Path Planning in Time Dependent Flows using Level Set Methods

by

Sri Venkata Tapovan Lolla

B.Tech., Indian Institute of Technology Bombay (2010)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2012

Author .....
Department of Mechanical Engineering
August 17, 2012

Certified by .....
Pierre F. J. Lermusiaux
Associate Professor
Thesis Supervisor

Accepted by .....
David E. Hardt
Chairman, Department Committee on Graduate Theses

Path Planning in Time Dependent Flows using Level Set Methods

by

Sri Venkata Tapovan Lolla

Submitted to the Department of Mechanical Engineering
on August 17, 2012, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

Autonomous underwater vehicles such as gliders have emerged as valuable scientific platforms due to their increasing uses in several oceanic applications, ranging from security, acoustic surveillance and military reconnaissance to collection of ocean data at specific locations for ocean prediction, monitoring and dynamics investigation. Gliders exhibit high levels of autonomy and are ideal for long range missions. As these gliders become more reliable and affordable, multi-vehicle coordination and sampling missions are expected to become very common in the near future. This endurance of gliders however, comes at an expense of being susceptible to typical coastal ocean currents. Due to the physical limitations of underwater vehicles and the highly dynamic nature of the coastal ocean, path planning to generate safe and fast vehicle trajectories becomes crucial for their successful operation. As a result, our motivation in this thesis is to develop a computationally efficient and rigorous methodology that can predict the time-optimal paths of underwater vehicles navigating in continuous, strong and dynamic flow-fields. The goal is to predict a sequence of steering directions so that vehicles can best utilize or avoid flow currents to minimize their travel time.

In this thesis, we first review existing path planning methods and discuss their advantages and drawbacks. Then, we discuss the theory of level set methods and their utility in solving front tracking problems. Then, we present a rigorous (partial differential equation based) methodology based on the level set method, which can compute time-optimal paths of swarms of underwater vehicles, obviating the need for any heuristic control based approaches. We state and prove a theorem, along with several corollaries, that forms the foundation of our approach for path planning. We show that our algorithm is computationally efficient - the computational cost grows linearly with the number of vehicles and geometrically with spatial directions. We illustrate the working and capabilities of our path planning algorithm by means of a number of applications. First, we validate our approach through simple benchmark applications, and later apply our methodology to more complex, realistic and numerically simulated flow-fields, which include eddies, jets, obstacles and forbidden regions. Finally, we extend our methodology to solve problems of coordinated

motion of multiple vehicles in strong dynamic flow-fields. Here, coordination refers to maintenance of specific geometric patterns by the vehicles. The level-set based control scheme that we derive is shown to provide substantial advantages to a local control approach. Specifically, the illustrations show that the resulting coordinated vehicle motions can maintain specific patterns in dynamic flow fields with strong and complex spatial gradients.

Thesis Supervisor: Pierre F. J. Lermusiaux
Title: Associate Professor

Acknowledgments

I would like to extend my gratitude to my advisor, Prof. Pierre Lermusiaux, for his guidance and support throughout the course of this work. I thank him particularly for allowing me to choose a topic of interest, and also for carefully proofreading this thesis. His enthusiasm, words of encouragement and constructive criticism have helped me develop a broad perspective on research. I also thank Pat, Wayne, Marcia and Leslie for all their help over the last two years.

I thank all the members of MSEAS, particularly Matt, Themis, Thomas, Konur, Pete, Akash and Chris for many entertaining and fun-filled discussions in the office.

Finally, I thank my family and close friends for believing in me and always being there when I needed them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

1.1 Path Planning

The task of planning a path in a complex environment is a problem as old as antiquity. This problem has existed for several decades now, perhaps since the age of mobile robots. Therefore, this topic has a long history and it has received a great deal of attention from many branches of science and engineering (especially the robotics community). In the most general sense, 'path planning' refers to a set of rules provided to an autonomous robot which enables the unit to navigate from one configuration to another, in an 'optimal' fashion. Optimality of the navigation is generally governed by an objective criterion of performance. The term 'path planning' is used by many communities of engineering and science in a broad range of problems. It is applicable to any situation where a semi or fully autonomous system has to be maneuvered.

