direct parent of E,

$$\mathcal{Q}_k(E) - \mathcal{Q}_k(A) = \Delta x \mathbf{g}_k(E).$$

Since $\min(\mathcal{Q}_k(B), \mathcal{Q}_k(D)) > \mathcal{Q}_k(E)$, we obtain

$$\min(\mathcal{Q}_k(B), \mathcal{Q}_k(D)) - \mathcal{Q}_k(A) \geq \mathbf{g}_k(E)\Delta x. \tag{5.25}$$

By *Proposition 5.2.2*, $\mathcal{Q}_{k+1} \geq \mathcal{Q}_k$ for all the neighbor nodes B, C and D. By hypothesis, $\mathcal{Q}_{k+1}(A) = \mathcal{Q}_k(A)$ since node A is a direct parent of node E in Σ_k, and $\mathbf{g}_{k+1}(E) = \mathbf{g}_k(E)$. From (5.24),

$$\mathcal{Q}_{k+1}(A) \leq \mathcal{Q}_{k+1}(C), \tag{5.26}$$

and from (5.25)

$$\min(\mathcal{Q}_{k+1}(B), \mathcal{Q}_{k+1}(D)) - \mathcal{Q}_{k+1}(A) \geq \mathbf{g}_{k+1}(E)\Delta x. \tag{5.27}$$

Recalling a and b from (5.6),

$$\begin{aligned} a &= \min(\mathcal{Q}_{k+1}(B), \mathcal{Q}_{k+1}(D)), \\ b &= \min(\mathcal{Q}_{k+1}(A), \mathcal{Q}_{k+1}(C)) = \mathcal{Q}_{k+1}(A), \end{aligned}$$

we note that from (5.27), $\min(a, b) = \mathcal{Q}_{k+1}(A)$. Thus from (5.7)

$$\begin{aligned} \mathcal{Q}_{k+1}(E) =& \mathcal{Q}_{k+1}(A) + \mathbf{g}_{k+1}(E)\Delta x \\ =& \mathcal{Q}_k(A) + \mathbf{g}_k(E)\Delta x \\ =& \mathcal{Q}_k(E) \end{aligned} \tag{5.28}$$

which completes the proof for *Case 1.* □

Applying Proposition 5.2.3 from the node corresponding to the target $\mathbf{z}$, the following corollary follows.

Corollary 5.2.4. *Define the set*

$$\Upsilon := \{(l, m) \in \Psi : \ \mathbf{g}_{k+1}(l, m) > \mathbf{g}_k(l, m)\}. \tag{5.29}$$

For a node (i, j) and given the graph Σ_k, if $(i, j) \notin \Upsilon$, and if (i, j) is not a child of an element in Υ with respect to the graph Σ_k, then $\mathcal{Q}_{k+1}(i, j) = \mathcal{Q}_k(i, j)$.

Proof of Corollary 5.2.4. From Proposition 5.2.3, for any grid element $(i, j) \in \Psi$, $\mathcal{Q}_k(i, j) = \mathcal{Q}_{k+1}(i, j)$ if $(i, j) \notin \Upsilon$ and if $\mathcal{Q}_k(l, m) = \mathcal{Q}_{k+1}(l, m)$ for all direct parents (l, m) of (i, j). Starting from node $(0, 0)$, repeatedly applying Proposition 5.2.3 forwards through Σ_k, we conclude that if $(i, j) \notin \Upsilon$, and if (i, j) is not a child of the nodes in Υ, then $\mathcal{Q}_k(i, j) = \mathcal{Q}_{k+1}(i, j)$ which yields the desired conclusion. □

An example of the application of the corollary is shown in Figure 5.2. This figure shows the graph Σ_k. The black node is detected as an obstacle. The white nodes are not the children of the black node and thus they do not need recalculation. Since the grey nodes are the children of the obstacle, their values need to be updated.

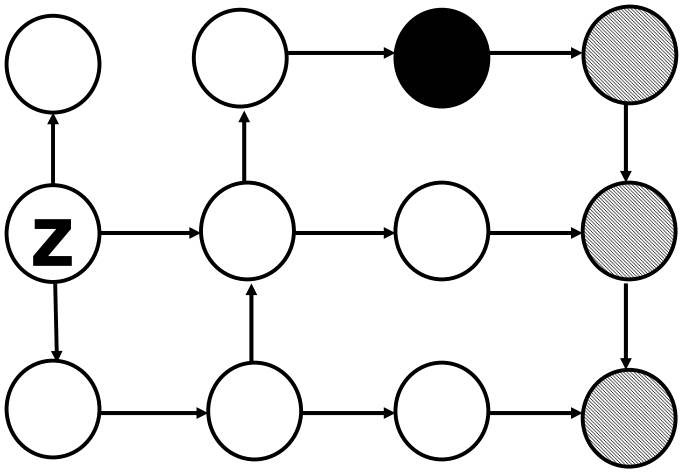

Figure 5.2: Given the graph Σ_k, the nodes that need to be recomputed are the black node and its children.

5.2.2 Algorithm

We now describe the proposed method to update level sets. From Corollary 5.2.4, we only need to recalculate the nodes that are children of the node whose cost has increased. Therefore, the first step is to identify these nodes, using, for example, the depth first search (see e.g. pp. 477, [9]). The second step is to recompute $\mathcal{Q}_{k+1}$ values for all children nodes. We employ the same principle of the fast marching method that propagates $\mathcal{Q}_{k+1}$ from smaller to larger values. Once the node corresponding to the location of the vehicle has been calculated,

then an updated optimal path exists and further update of the level set is not needed. Since obstacles are detected near the autonomous vehicle, the number of the nodes that must be recalculated is often very small.

Implementation for the Hybrid Receding Horizon Control

The dynamic fast marching method can be implemented to the case in which one needs to solve the global Eikonal equation in the hybrid receding horizon control proposed in Chapter 3. We state some remarks that shall be noted. First, in the hybrid receding horizon control, we discard the region corresponding to the newly detected obstacles when computing the new level sets. In so doing, we set their approximate level set values to be very large. Then, we can follow the analysis in *Proposition 5.2.2* and show that the level set values for the rest of the area monotonically increase as new obstacles are detected. Thus, the conclusions in both *Proposition 5.2.3* and *Corollary 5.2.4* hold. This indicates that the dynamic fast marching method is applicable to the hybrid receding horizon control proposed in Chapter 3. Second, since the hybrid receding horizon control requires to check one of the matching conditions (*Condition 1* in Section 3.4) which utilizes the gradient of the solutions of the global Eikonal equation, except for the nodes corresponding to the new obstacles, we need to recompute the level sets of each node in the domain for which level sets values need to be recomputed.

5.3 Level Set Updates When Empty Areas Are Detected

We present the dynamic fast marching method when the empty areas are detected. Thus, we assume that g_k monotonically decreases with the time sequence t_k. Following the discussion of Proposition 5.2.1 and Proposition 5.2.2, we can immediately conclude that the continuous

level set satisfies $Q_{k+1} \leq Q_k$ and that the corresponding discrete approximation satisfies $\mathcal{Q}_{k+1} \leq \mathcal{Q}_k$.

As in Section 5.2, we seek to identify the nodes that need to be recomputed. In parallel to Corollary 5.2.4, we infer the following corollary.

Corollary 5.3.1. *Suppose* $\mathbf{g}_k \geq \mathbf{g}_{k+1}$. *Define the set*

$$\Theta := \{(l,m) \in \Psi : \ \mathbf{g}_{k+1}(l,m) < \mathbf{g}_k(l,m)\}.$$

Denote by Σ_{k+1} *the new graph representing the dependence of new level set values* $\mathcal{Q}_{k+1}$. *For a node* (i,j), *if* $(i,j) \notin \Theta$, *and if* (i,j) *will not become a child of an element in* Θ *with respect to the graph* Σ_{k+1}, *then* $\mathcal{Q}_{k+1}(i,j) = \mathcal{Q}_k(i,j)$.

From Corollary 5.3.1, the nodes that need to be re-computed are the children of the nodes in Θ in the new value dependency graph Σ_{k+1}, as illustrated in Figure 5.3. We assume that an obstacle disappears at the black node at time t_{k+1}. If the graph from Figure 5.3 corresponds to the new value dependency graph Σ_{k+1}, the level set value of both black and grey nodes need to be recomputed. We also infer, from Corollary 5.3.1, that among all the nodes (i,j) for which $\mathcal{Q}_k(i,j) > \mathcal{Q}_{k+1}(i,j)$, the node with the smallest value $\mathcal{Q}_{k+1}$ is in Θ. Thus, if we update the level set values in the upwind direction we shall start recomputing the $\mathcal{Q}_{k+1}$ values for the nodes in Θ. In Figure 5.3, the black node is recomputed first. The hurdle that prevents us from directly identifying the nodes that will become the children of Θ is that the new graph Σ_{k+1} remains unknown before we re-compute the new level set. The following proposition allows us to start with $\mathcal{Q}_k$ and recompute the nodes that will become the children of Θ in the upwind directions.

Proposition 5.3.2. *Suppose* $\mathbf{g}_k \geq \mathbf{g}_{k+1}$. *Let* Σ_{k+1} *be the graph of the computational dependence corresponding to* $\mathcal{Q}_{k+1}$. *Consider a node* $(i,j) \in \Psi$ *satisfying*

$$\mathcal{Q}_{k+1}(i,j) < \mathcal{Q}_k(i,j).$$

Let (m,n) be any neighbor of (i,j). We define the intermediate term $\mathcal{V}(m,n)$ satisfying

$$\begin{cases} \mathcal{V}(m,n) = \mathcal{Q}_{k+1}(m,n), & \text{if } (m,n) \text{ is a direct parent} \\ & \text{of } (i,j) \text{ in } \Sigma_{k+1}, \\ \mathcal{V}(m,n) \geq \mathcal{Q}_{k+1}(m,n), & \text{otherwise.} \end{cases} \tag{5.30}$$

Then, the solution $\mathcal{Q}_{k+1}(i,j)$ corresponding to $\mathbf{g}_{k+1}$ can be computed by solving $\mathcal{Q}_{k+1}(i,j)$ satisfying the following equation

$$\begin{aligned} &\max\left(\frac{\mathcal{Q}_{k+1}(i,j) - \min(\mathcal{V}(i-1,j), \mathcal{V}(i+1,j))}{\Delta x}, 0\right)^2 \\ &+ \max\left(\frac{\mathcal{Q}_{k+1} - \min(\mathcal{V}(i,j+1), \mathcal{V}(i,j-1))}{\Delta x}, 0\right)^2 \\ &- \mathbf{g}_{k+1}^2(i,j) = 0, \quad \text{if } (i,j) \neq (0,0). \end{aligned} \tag{5.31}$$

From Proposition 5.3.2, as long as the intermediate terms $\mathcal{V}$ satisfies (5.30) for each grid element, we can always correctly compute $\mathcal{Q}_{k+1}(i,j)$. To define such $\mathcal{V}$ values, we initially choose $\mathcal{V} = \mathcal{Q}_k$ for each node. During the re-computation in the upwind direction, for any grid element (i,j), we always replace $\mathcal{V}(i,j)$ value with $\mathcal{Q}_{k+1}(i,j)$ as soon as it is available. This guarantees that the inequality in (5.30) always holds since $\mathcal{V}(i,j)$ value of a grid element is either $\mathcal{Q}_k(i,j)$ or $\mathcal{Q}_{k+1}(i,j)$. The equation in (5.30) holds too because in the upwind direction if a node (l,m) becomes a parent of (i,j), $\mathcal{Q}_{k+1}(l,m)$ value must be recomputed before $\mathcal{Q}_{k+1}(i,j)$. Thus, $\mathcal{V}(l,m)$ must be equal to $\mathcal{Q}_{k+1}(l,m)$ as we keep $\mathcal{V}$ values up to date. Therefore, from (5.30) and (5.31), we infer that we can start from initially setting $\mathcal{Q}_k$ as $\mathcal{V}$ in (5.30) and recompute the nodes according to (5.31) in the upwind directions.