In most cases, path planning is performed for autonomous vehicles. The optimal plans are then provided to these vehicles and they can further adapt these plans as they execute their missions. Since autonomous robots navigate with little or no human intervention, path planning becomes a crucial task for their successful operation and safety. In contemporary science and engineering, modeling and computational approaches are utilized to plan paths for these autonomous vehicle systems.

Autonomous robotic platforms are designed to accomplish a wide range of tasks in

a variety of fields. There does not exist a universal solution to path planning, primarily because it is applicable in a number of diverse robotic scenarios. These scenarios may require the robots to optimize dissimilar objective criteria and consequently, the levels of complexity associated with these tasks may be quite different. Due to this, there does not exist a universal path planner which is applicable to all autonomous platforms. For example,

- Autonomous Underwater Vehicles (AUVs) (such as gliders and other propelled vehicles) are a class of autonomous robots that are often used for ocean mapping, commercial exploration, military reconnaissance and harbor protection (Stommel, 1989). They are also used to make measurements of field quantities of interest in the ocean which aid in ocean prediction and other types of scientific research. Underwater gliders are also used by oil and gas industries to make a detailed map of the environment before embarking up on any drilling activity. Path planning for AUVs may thus, involve minimization of time of travel or energy spent or the mapping information gained by the AUV.

- Robots for military surveillance and reconnaissance may be navigated to maximize visibility or gather intelligence by inspecting a field. A path planner for these robots may be used to generate safe paths, away from physical obstacles in the environment. The quality of the path may be governed, in this case, by the amount of intelligence gathered by the robot.

- Other domains involving robotic applications include manufacturing, medicine, molecular biology etc. Autonomous robots in each of these fields may be navigated to achieve widely different goals.

It is therefore, very challenging to derive a single theory that holds for every possible application. However, one aspect that all mobile robots have in common is that they must find a trajectory to their destination taking into account, the possibly dynamic nature of the environment and limited capabilities of the robot itself.

Optimal navigation of autonomous vehicles (such as underwater gliders) in the coastal ocean has become crucial for many applications, ranging from security and

acoustic surveillance, to collection of ocean data at specific locations, for ocean prediction and monitoring. Underwater gliders are ideal for long range missions due to their low power consumption and high levels of autonomy. This endurance, however, comes at the expense of smaller travel speeds of the glider. In many cases, the speed of the glider becomes comparable to, or even lesser than the speeds of the ocean currents in which the glider operates. As these gliders have become more reliable and affordable, the simultaneous use of several vehicles has recently emerged as a viable option. Multi-vehicle sampling and exploratory missions are expected to become very common in the near future. This will not only make missions with coordination possible, but also enable individual robotic units to benefit from the information obtained by other members of the group (Davis et al. (2009), Bahr et al. (2009)). This naturally raises the question of navigating gliders through dynamic ocean currents.

Path planning for autonomous underwater vehicles (AUVs) in general aims to optimize at least one of the following aspects of performance:

1. Travel time between two given points
2. Energy spent by the vehicle
3. Safety of the vehicle
4. Quality of data gathered by the vehicle

The environment of AUVs is the ocean, a highly dynamic and multi-scale system with considerable variability in both time and three-dimensional space. Planning of optimal AUV paths in the ocean is quite challenging. The speeds of ocean currents can be comparable to vehicle speeds (e.g. for AUVs (see Schmidt et al. (1996), Elisseeff et al. (1999))) or even much larger than vehicle speeds (e.g. for gliders) in some cases. In these cases, the dynamic nature of ocean currents and their effect on the vehicle path should not be neglected. Ocean geometry is often complex, especially in the coastal zone. The challenge therefore, is to develop computationally efficient and rigorous frameworks that accommodate both the environmental constraints and robotic limitations while at the same time providing an accurate path for the robot

(Belta et al., 2007). Consequently, our motivation is to develop and illustrate efficient but rigorous methodologies that predict the optimal paths of swarms of ocean vehicles in dynamic ocean currents, without any limitation on the currents nor on the number of vehicles.

This thesis focusses on time optimal path planning for autonomous underwater vehicles such as gliders, which navigate in strong and dynamic ocean flow-fields. We aim to predict how these vehicles must best utilize or avoid currents in order to minimize their travel time. We present a rigorous (partial differential equation (PDE) based) methodology based on the level set method, which can compute optimal paths of swarms of underwater vehicles, obviating the need for any heuristic control based approaches. As will be shown later, this algorithm is computationally efficient - the computational cost grows linearly with the number of vehicles and geometrically with spatial dimensions. We illustrate, by means of a number of examples, that the algorithm automatically generates vehicle trajectories that avoid any obstacles in the domain. These obstacles can either be stationary or time dependent. In what follows, we will show that if vehicles travel at maximum nominal speed in a time dependent flow-field, our methodology will efficiently compute the exact fastest path between any two locations along with the sequence of headings that the vehicle must take to realize this fastest path. We describe a number of benchmark examples to validate our methodology and present results of the algorithm in more realistic ocean scenarios that include obstacles, eddies and forbidden regions. Finally, we show how this algorithm can be used to plan paths for multiple vehicles, either independently or in a coordinated fashion, where coordination refers to formation and maintenance of specific geometric patterns.

1.2 Layout of Thesis

This thesis is organized as follows: In chapter 2, we formally define our problem statement and introduce all the relevant notation. We also state all the assumptions that we use in this work. In chapter 3, we perform a thorough literature review of

the relevant existing methodologies for underwater path planning. We discuss their advantages and various drawbacks describe how our methodology can overcome them. Chapter 4 introduces the reader to level set methods. In this chapter, we describe the theory of front propagation and the utility of level set methods in interface tracking. We also discuss level set methods from a mathematical point of view by describing the link between viscosity solutions of Hamilton-Jacobi equations and other hyperbolic conservation laws. In chapter 5, we describe our path planning algorithm. Here, we state and prove a theorem along with several corollaries which forms the basis of our methodology. In chapter 6, we discuss the numerical details of implementation of the algorithm and provide estimates of the computational cost of the method. In chapter 7, we illustrate a number of applications of the path planning algorithm. We first validate the algorithm by using benchmark examples for which analytical solutions can be easily computed. Later, we apply the algorithm to more complex and realistic ocean flow scenarios. In chapter 8, we describe how our methodology can be used for coordinated path planning of multiple underwater vehicles. Here, we present two novel approaches for pattern formation of swarms of vehicles and discuss their features. In chapter 9, we summarize and highlight the main contributions of this thesis and describe some possible directions for future work.

Chapter 2

Problem Statement

Consider the motion of a vehicle in a physical domain Ω, in a dynamic flow-field, denoted by $\mathbf{V}(\mathbf{x}, t)$. Let $\mathbf{x_s}$ and $\mathbf{x_f}$ respectively denote the position vectors of start and end points of the vehicle. Let the maximum speed of the vehicle relative to the flow-field (i.e. its maximum speed in still flow) be a constant, F. We wish to develop a steering rule for the vehicle that minimizes its travel time. In other words, we wish to develop an algorithm that predicts the sequence of headings (steering directions) for the vehicle that would result in the fastest time path between $\mathbf{x_s}$ and $\mathbf{x_f}$. Let the continuous trajectory traced by the vehicle be denoted as $\mathbf{p}(t)$. We refer the reader to figure (2-1) for a visual depiction of the vehicle motion and various parameters involved.