Proof of Proposition 5.3.2. The proof is indeed a direct application of the upwind property of the first order scheme (5.5) which supports the principle of the conventional fast marching method. We only detail the proof for *Case 1.* The proof for *Case 2* follows a similar procedure. For notational simplicity, we define $E := (i,j)$ and neighbors of E by A, B, C

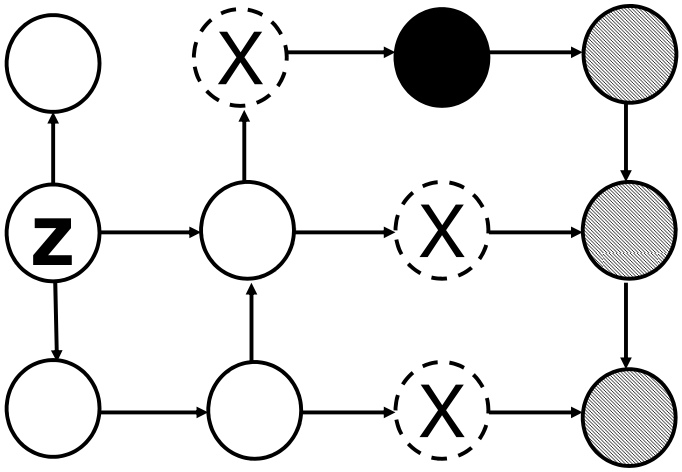

Figure 5.3: The black node is detected as a new empty area and the grey nodes are its children in Σ_{k+1}.

and D as shown in Figure 5.1 (a). Thus, from the first order scheme (5.5), we have

$$\begin{aligned} &\max\left(\frac{\mathcal{Q}_{k+1}(E)-\min(\mathcal{Q}_{k+1}(A),\mathcal{Q}_{k+1}(C))}{\Delta x},0\right)^2 \\ &+\max\left(\frac{\mathcal{Q}_{k+1}(E)-\min(\mathcal{Q}_{k+1}(B),\mathcal{Q}_{k+1}(D))}{\Delta x},0\right)^2 \\ &-\mathbf{g}^2_{k+1}(E)=0. \end{aligned} \tag{5.32}$$

We assume that A is the only direct parent of E in the graph Σ_{k+1}. Thus, the solution of $\mathcal{Q}(E)$ satisfies

$$\mathcal{Q}_{k+1}(E)=\mathcal{Q}_{k+1}(A)+\mathbf{g}_{k+1}(E)\Delta x. \tag{5.33}$$

From (5.32), we infer

$$\mathcal{Q}_{k+1}(A)\leq\mathcal{Q}_{k+1}(C), \tag{5.34}$$

and

$$\mathcal{Q}_{k+1}(E)-\min(\mathcal{Q}_{k+1}(B),\mathcal{Q}_{k+1}(D))\leq 0. \tag{5.35}$$

We must also have $\mathcal{Q}_{k+1}(A) < \mathcal{Q}_{k+1}(E)$, since A is the parent of E. Thus, according to (5.30), we obtain

$$\mathcal{Q}_{k+1}(A) = \mathcal{V}(A). \tag{5.36}$$

For the other neighbor nodes B, C and D, we infer, by the definition of $\mathcal{V}$ in (5.30), that

$$\mathcal{Q}_{k+1}(F) \leq \mathcal{V}(F), \quad \forall\ F \in \{B, C, D\}. \tag{5.37}$$

Thus, from (5.34), (5.36), and (5.37), we infer that

$$\mathcal{V}(A) = \mathcal{Q}_{k+1}(A) \leq \mathcal{Q}_{k+1}(C) \leq \mathcal{V}(C). \tag{5.38}$$

From (5.38),

$$\begin{aligned} &\max\left(\frac{\mathcal{Q}_{k+1}(E) - \min(\mathcal{V}(A), \mathcal{V}(C))}{\Delta x}, 0\right)^2 \\ =&\max\left(\frac{\mathcal{Q}_{k+1}(E) - \mathcal{V}(A)}{\Delta x}, 0\right)^2 \end{aligned} \tag{5.39}$$

From (5.35) and (5.37),

$$\begin{aligned} &\mathcal{Q}_{k+1}(E) - \min(\mathcal{V}(B), \mathcal{V}(D)) \\ \leq&\mathcal{Q}_{k+1}(E) - \min(\mathcal{Q}_{k+1}(B), \mathcal{Q}_{k+1}(D)) \\ \leq&0. \end{aligned} \tag{5.40}$$

Given (5.39) and (5.40), we conclude that the solution satisfying (5.31) is

$$\mathcal{Q}_{k+1}(E) = \mathcal{V}(A) + \mathbf{g}_{k+1}(E)\Delta x. \tag{5.41}$$

From (5.36) and (5.33), we conclude the assertion. □

5.3.1 Algorithm

Following the above discussion, we now state the principal steps for efficiently updating the new level sets. Note that from Corollary 5.3.1, the nodes need to be recomputed are either in

Θ corresponding to the nodes detected as new empty area or will become the future children of Θ. Unlike the algorithm proposed in Section 5.2.2, we have to identify these nodes during the re-computation. Note that we start recomputing the trial values of $\mathcal{Q}_{k+1}$ for the nodes in Θ by letting $\mathcal{V} = \mathcal{Q}_{k+1}$ everywhere. Picking up the one with the smallest trial value, we start re-computation in the upwind direction. Values of $\mathcal{V}$ are updated throughout the computation and we assign new values for $\mathcal{Q}_{k+1}$ as in (5.31). Due to Corollary 5.3.1, after we recompute the new value of a node, we check if this node becomes a child of Θ. If not, we do not propagate its value to the neighbors. In so doing, we stop propagating the new level set at the nodes that are on the border between those who are the children of Θ and those who are not. For example, as shown in Figure 5.3, the algorithm will stop propagating the level set value for the nodes marked as "X".

5.4 Illustrations

To illustrate the principal conclusions in this chapter, we show two examples. First, in a canonical example, we show how level sets are influenced by an unexpected obstacle. Second, we show a simulation result for an autonomous surface vehicle (ASV) navigating in a riverine environment. In this section, we show only the case when new obstacles are detected.

A Canonical Example

To illustrate how an obstacle can influence the level set values, we show a canonical example for which an ASV is navigating in an open area. Consider a $1000m \times 1000m$ open area which is represented by a map whose grid size is $1m \times 1m$. We assume that the goal location is at the center of the square and that the cost function is $\mathbf{g}_0 = 1$ for each grid element except at the goal where $\mathbf{g}_0 = 0$. Figure 5.4 show the corresponding level sets $\mathcal{Q}_0$ before the detection of the new obstacles. We suppose that at time t_1 an obstacle is detected at location $[250m, 250m]^T$. Then, $\mathbf{g}_1 = 10^7$ at that location and $\mathbf{g}_1 = \mathbf{g}_0$ elsewhere. Figure

5.5 shows the difference between $\mathcal{Q}_0$ and $\mathcal{Q}_1$. The area included in the dash lines are where the level set values are recomputed because the corresponding nodes of the level set are the children of the node that corresponds to the obstacle. Note that changes in the level set value are extremely small except in the area for which the obstacle occludes the target. It is this area in which an path would need to be altered due to presence of the obstacle.

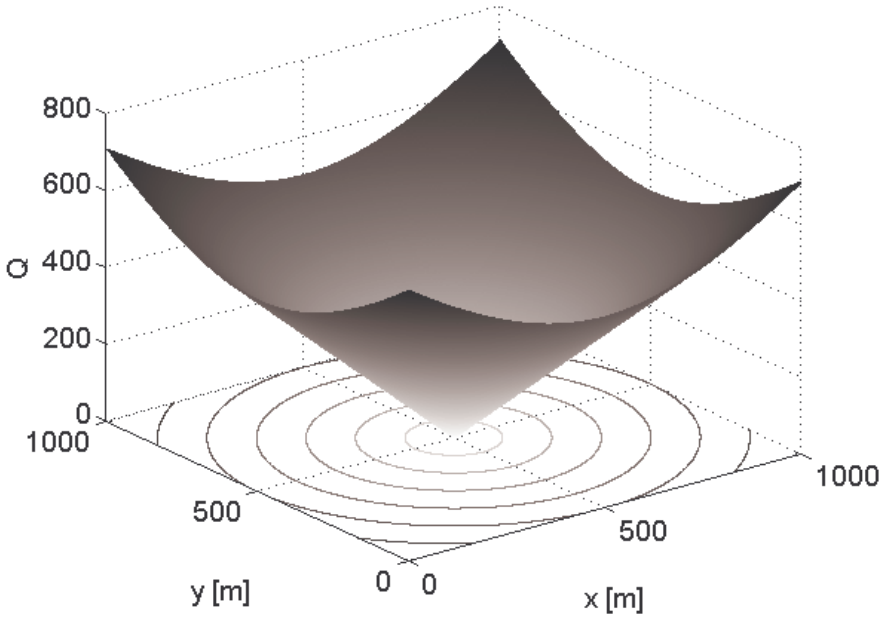

Figure 5.4: The old level set values.

The principal tasks are to identify the children of the nodes for which the risk value has changed and to recalculate a subset of the level set values. Let H be the number of child nodes and let K be the number of the nodes that are recalculated. The computational cost to identify the children of obstacles is $O(H)$ (see e.g., pp. 477 [9]) and the cost to sort and recalculate K nodes is $O(K \lg K)$ (see e.g., pp.140 [9]). Since H and K depends on environment changes and the autonomous vehicle locations during the mission, the execution time varies for different scenarios. However, the newly detected obstacles are often close to the vehicle and thus the number of nodes K that need to be recomputed is small in

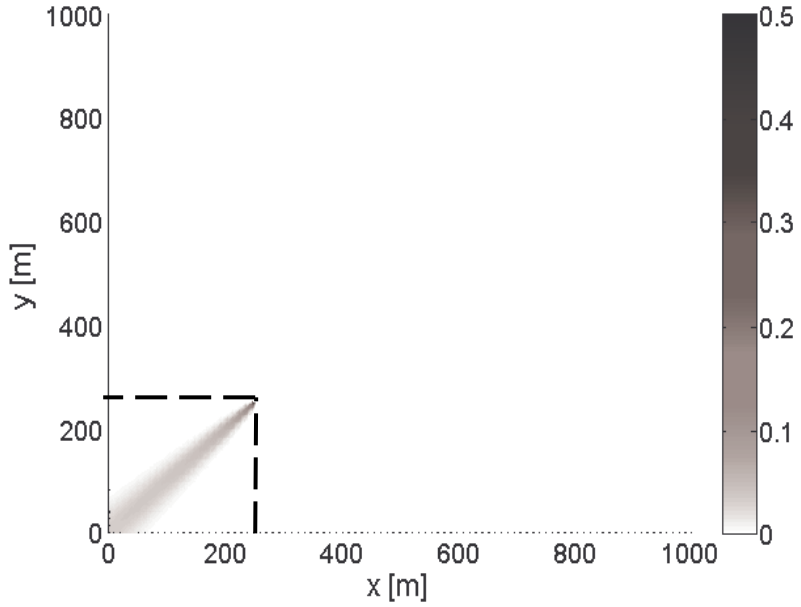

Figure 5.5: The contour of the difference between the new and the old level set values.

most cases. To illustrate this point, we conduct several other tests by locating obstacles to different positions. The numerical tests are conducted on a PC with one single Intel Pentium M 1.86 GHz processor and 1 GB RAM. In these tests, we compare the execution time of the proposed method to that of recalculating the entire domain by using conventional fast marching method. In Table 5.1, we can see that the execution time varies depending on the number of obstacle children H and recalculated nodes K. Table 5.1 shows the execution time for both using the proposed method and recalculating the level sets over the entire domain. It takes about 13 seconds to recalculate the level sets for the entire domain whereas the proposed method takes less than 1 second to update in most cases. The dynamic fast marching method saves more than 90% of the computation time.

Table 5.1: Execution Time for Different Obstacles and ASV Locations

Total Number of Nodes (N): 10^6.

Execution Time to Calculate the Entire Level Set: 13.25 secs.

Obstacle Location	ASV Location	Children of Obstacle (H)	Nodes Recalculated (K)	Percentage (K/N)	Execution Time (secs)
$[250m, 250m]^T$	$[214m, 214m]^T$	62,001	2,560	0.138%	0.22
$[250m, 500m]^T$	$[200m, 500m]^T$	248,502	11,130	1.113%	0.70
$[499m, 499m]^T$	$[463m, 463m]^T$	248,004	2,142	0.2142%	0.56
$[499m, 500m]^T$	$[449m, 500m]^T$	497,004	4,030	0.4030%	1.25

An Example for ASV Navigation

We show an example of an ASV navigating in a riverine environment. Figure 3.1 represents the *a priori* map which spans an area of $800m \times 600m$. In Figure 5.6, the grey area represents obstacles that do not appear in the *a priori* map. The grid size is $3m \times 3m$. We assume that the ASV detects obstacles up to 35 meters range. There are eight occasions where an obstacle is detected on the path and a portion of the level set is recomputed. The location of each path update event is indicated in Figure 5.6. Between the 5th to 7th update events, the ASV entered a U-shaped trap taking the route as a shortcut to the goal. The ASV computes a correct path to the goal at the 8th update event when it detects the dead end of the trap.

As an example to show how level sets are updates partially in this application, in Figure 5.7, the shaded area shows the locations of the children of the newly detected obstacle. However, the level set needs to be recomputed for only $K < H$ of these nodes, as shown in Figure 5.8. Figures 5.7 and 5.8 illustrate that only a small portion of the nodes are updated. For each update event shown in Figure 5.6, Table 5.2 lists the number of the nodes which are detected as obstacles, children of the obstacles H and nodes which are recomputed K before the node corresponding to the location of the vehicle has been calculated. The last column of the table is the ratio between the number of the nodes that are recalculated and the total number of the nodes. This column shows that the proposed method significantly reduces the computation cost compared to computing the level sets values over the entire

domain. Since the trap scenario at the 8th update is more complicated, the number of the nodes recalculated is correspondingly the largest among the eight update events in Table 5.2.

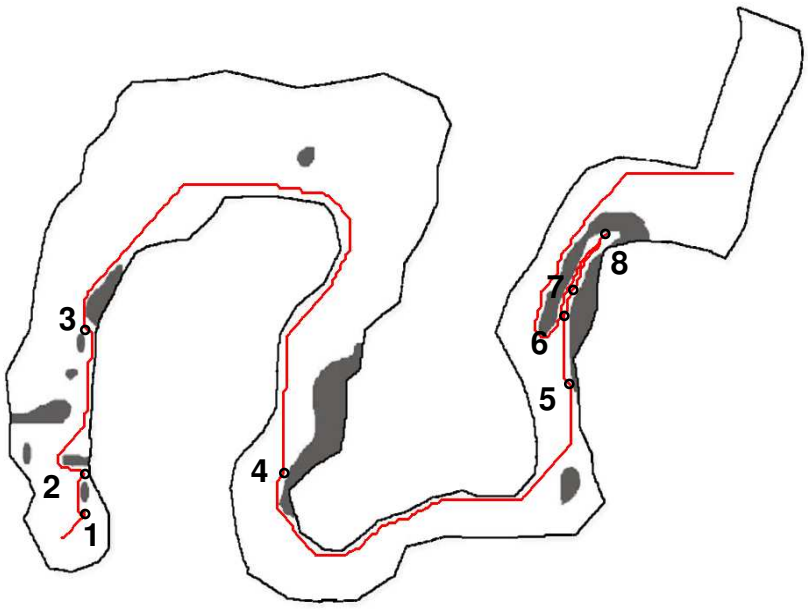

Figure 5.6: The grey colored areas are some unexpect obstacles due to the inaccuracy and incompleteness of the *a priori* map in Figure 3.1. The actual trajectory is marked by a red line.

5.5 Concluding Remarks

In this chapter, we propose an efficient algorithm that uses the level set method to replan paths. Although the fast marching method is a fast approach to path planning with level sets, even this method can be prohibitively slow for very large areas. The proposed method reduces computation expenses in two aspects. First, we do not update level sets unless there are obstacles on the current optimal trajectory of the vehicle. Second, when we update level

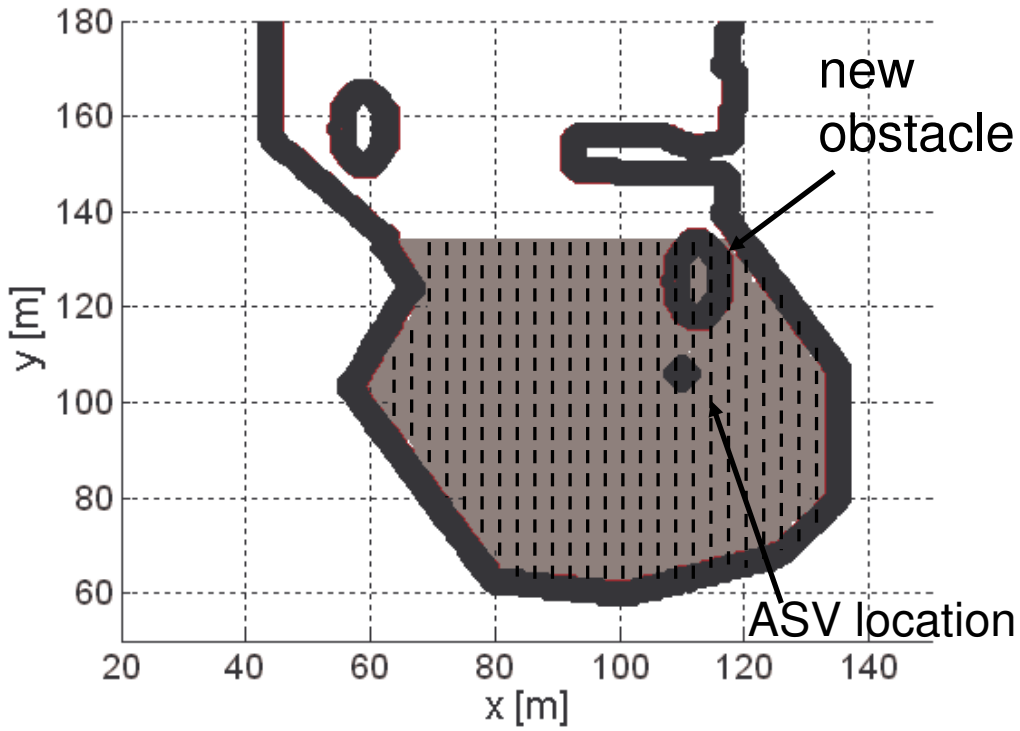

Figure 5.7: The shaded area correponds to the nodes that level set values need to be recomputed.

sets, we recompute only a portion of them. In order to justify the proposed method, we provide formal analysis of how level sets change when new obstacles or new empty areas are detected. The proposed method can function as an independent path planner and it can also be employed for the hybrid receding horizon control in Chapter 3 when solving the global Eikonal equation over the entire domain is necessary.

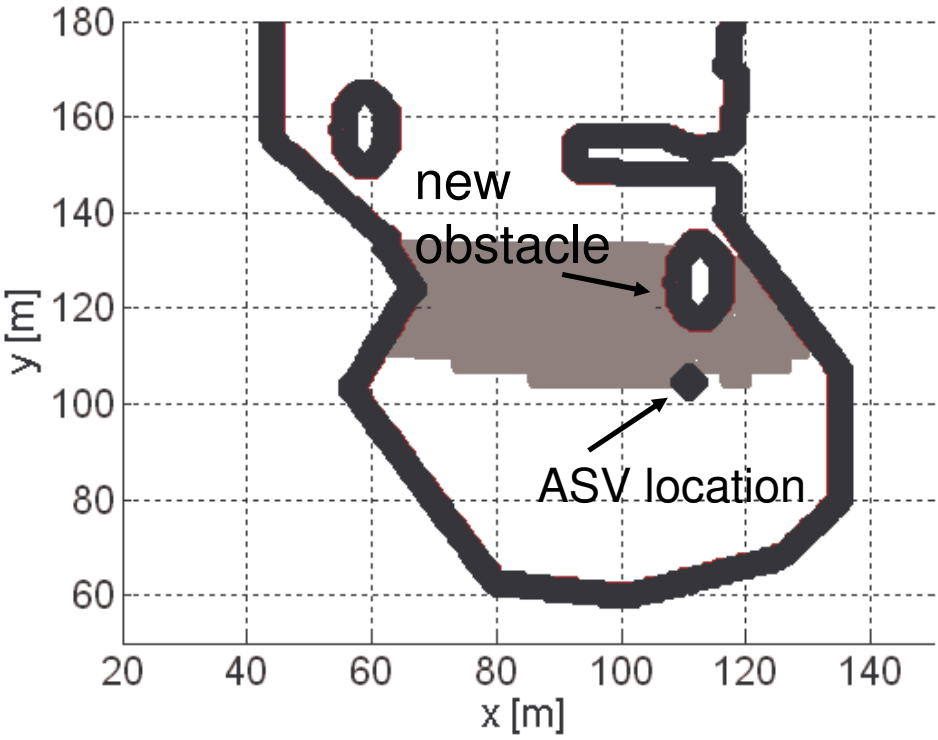

Figure 5.8: The grey area correponds to the nodes that level set values are recomputed.