The vehicle motion is governed by the kinematic relation (2.1). In addition to its nominal motion due to steering ($F_v(t)\hat{\mathbf{h}}(t)$), the vehicle also gets advected by the flow-field ($\mathbf{V}(\mathbf{x}, t)$). Therefore, the total velocity of the vehicle ($\mathbf{U}(\mathbf{x}, t)$) is composed of both, nominal velocity and the advection due to the flow-field.

$$\frac{d\mathbf{p}}{dt} = \mathbf{U}(\mathbf{p}(t), t) = |\mathbf{U}(\mathbf{p}(t), t)|\hat{\mathbf{u}}(t) = F_v(t)\hat{\mathbf{h}}(t) + \mathbf{V}(\mathbf{p}(t), t) \tag{2.1}$$

where $F_v(t)$ is the speed of the vehicle relative to the flow-field (i.e. speed of the

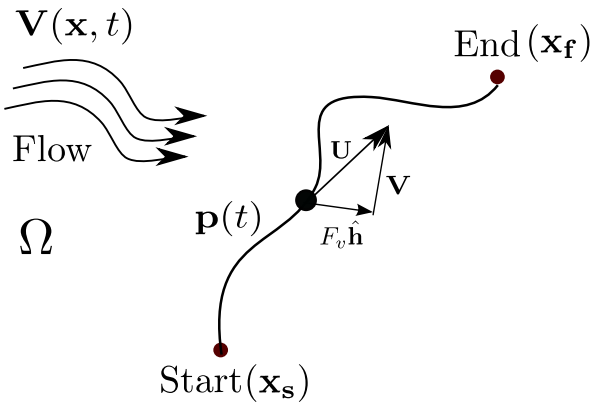


Figure 2-1: Vehicle motion in a dynamic flow-field given by $\mathbf{V}(\mathbf{x},t)$. The start point ($\mathbf{x_s}$) and the end point ($\mathbf{x_f}$) are shown along with the vehicle path $\mathbf{p}(t)$. The overall velocity of the vehicle ($\mathbf{U}(\mathbf{x},t)$) is tangential to $\mathbf{p}(t)$ and is composed of the nominal steering velocity ($F_v(t)\hat{\mathbf{h}}(t)$) and the flow-field velocity ($\mathbf{V}(\mathbf{x},t)$). The steering direction ($\hat{\mathbf{h}}$) is different from the direction in which the vehicle moves ($\hat{\mathbf{u}}$).

vehicle in still flow), and is bounded above by F, i.e.,

$$0 \leq F_v(t) \leq F$$

$\hat{\mathbf{h}}(t)$ is the vehicle heading direction at time t. This is the direction that the vehicle moves *relative to the flow field*. In other words, $\hat{\mathbf{h}}(t)$ is the direction in which the vehicle is *steered*. Let $T(\mathbf{x})$ denote the 'first arrival time' field, i.e. the first time the vehicle can reach any given $\mathbf{x}$, starting from $\mathbf{x_s}$. Clearly, $T(\mathbf{x_s}) = 0$, since the vehicle starts moving from $\mathbf{x_s}$. For limiting conditions on $\mathbf{p}(t)$, we have,

$$\mathbf{p}(0) = \mathbf{x_s}, \qquad \mathbf{p}(T(\mathbf{x_f})) = \mathbf{x_f} \tag{2.2}$$

The goal is thus, to develop a control for the vehicle heading $\hat{\mathbf{h}}(t)$ and for the vehicle speed $F_v(t)$ that minimizes the travel time, $T(\mathbf{x_f})$, subject to constraints imposed by equations equation (2.1) and equation (2.2).

We note that this problem is a modified version of Zermelo's navigation problem (Zermelo, 1931). Zermelo's navigation problem is a standard problem in the theory

of optimal control (see e.g. Bryson and Ho (1975)). The original version of Zermelo's navigation problem (Zermelo, 1931) discusses time optimal path planning in *steady* flow-fields. The reader is referred to (Falcone and Zidani (2012), Mitchell et al. (2005), Bryson and Ho (1975)) for a detailed discussion on Zermelo's navigation problem.

2.1 Assumptions

1. In this thesis, we assume that the flow-field $\mathbf{V}(\mathbf{x}, t)$ is exactly known. Since the work is primarily directed towards oceanic applications, in which flow-fields are always associated with some levels of uncertainty, this assumption is somewhat simplifying. For the purposes of this work, however, we assume that $\mathbf{V}(\mathbf{x}, t)$ is completely known.

2. We aim to obtain globally optimal long-distance trajectories for the autonomous vehicle. In other words, the distance traveled by the vehicle is much larger than its geometric dimensions.