Table 5.2: Computational Cost of Dynamic Fast Marching Method for ASV Navigation Example

Update	Number of Obstacles	Children of Obstacles (H)	Nodes Recalculated (K)	Percentage (K/N)
1	109	443	119	1.01 %
2	142	251	141	1.19%
3	757	1132	173	1.47%
4	811	6105	144	1.22%
5	413	2029	973	8.26%
6	306	1211	117	0.99%
7	365	420	100	0.85%
8	558	420	1940	16.48%
Total Number of Nodes (N): 11775				

Chapter 6

Experiments

In this chapter, we present the results of experimental field trials of the proposed dynamic fast marching method in which an autonomous surface vehicle (ASV) navigates in a riverine environment. A description of the ASV and the vision algorithms that were used to detect features in a riverine environment are briefly reviewed in Section 6.1. In Section 6.2, the experimental results are presented to demonstrate the effectiveness of the proposed method.

6.1 Autonomous Surface Vehicle

The ASV is designed to operate in a shallow-water environment. It can support experiments in long-term autonomous operations in unstructured and dynamic environments. Figure 6.1 shows the structural layout and components of the ASV. The basic platform of the ASV is an inflatable pontoon hull which has been modified to include a motor mount, a camera mount, and three watertight boxes. The platform is capable of varying 110 kilograms of payload. When excess payload is used for fuel, the ASV can operate continuously for 3-4 days. Three watertight boxes house computers and navigation sensors, a motor controller and a generator, respectively. Two electric thrusters provide propulsion and differential steering. The

ASV is equipped with a GPS receiver and a three-axis gyro as navigation sensors. An omnidirectional camera on the ASV detects hazards to navigation, including shoreline, docks, buoys, and unstructured obstacles. Two laptops perform the computational tasks required by the ASV. Control algorithms including waypoint tracking are implemented on one laptop. Machine vision and navigation algorithms are implemented on the other. In this work, the two laptops will be referred as the control laptop and vision laptop, respectively. Communication with a remote base station is accomplished through a mesh network, enabling an operator to monitor the system performance and have remote control of the ASV if necessary. Table 6.1 shows the specifications of the vehicle.

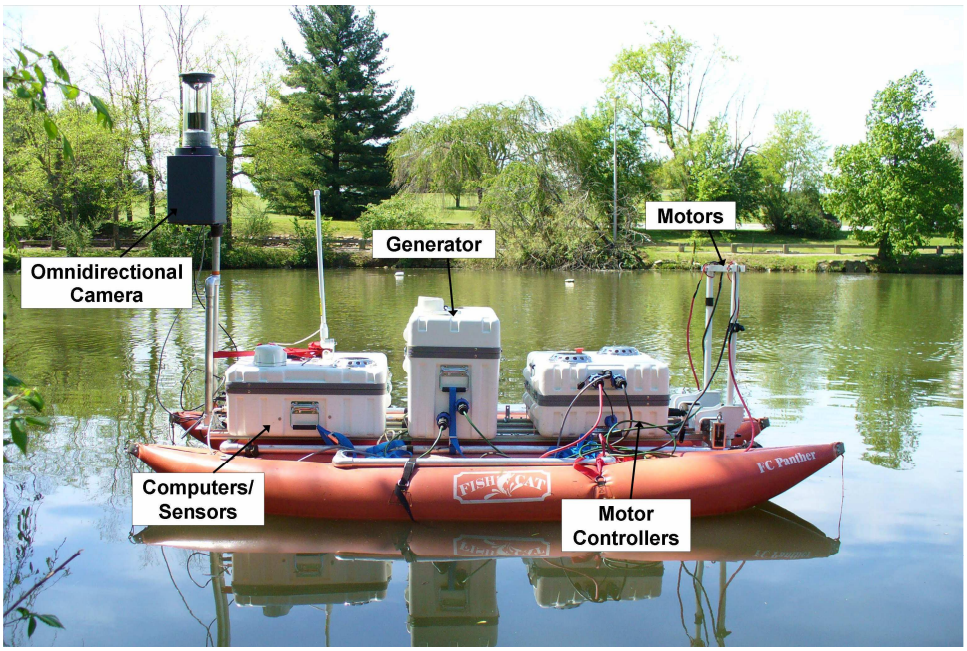

Figure 6.1: Configuration of the ASV.

Obstacle Detection: Vision System

An omnidirectional camera captures 360^o images of the environment at a rate of 5 Hz. Figure 6.2 shows an example of images that are taken by the camera. Vision algorithms

Table 6.1: Specifications of the ASV

Dimensions	2.7 $m \times 1.5$ m
Weight	125 kg
Cruise Speed	$2-3$ knots
Payload	110 kg
Endurance	$3-4$ days at full throttle
Computer control system	Two PC laptops running Visual C++ and Labview
Communication	2.5 GHz mesh network
Navigation sensors	• WAAS-GPS • Gyro-stabilized heading, pitch and roll • Depth • Single omnidirectional camera for geometric shoreline reconstruction

were developed in [21] to detect, track, and localize obstacles. The algorithms are described in detail in [21], [22], [61], and [70]. Figure 6.3 shows an experimental result conducted in Peak Creek, VA. The entire run is about 3 km. As shown in the figure, the red points are the obstacle features detected by the camera.

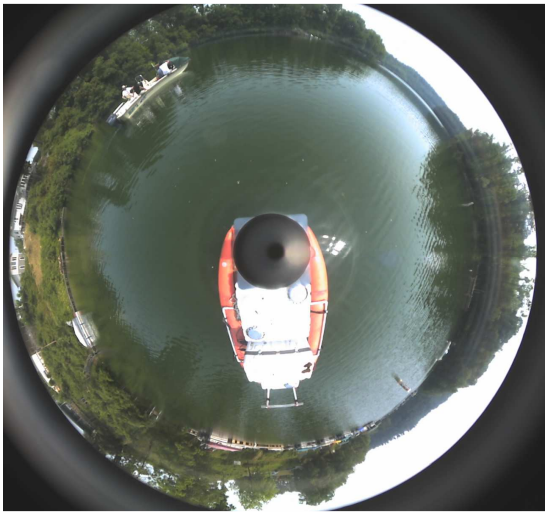

Figure 6.2: An image taken by the omnidirectional camera.

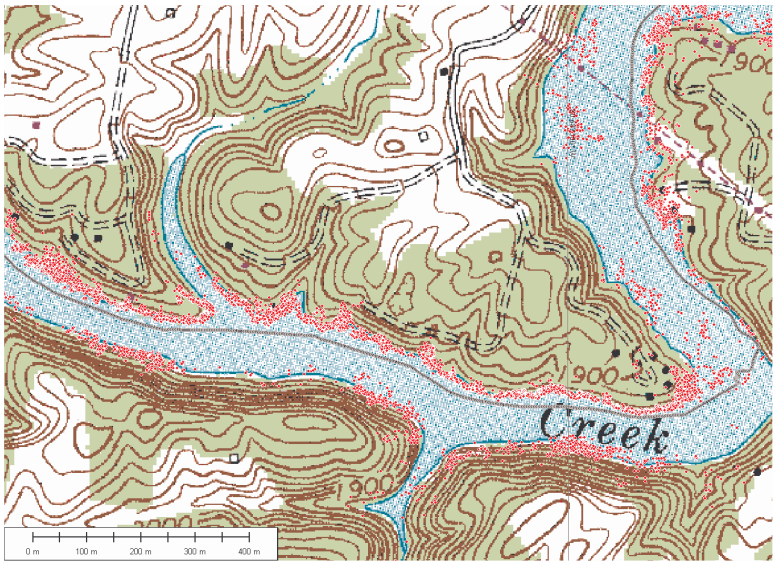

Figure 6.3: The results of online feature estimation [21]: the red dots are the features localized.

Occupancy Map Update

We employ an probabilistic occupancy map to model the environment (see e.g. [11]). Occupancy maps are widely used for robotic navigation applications, see, for example, [49], [52], and [73].

The environment is represented by a two dimensional uniform grid for which each element of the grid is associated with a probability of occupancy. We say that if a location is not safely navigable, then it is occupied. Given an *a priori* map, we can generate an *a priori* occupancy map. For example, Figure 6.4 illustrates an *a priori* occupancy map that is generated by using the map of Peak Creek shown in Figure 6.5, VA. In this occupancy map, we set the occupied probability of shores to be 1. Because most of the obstacles such as docks are built close to shore, we set the *a priori* probability higher, say 0.6, for the water part that is within 15 meters to the shore. The probability for the rest of the water part is 0.5. Feature localization is then modeled as a stochastic process such that the probability of occupancy grids will be updated by a standard Bayesian update rule. For example, Figure

6.6 shows a closeup of a part of the *a priori* map and its corresponding occupancy map. In Figure 6.7, it shows that after the vehicle detects new obstacles (red features) in the same area, the occupancy map is then updated. In the experiment, the occupancy map is updated at the rate of 0.2 Hz.

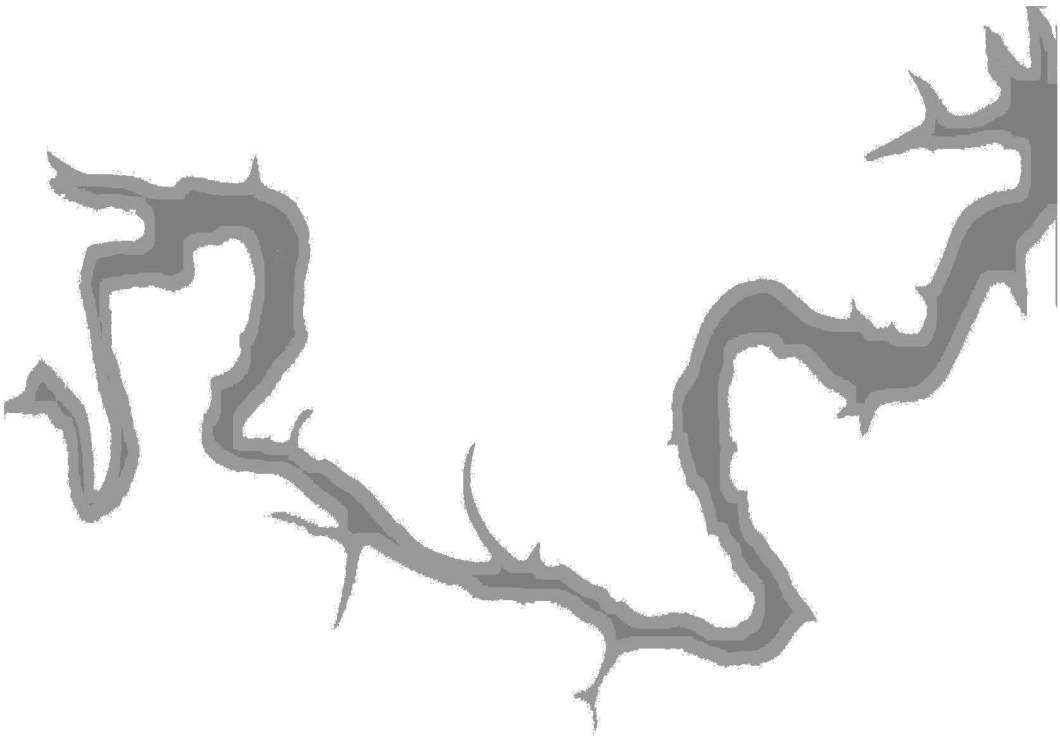

Figure 6.4: The *a priori* occupancy map: the color changes from dark gray to white color indicate that the occupied probability increases from 0.5 to 1.0.

6.2 Experimental Results

Replanning Strategy

The replanning strategy is schematized in Figure 6.8. It runs at a rate of 0.2 Hz, the same rate as the map updating frequency. For time t_{k+1}, the strategy stacks a grid (i, j) into

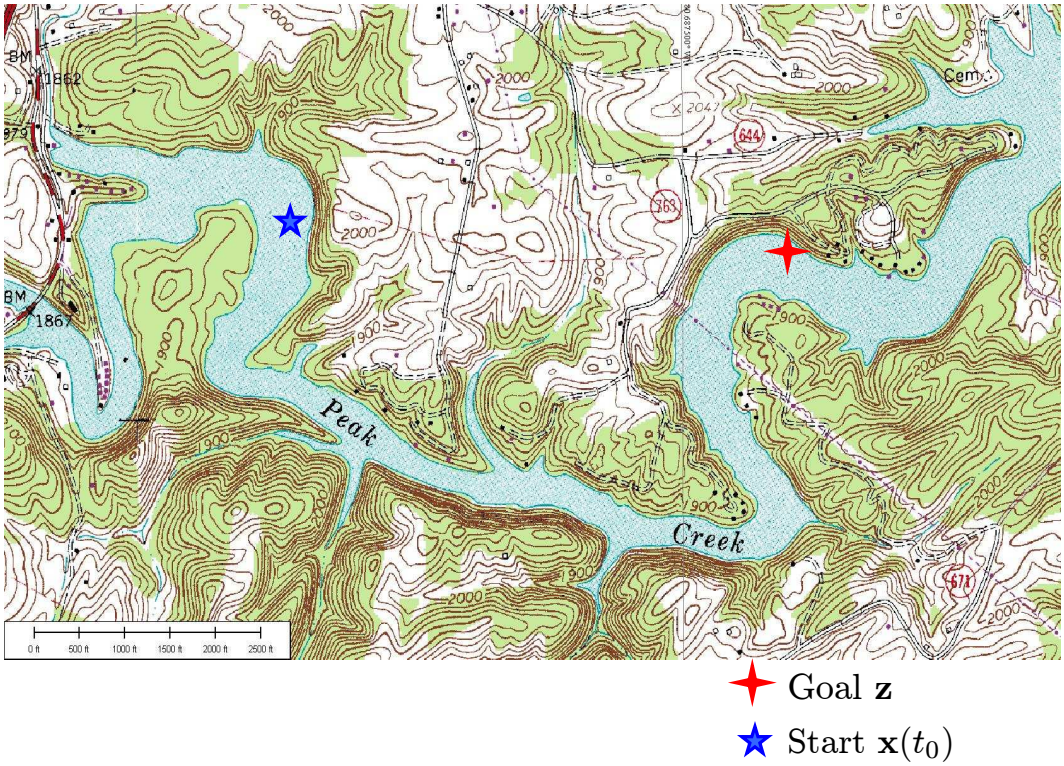

Figure 6.5: The *a priori* map for Peak Creek, VA: the start and the goal are marked by blue and red stars respectively.

a queue of the newly detected obstacles if its cost function satisfies $\mathbf{g}_{k+1}(i,j) > \mathbf{g}_k(i,j)$. However, the strategy does not update the level sets and it does not seek for a new path unless there is an element in the obstacle queue on the vehicle's current path. If such scenario happens, we call the dynamic fast marching method to update the solutions of new Eikonal equations which accounts for unexpected obstacles only. After updating level sets, we clear the obstacle queue. In the worst case, if we do not find new feasible paths by considering the new obstacles only, we update the level sets taking account of the newly detected empty areas by the dynamic fast marching method proposed in Section 5.3. This process is repeated until the vehicle reaches the goal.

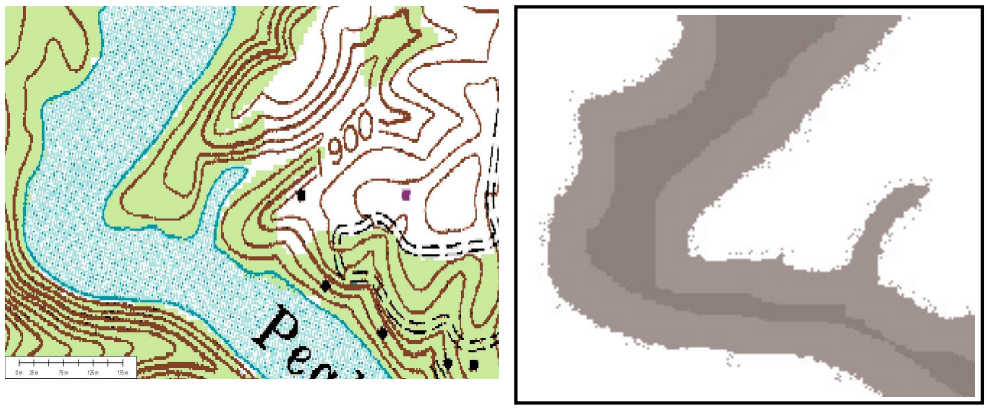

Figure 6.6: A closeup of the *a priori* map and its occupancy map. The color changes from dark gray to white indicate that the occupied probability increases from 0.5 to 1.0.

Field Trial

We deployed the ASV in Peak Creek, VA, and we selected a destination approximately four kilometers away from a starting location (Figure 6.5). Given the *a priori* occupancy map in Figure 6.4, we used the dynamic fast marching method to compute the *a priori* level sets and find an initial path. Both are shown in Figure 6.9.

The ASV's forward velocity was at about 2.5 knots. It managed to reach the target in about 45 minutes without any human interaction. Figure 6.10 shows the entire ASV course and the new features that were detected along the route. Since we set the cost function $\mathbf{g}_k$ higher for the area close to the shore than that in the middle, the ASV traveled mostly in the middle of the river. The path was replanned for a total number of four times during

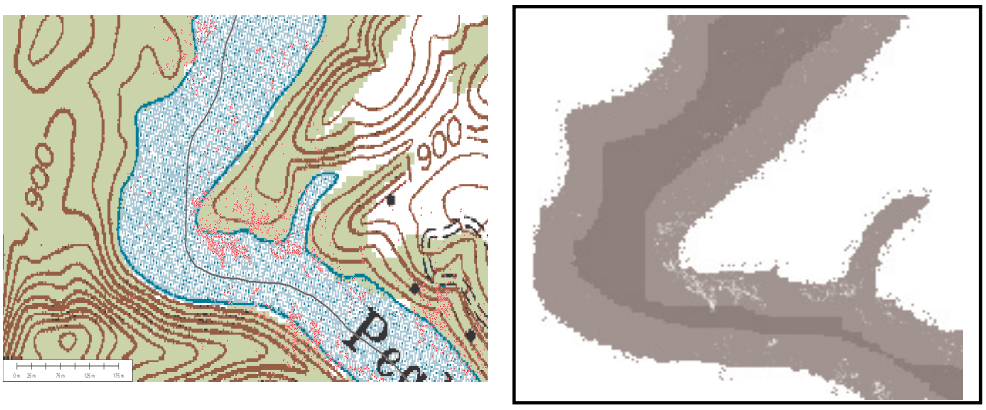

Figure 6.7: The unexpected obstacle features are marked in red in the left figure. The right figure is the updated occupancy map of the corresponding area. The color changes from dark gray to white indicate that the occupied probability increases from 0.5 to 1.0.

the entire mission. The locations where the replanning occurred are marked in the final occupancy map in Figure 6.11. Among these four updates, the first three updates were caused by the detection of the chase boat as shown in Figure 6.12. Figure 6.13 is a closeup that shows that upon the detection of the new obstacles, the ASV successfully replanned its path which would potentially cause the collisions and detoured the unexpected obstacles. The 4th update was caused by the dock that is marked in Figure 6.14. A closeup of the replanning at the corresponding location is shown in Figure 6.15.

Note that the vision laptop that replans paths is supplied with Centrino Duo Core 1.83 GHz and 1 GB RAM. The computation expenses of these four updates are listed in Table

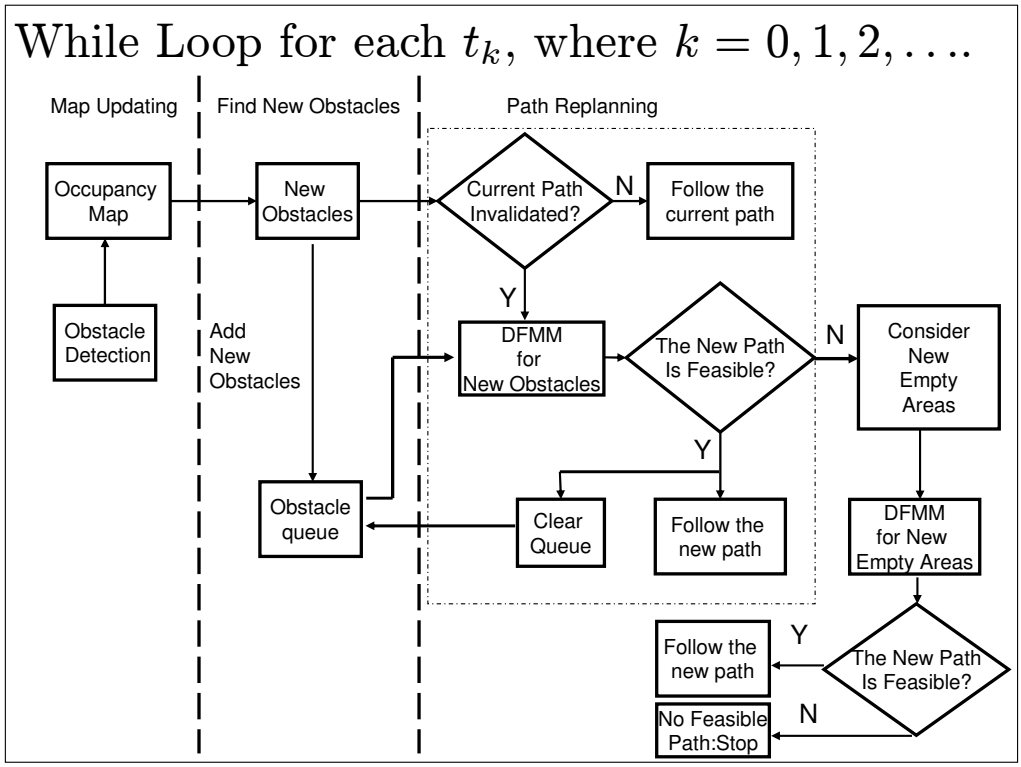

Figure 6.8: Path replanning strategy.