3. We assume that the interaction between the vehicle and the flow-field is purely kinematic. In other words, we assume that the vehicle behaves as a point mass and exhibits no inertia.

4. We plan a continuous trajectory for the vehicle, i.e. $\mathbf{p}(t)$ is always continuous. This imposes a constraint on the type of the flow-field ($\mathbf{V}(\mathbf{x}, t)$) the vehicle can experience. $\mathbf{V}(\mathbf{x}, t)$ cannot be a Dirac delta distribution. If $\mathbf{V}(\mathbf{x}, t)$ is a Dirac delta distribution, the vehicle trajectory (obtained by integrating equation (2.1)) will no longer be continuous. Any other type of flow-field $\mathbf{V}(\mathbf{x}, t)$ is acceptable.

Chapter 3

Literature Review

The goal of this chapter is to provide the reader with a thorough review of existing literature on underwater path planning. The goals of general robotic path planning and underwater path planning are different at a very basic level. Path planning for robotic applications has been extensively studied and ample literature is available on this topic. Underwater path planning has received far lesser attention in comparison. Among the researchers who have worked on this problem, most of them have tried to extend robotic path planning algorithms to address underwater path planning. Therefore, in this chapter, we first review some useful robotic path planning algorithms, namely the A* algorithm and Rapidly exploring Random Trees (RRTs). We then explain the difference between robotic path planning and underwater path planning. Later, we provide an extensive literature review on existing underwater path planning algorithms.

3.1 Robotic Path Planning

3.1.1 Theory - A* Algorithm

In this section, we briefly describe the A* search algorithm. The A* algorithm is widely used by the mobile robotics community for computing obstacle free paths for robots. The algorithm is well suited to solve problems where the environment of the

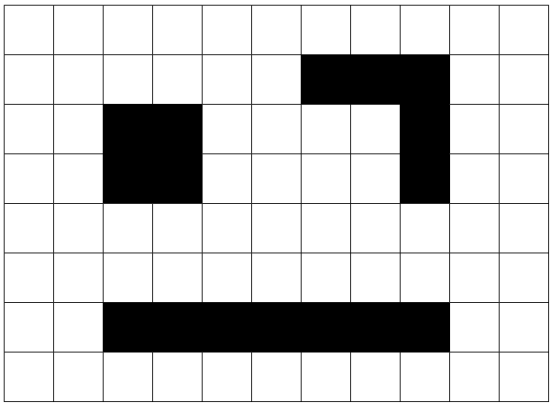
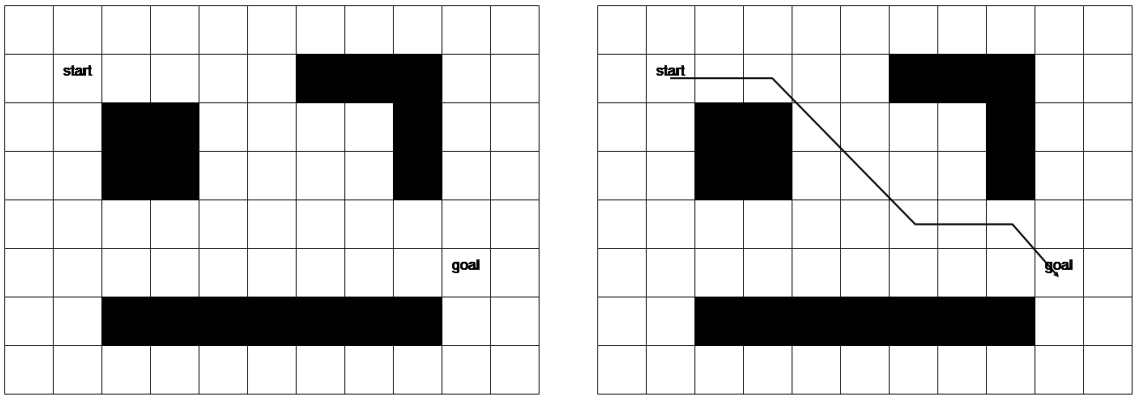


Figure 3-1: Uniform discretization of a continuous robot workspace into cells. The empty cells represent areas that the robot is free to visit while the solid cells represent the obstacles to the robot which it must avoid.

robot is filled with stationary physical obstacles which the robot must avoid. We define the *workspace* of the robot as the set of all possible points that the robot can visit.