6.2. The table shows that to calculate level sets values for the entire domain takes about 3 seconds whereas the proposed method takes only 0.016 to 0.3 seconds to update the level sets. The last column of the table illustrates that each replanning by using the proposed dynamics fast marching method costs as low as 0.53% and less than 10% computational cost of the conventional fast marching method.

6.3 Concluding Remarks

We have presented a field trial in which the proposed dynamic fast marching method is evaluated for an ASV navigating in a riverine environment. Note that in this experiment, since the ASV traveled in the middle of the river and there were few obstacles, the obstacle

Figure 6.9: The initial level sets for the *a priori* Map: the level sets values increase from dark to light color. The white line represents the *a priori* trajectory.

avoidance occurred four times only. Nevertheless, through this experiments, it is validated that the proposed method is often faster than the conventional fast marching method.

As shown in [80], a single omnidirectional camera functions as a bearing sensor and, thus, is not capable of detecting features that are along the heading direction of the ASV. In [19], the dynamic fast marching algorithm proposed herein was implemented for an ASV equipped with a laser line-scanner which measures bearing and range to obstacles. In order to validate the proposed method, a large portion of the shores was purposely removed in the *a priori* map used for path planning. The missing portions of the shoreline were chosen so that the ASV would directly collide with the shores without replanning the routes. As the result, the ASV detected these missing shorelines, quickly replanned new obstacle-free paths,

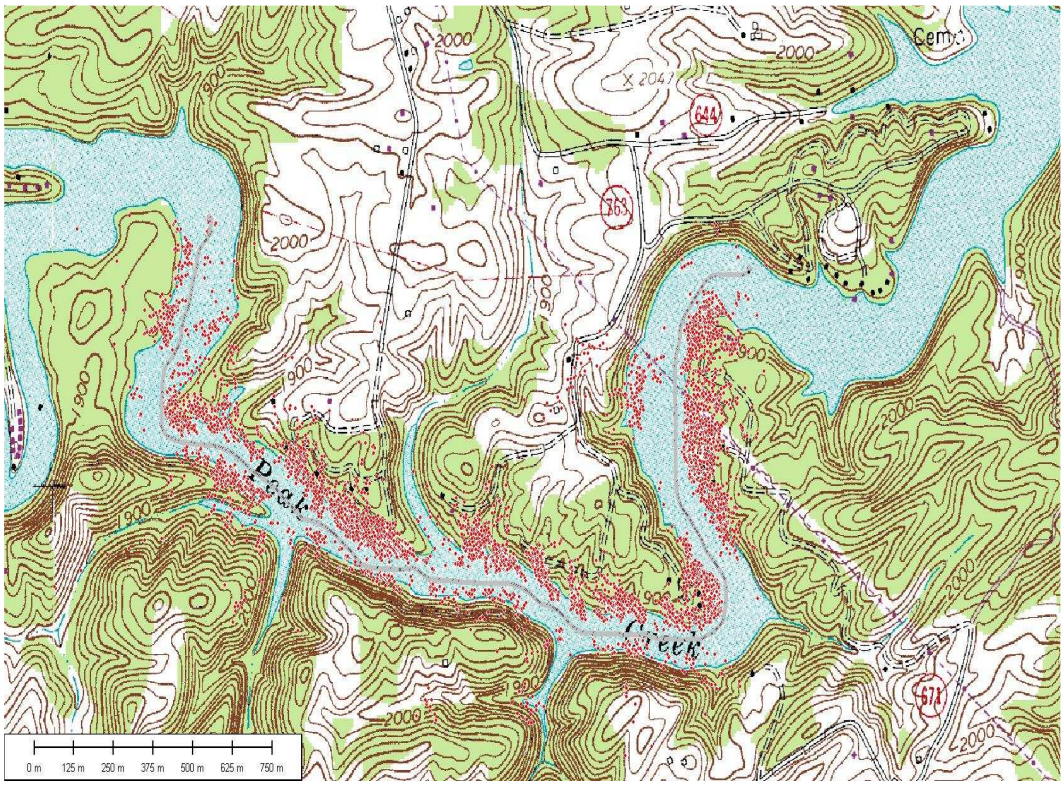

Figure 6.10: The black line is the ASV trajectory and the red dots are the detected features.

and successfully reached a predefined goal location. The history of the detected shorelines and the actual trajectory of the ASV following the replanned paths are plotted in Figure 6.16. This experiment further validates the effectiveness of the proposed method for fast path replanning in complex and uncertain riverine environment.

Figure 6.11: The final occupancy map: the color changes from dark gray to white indicate that the occupied probability increases from 0.5 to 1.0.

Table 6.2: Computational Cost of Dynamic Fast Marching Method for ASV Navigation Field Trial

unit: number of nodes

Update	Number of Obstacles	Children of Obstacles (H)	Accepted Nodes (K)	Replanning Time (secs)	Percentage
1	8	3627	37	0.016	0.53%
2	4	313	26	0.016	0.53%
3	11	1231	1206	0.031	1.03%
4	3243	132177	3678	0.297	9.8%
Execution Time to Calculate the Entire Level Set: 3.020 secs.					

Figure 6.12: The chase boat marked by red retangular was detected as obstacles.

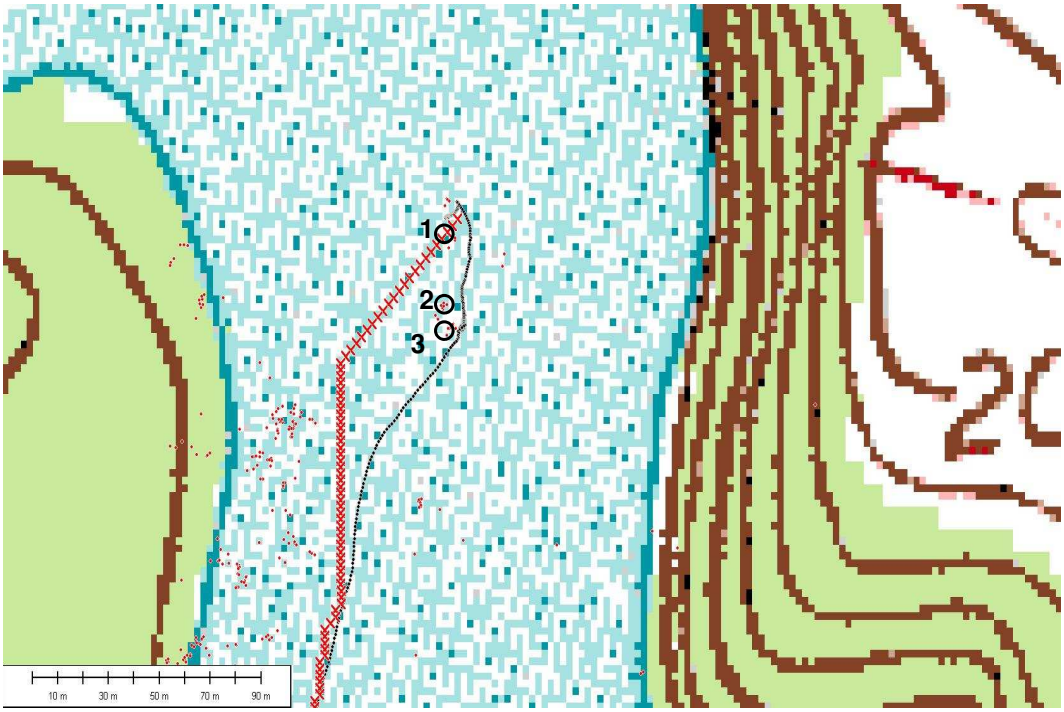

Figure 6.13: The red dots are the detected obstacles. The blacks circles represent the locations of the unexpected obstacles that caused the first three replannings. The trajectory composed of red "X" is the planned trajectory before the first replanning and the black line is the ASV actual trajectory.

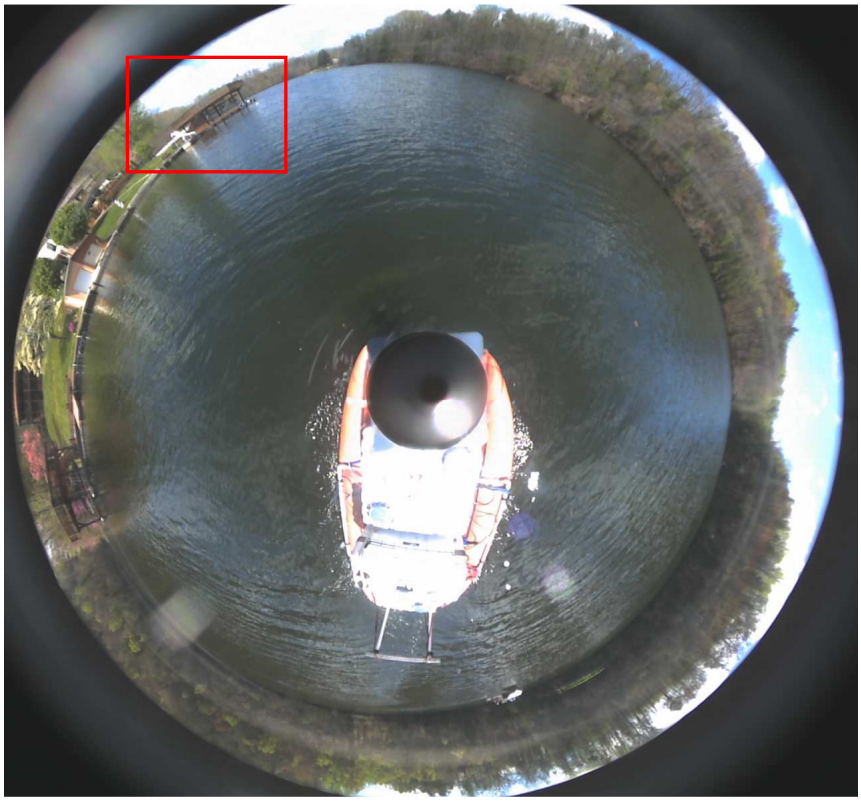

Figure 6.14: The dock marked by red retangular was detected as obstacles.

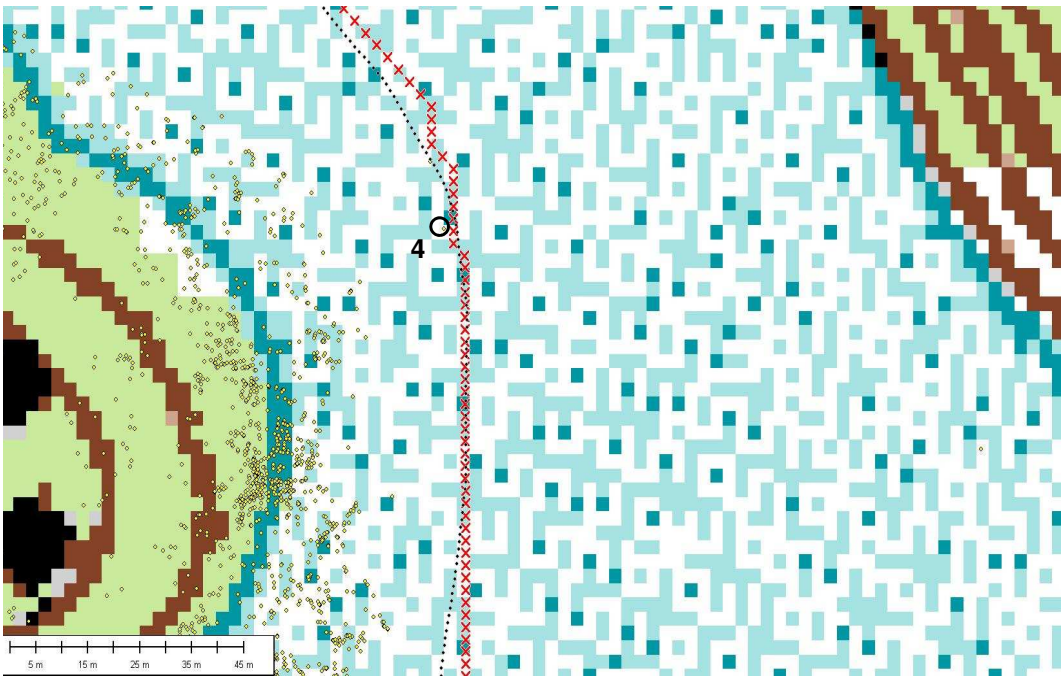

Figure 6.15: The yellow dots are the detected obstacles. The black circle represents the locations of the unexpected obstacles that caused the 4th replanning. The trajectory composed of red "X" is the planned trajectory before the fourth replanning and the black line is the ASV actual trajectory.

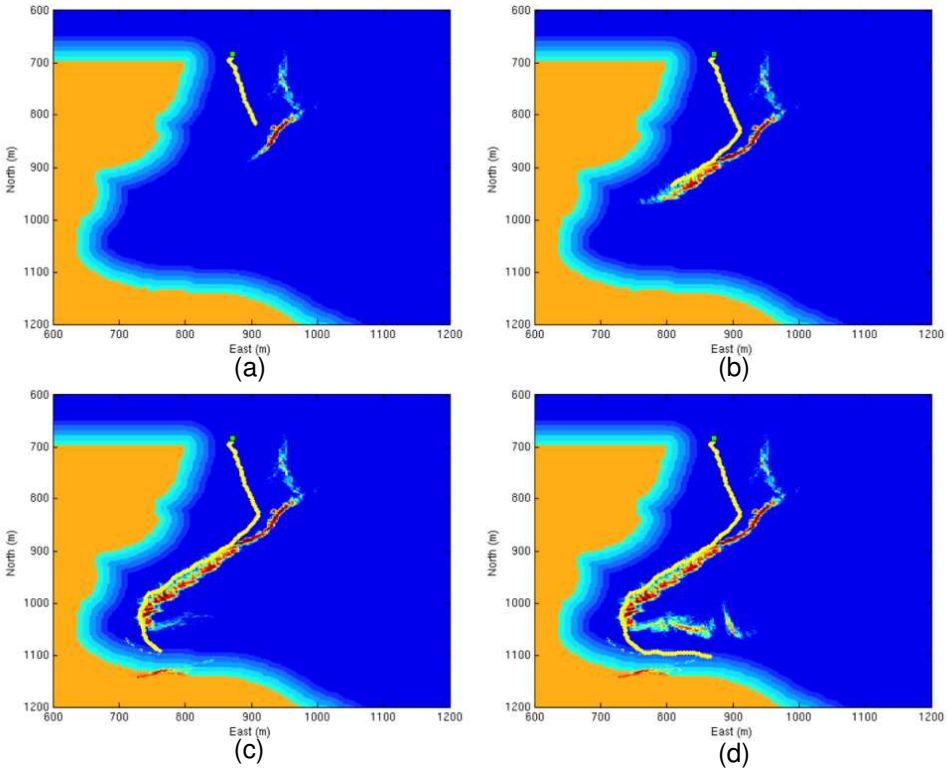

Figure 6.16: From (a) to (d): The history of the detected shorelines (red lines) and the trajectory of the ASV (yellow line).

Chapter 7

Conclusions

7.1 Summary

In this dissertation, we have proposed several fast path planning methods for autonomous navigation in uncertain environments. For the case in which there are only static obstacles in the environment, we propose a hybrid receding horizon control method which solves some challenging problems that exist in current receding horizon control methods for path planning. These challenges include stability and trapping issues as well as the requirements for simple obstacles geometries which are often not the case in a realistic environment. For the stability issue and trapping problem, the proposed method incorporates the solutions of certain Eikonal equations as a terminal cost. We have shown that the RHC formulation with this selection of the terminal cost is asymptotically stable using the results of [48] in the case that the domain is completely known. In addition, we have derived a sufficient condition for the convergence to the target. The convergence proof relies on the definition of the matching conditions that determine when and if a new global solution of the Eikonal equations should be incorporated as a terminal cost in the RHC formulation. The proposed method does not need to assume that obstacles are simple since it adopts fast marching method [64]. The advantage of the proposed method is that it captures the desirable characteristics of both

global and local path planning: fast local obstacle avoidance and long-term global planning that ensures the convergence to the goal.

We then address the path planning problem in the presence of moving obstacles. To simplify the planning problem, we first assume that both moving and static obstacles are completely known for the duration of the mission. We employ the Euler-Lagrange equations and derive a partial differential equation ([78]) whose domain is the original two dimensional environment. The suboptimal controller is simply along the gradient direction of the solution of the derived partial differential equation. The computational expense is saved by numerically solving the partial differential equation in the two dimensional environment only without considering the time space. Then, we discuss the case when moving and static obstacles are detected in the mission by a sensor with limited range. Incorporating the proposed PDE for moving obstacles, we propose another receding horizon controller. We rigorously derive a sufficient condition, by enforcing the matching condition for each end-point in every planning horizon, that guarantees that the vehicle will converge to the goal. In terms of the computational expense, we need only to compute the solution of the PDE over a small, local domain due to the limited length of the planning horizon. Therefore, the computational expense of the proposed receding horizon control is further reduced.

For the case in which one needs to recompute the new global level sets upon the detection of environmental changes, we have proposed a dynamic fast marching method. The main feature of the proposed method is that it updates level sets values over a portion of the domain. Thus, the computational expense is reduced compared to recomputing the level sets over the entire domain. To justify the proposed method, we give the formal analysis investigating how level sets change when new obstacles and new empty areas are detected.

In the end, we have shown a field trial for an autonomous surface vehicle navigating in a complex and large scale riverine environment. The vehicle managed to reach a target four kilometers away in about 45 minutes with complete autonomy.

7.2 Future Work

The scope of the future research includes:

- The current method idealizes the autonomous vehicle as a point particle. Thus, it works only on the class of autonomous vehicles that can follow an arbitrary trajectory with high accuracy, but it may fail in some occasions. This is because the proposed methods do not account for the dynamics of the autonomous vehicle which result in the lack of constraints on the feasible trajectory with respect to the vehicle's dynamics. Thus, we shall incorporate the vehicle dynamics in the path planning model;

- We shall extend our work to the path planning for multiple vehicles collaborations. A challenge for this effort is due to the significant computational cost caused by the increasing dimension of the search space that is based on the number of autonomous vehicles.

Bibliography

[1] K. Alton and I.M. Mitchell, "Optimal path planning under different norms continuous state spaces," *Proc. of IEEE International Conf. on Robotics and Automation*, pp. 866-872, 2006.

[2] J.-P. Aubin, *Viability Theory*, Birkhäuser, 1991.

[3] J.-P. Aubin, *Differential Inclusions: Set-Valued Maps and Viability Theory*, Springer-Verlag, 1984.

[4] M. Bardi and I. Capuzzo-Dolcetta, *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*, Birkhäuser, 1997.

[5] J. Bellingham, A. Richards and J.P. How, "Receding Horizon Control of Autonomous Aerial Vehicles," *Proc. of American Control Conference*, 2002.

[6] A.E. Bryson, Jr. and Y.C. Ho, *Applied Optimal Control*, Taylor & Francis, 1975.

[7] W. Choi, D. Zhu and J.C. Latombe, "Contingency-tolerant robot motion planning and control," *Proc. of IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pp.78-86, 1989.

[8] L.D. Cohen and R. Kimmel, "Global minimum for active contours models: a minimal path approach," *International Journal of Computer Vision*, vol.24, no.1, pp.57-78, 1997.

[9] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to algorithms*, The MIT Press, Cambridge, MA, 1989.

[10] K. Culligan, M. Valenti, Y. Kuwata and J.P. How, "Three-Diementional Flight Experiments Using On-Line Mixed Integer Linear Programming Trajectory Optimization, " *Proc. of American Control Conference*, pp. 5322-5327, 2007.

[11] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol.22, no.6, pp.46-57, 1989.

[12] G.M. Ewing, *Calculus of Variations with Applications*, W.W. Norton & Co., New York, 1969.

[13] D. Ferguson, M. Likhachev and A. Stentz, "A guide to heuristic path planning," *Proc. of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling*, 2005.

[14] D. Ferguson and A. Stentz, "The delayed D* algorithm for efficient path replanning," *Proc. of the IEEE International Conf. on Robotics and Automation*, pp.2045-2050, 2005.

[15] P. Fiorini and Z. Shiller, "Time optimal trajectory planning in dynamic environments," *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 1553–1558, 1994.

[16] T. Fraichard, "Dynamic trajectory planning with dynamic constraints: a 'state-time space' approach," *Proc. of IEEE/RSJ conference on Intelligent Robts and Systems*, pp.1393–1399, 1993.

[17] K. Fujimura and H. Samet, "A hierarchical strategy for path planning among moving obstacles," *Trans on Robotics and Automation*, vol. 5, no. 1, pp. 61–69, 1989.

[18] K. Fujimura, "Time-minimum routes in time-dependent networks," *Trans on Robotics and Automation*, vol. 11, no. 13, pp.343–351, 1995.

[19] A.S. Gadre, S. Kragelund, T. Masek, D.J. Stilwell, C. Woolsey, and D. Horner, "Subsurface and Surface Sensing for Autonomous Navigation in a Riverine Environment," *Proc. of AUVSI's Unmanned System North America*, 2009.

[20] I.M. Gelfand and S.V. Fomin, translated and edited by R.A. Silverman, *Calculus of Variations*, Dover Publications, Mineola, NY, 2000.

[21] X. Gong, "Omnidirectional vision for an autonomous surface vehicle," *Dissertation (Ph.D.)*, Virginia Polytechnic Institute and State University, 2008.

[22] X. Gong, B. Xu, C. Reed, C. Wyatt and D.J. Stilwell,"Real-time robust mapping for an autonomoous surface vehicle using an omnidirectional camera," *Workshop on Applications of Computer Vision*, 2008.

[23] G.C. Goodwin, M.M. Seron and J.A. De Doná, *Contrstrained Control and Estimation: An Optimisation Approach*, Springer-Verlag, 2005.

[24] Y. Guo and T. Tang, "Optimal trajectory generation for nonholonomic robots in dynamic environments," *Proc. of International Conference on Robotics and Automation*, pp. 2552–2557, 2008.

[25] M.S. Hassouna, A.E. Abdel-Hakim and A.A. Farag, "Robust robotic path planning using level sets," *Proc. of IEEE International Conf. on Image Processing*, vol.3, pp.473-476, 2005.

[26] Y.K. Hwang and N. Ahuja, "Gross motion planning- a survey," *ACM Computing Surveys*, vol. 24, no.3, pp. 219–291, 1992.

[27] K.S. Hwang and M.Y. Ju, "Speed planning for maneuvering motion," *Journal of Intelligent and Robotic Systems*, no. 33, pp. 25–44, 2002.

[28] H. Ishii, "A simple, direct proof of uniqueness of solutions for the Hamilton-Jacobi equations of Eikonal type," *Proc. of the Americane Mathematical Society*, vol. 100, no.2, pp. 247–251, 1987.

[29] A. Jadbabaie, J. Yu, and J. Hauser, "Stabilizing receding horizon control of nonlinear systems: a control Lyapunov function approach," *Proc. of the American Control Conference*, pp. 1535–1539, 1999.

[30] A. Jadbabaie, J. Yu, and J. Hauser, "Unconstrained receding horizon control of nonlinear systems," *Proc. of the 38th Conference on Decision and Control*, pp. 3376–3381, 1999.

[31] A. Kelly and A. Stentz, "Rough terrain autonomous mobility, part 1: A theoretical analysis of requirements," *Autonomous Robots*, 5, pp. 129–161, 1998

[32] H.K. Khalil, *Nonlinear Systems, Third Edition*, Prentice Hall, 2002.

[33] M. Khatib, H. Jaouni, R. Chatila, J.P. Laumond, "Dynamic path modification for car-like nonholonomic mobile robots," *Prof. of IEEE International Conf. on Robotics and Automation*, pp.2920-2925, 1997.

[34] R. Kimmel, A. Amir and A.M. Bruckstein, "Finding shortest paths on surfaces using level set methods," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.17, no.6, pp.635-640, 1995.

[35] R. Kimmel, N. Kiryati, and A.M. Bruckstein, "Multivalued distance maps for motion planning on surfaces with moving obstacles," *IEEE Trans on Robotics and Automation*, vol. 14, no. 3, 1998.

[36] R. Kimmel and J.A. Sethian, "Optimal algorithm for shape from shading and path planning," *Journal of Mathematical Imaging and Vision*, vol. 14, pp.237-244, 2001.

[37] R. Kimmel, M. Bronstein, and A. Bronstein, *Numerical Geometry of Images: Theory, Algorithms, and Applications*, Springer-Verlag, NY, 2003.

[38] S. Koenig and M. Likhachev. "Improved fast replanning for robot navigation in unknown terrain," *Technical Report GIT-COGSCI-2002/3*, College of Computing, Georgia Institute of Technology, Atlanta, GA, 2002.

[39] B.H. Krogh and C.E. Thrope, "Integrated path planning and dynamic steering control for autonomous vehicles," *Proc. of IEEE International Conf. on Robotics and Automation*, pp.1664-1669, 1986.

[40] Y. Kuwata and J. How, "Three dimensional receding horizon control for UAVs" *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004.

[41] Y. Kuwata, T. Schouwenaars, A. Richards, and J. How, "Robust constrained receding horizon control for trajectory planning" *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005.

[42] K.J. Kyriakopoulos and G.N. Saridis, "An integrated collision prediction and avoidance scheme for mobile robots in non-stationary environments," *Automatica*, vol. 29, no. 2, pp.309–322, 1993.

[43] F. Lamiraux, D. Bonnafous and O. Lefebvre, "Reactive path deformation for nonholonomic mobile robots," *IEEE Trans. on Robotics*, vol.2, no.6, pp.967-977, 2004.

[44] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[45] J. Larson, M. Bruch, and J. Ebken, "Autonomous navigation and obstacle avoidance for unmanned surface vehicles ," *Proc. of SPIE*, vol. 6230, 2006

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

[47] P.L. Lions, *Generalized Solutions of Hamilton-Jacobi Solutions*, Pitman Publishing INC, 1982.

[48] D.Q. Mayne, J.B. Rawlings, C.V. Rao and P.O.M. Scokaert, "Survey Paper: Constrained model predictive control: Stability and optimality", *Automatica*, no.36, pp. 789-814, 2000.

[49] B. Merhy, P. Payeur, and E. Petriu, "Application of segmented 2D probabilistic occupancy maps for mobile robot sensing and navigation," *Proc. of Instrumentation and Measurement Technology Conf.*, pp. 2342–2347, 2006.

[50] I.M. Mitchell and S. Sastry, "Continuous path planning with multiple constraints," *Proc. of the 42nd IEEE Conf. on Decision and Control*, pp.5502-5507, 2003.

[51] S.J. Osher and J.A. Sethian, "Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations," *Journal of Computational Physics*, 79, pp.12-49, 1988.

[52] D. Pagac, E. Nebot, and H. Durrant-Whyte, "An evidential approach to map-building for autonomous vehicles," *IEEE Trans. on Robotics and Automation*, vol. 14, pp. 623–629, 1998.

[53] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans and D. Lane, "Path planning for autonomous underwater vehicles," *IEEE Trans. on Robotics*, vol.23, no.2, pp.331-341, 2007.

[54] R. Philippsen and R. Siegwart, "An Interpolated Dynamic Navigation Function," *Proc. of the IEEE International Conf. on Robotics and Automation*, pp.3782-3789, 2005.

[55] R. Phillippsen, "A light formulation of the E* interpolated path replanner," *Technical Report*, Autonomous Systems Lab, Ecole Polytechnique Federale de Lausanne, Switzerland, 2006.

[56] R. Prazenica, A. Kurdila, R. Sharpley and J. Evers, "Vision-Based Geometry Estimation and Receding Horizon Path Planing for UAVs Operating in Urban Envrionments, " *Proc. of American Control Conference*, 2006.

[57] J.A. Primbs, "Nonlinear Optimal Control: A Receding Horizon Approach," *Ph.D Dissertation*, California Institute of Technology, 1999.

[58] Z. Qu, J. Wang, and C.E. Plaisted, "A new analytical solution to mobile robot trajectory generation in the presence of moving obstacles," *IEEE Trans. on Robotics*, vol. 20, no. 6, 2004.

[59] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," *Proc. of IEEE International Conf. on Robotics and Automation*, pp.802-807, 1993.

[60] C. Reed, "Experiments in real-time path planning for riverine environments," *Thesis (M.S.)*, Virginia Polytechnic Institure and State University, 2008.

[61] J.N. Riggins, "Location Estimation of Obstacles for an Autonomous Surface Vehicle," *Thesis (M.S.)*, Virginia Polytechnic Institure and State University, 2006.

[62] E. Rouy and A. Tourin, "A viscosity solutions approach to shape-from-shading," *SIAM Journal on Numerical Analysis*, vol.29, no.3, pp.867-884, 1992.

[63] T. Schouwenaars, J. How, and E. Feron, "Receding Horizon Path Planning with Implicit Safety Guarantees," *Proc. of American Control Conference*, 2002.

[64] J.A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, 1999.

[65] J.A. Sethian, "Fast marching method," *SIAM Review*, vol.41, No.2, pp.199-235, 1999.

[66] J.A. Sethian and A. Vladimirsky "Ordered upwind methods for static Hamilton-Jacobi equations," *Proc. Nat. Acad. Sci. USA*, vol.98, pp. 11069-11074, 2001.

[67] G.V. Smirnov, *Introduction to the Theory of Differntial Inclusions*, American Methematical Society, 2001.

[68] A. Stentz, "Optimal and efficient path planning for partially-known environments," *Proc. of IEEE International Conf. on Robotics and Automation*, vol.4, pp.3310–3317, 1994.

[69] A. Stentz, "A complete navigation system for goal acquisition in unknown environments," *Autonomous Robots*, vol.2, no. 2, pp. 127–145, 1995.

[70] A. Subramanian, X. Gong, J. Riggins, D. J. Stilwell and C. Wyatt, "Shoreline mapping using an omnidirectional camera for autonomous surface vehicle applications," *Proc. of Oceans MTS/IEEE*, Boston, MA, pp. 1–6, 2006.

[71] P. G. Trepagnier et al., "KAT-5: robust systems for autonomous vehicle navigation in challenging and unknown terrain," *Journal of Field Robotics*, pp. 509–526, 2006.

[72] S. Thrun et al., "Stanley: the robot that won the DARPA grand challenge," *Journal of Field Robotics*, pp. 661–692, 2006.

[73] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, The MIT Press, 2005.

[74] J.N. Tsitsiklis, "Efficient algorithm for globally optimal trajectories," *IEEE Trans. on Automatic Control*, vo. 40, no. 9, pp. 1528–1538, 1995.

[75] A. Tsoularis and C. Kambhampati, "Avoiding moving obstacles by deviation from a mobile robot's nominal path", *International Journal of Robtics Research*, vol. 18, no.5, pp. 454–465, 1999.

[76] C. Urmson et al., "A robust approach to high-speed navigation for unrehearsed desert terrain," *Journal of Field Robotics*, pp. 467–508, 2006.

[77] R. Vinter, *Optimal Control*, Birkhauser, 2000.

[78] A. Vladimirsky, "Static PDEs for time-dependent control problems," *Interfaces and Free Boundaries*, no. 8, pp.281–300, 2006.

[79] A.S. Watkins, R. Prazenica, A. Kurdila and G. Wiens, "RHC for Vision-Based Navigation of a WMR in an Urban Environment," *Proc. of AIAA Guidance, Navigation and Control Conference and Exhibit*, 2005.

[80] B. Xu, D.J. Stilwell, A. Gadre, and A.J. Kurdila, "Analysis of local observability for feature localization in a maritime environment using an omnidirectional camera," *Proc.*

of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3666–3671, 2007.

[81] B. Xu, D.J. Stilwell, and A.J. Kurdila, "Efficient Computation of Level Sets for Path Planning", *IEEE/RSJ International Conf. on Intelligent Robots and Systems*, 2009.

[82] B. Xu, A.J. Kurdila, and D.J. Stilwell, "A Hybrid Receding Horizon Control Method for Path Planning in Uncertain Environments", *IEEE/RSJ International Conf. on Intelligent Robots and Systems*, 2009.

[83] B. Xu, D.J. Stilwell, and A.J. Kurdila, "A Receding Horizon Controller for Motion Planning in the Presence of Moving Obstacles", submitted to *IEEE International Conference on Robotics and Automation*, 2010.

[84] M. Zabarankin, A. Kurdila, O. Prokopyev, R. Causey, A. Goel and P. Pardalos, "Optimization Approaches for Vision-Based Path Planning for Autonomous Micro-Air Vechiels in Urban Environments," *Cooperative Networks: Control and Optimization*, Edward Elgar Publishing Ltd, 2008.