A common approach for robotic path planning is to first divide the workspace of the robot into *cells*. This discretization of the workspace converts a continuous path planning problem to a discrete one. Once the workspace has been discretized, the layout of the workspace can be represented by an *occupancy grid*. This grid can be considered as a map of the workspace where the free space is marked by clear cells and obstacles are represented by solid cells (see figure (3-1)). The A* algorithm computes the shortest distance between two points by determining the shortest sequence of free cells that connects these points.

The A* scheme converts the path planning problem into a *graph search* problem (Wikipedia). Each empty cell in the occupancy grid is represented as a node in a graph. An edge exists between two nodes only if the cells they represent are adjacent to each other and a feasible path exists between these cells. Once this graph is generated, the A* scheme solves this graph search to compute the shortest path between any two points. The graph search problem can be solved by a number of different approaches such as depth-first search, breadth-first search, Dijkstra's method etc. A* is similar to these methods in that, it is a graph search technique. What sets



(a) (b)

Figure 3-2: (a) **start** and **end** points for the robot in a workspace filled with obstacles, and (b), the optimal path computed by the A* algorithm.

A* apart from these methods is the use of a *heuristic function* during the search process.

The heuristic function, denoted by $h(x)$, provides an estimate of the cost of the best route that passes through a particular node. The algorithm keeps track of the cost of the route leading up to a particular node along with the heuristic cost function $h(x)$ to determine which node it must visit next. It is a special case of a *best-first* algorithm. In fact, if the heuristic cost function $h(x)$ is chosen to be zero, we can show that the A* algorithm is equivalent to Dijkstra's algorithm. Any heuristic function $h(x)$ is acceptable, as long as it underestimates the actual cost of the path from the cell x to the end point. For example, the Euclidean distance between cell point x and the end point is an acceptable choice of $h(x)$ because it is the shortest distance between these points. The choice of the heuristic function determines the number of iterations needed for the algorithm to find the optimal solution. The A* algorithm is both *complete* and *optimal*. This means that if the heuristic function $h(x)$ is appropriately chosen, the algorithm will yield an optimal solution if one exists, or terminate by reporting a failure if an optimal path does not exist.

We now look into the mechanics of the algorithm. We refer the reader to figure (3-2) for the locations of the start and end points for the robot in a typical example. The algorithm proceeds in the following steps:

Starting the Search

- Given the occupancy grid, the search begins at the **start** point. At every step, the algorithm maintains an *open list* and a *closed list* of cells. The open list contains the cells that can possibly fall on the optimal path which we wish to compute. In other words, these cells need to be further examined in the search process. The open list of cells will grow as we expand outward from the **start**.
- All the reachable cells, i.e. cells that do not contain obstacles, adjacent to **start** are added to the *open list*. For each of these newly added cells, the position of their parent cell, i.e, **start** is saved. This information will be used to compute the optimal path backwards from the end point.
- **start** is removed from the *open list* and added to the *closed list* as it does not have to be pursued any more.

The above steps are repeated starting from one of the cells in the *open list*. To determine which cell must be pursued next, the path cost associated with each cell in the *open list* is computed.

Path Costs

The path cost at any point x, denoted by $f(x)$ in the A* algorithm, is the sum of costs of a path leading to point x from **start**, $(g(x))$ and the heuristic estimate $h(x)$ of the cost to reach **end** from x. Thus,

$$f(x) = g(x) + h(x) \tag{3.1}$$

As mentioned earlier, $h(x)$ is an underestimate of the cost to go from x to the end point. Typically, it is chosen to be the Euclidean distance between x and **end**, disregarding all the obstacles in the domain. Thus, $f(x)$ is a conservative estimate of the total cost of the path from **start** to **end** through the current cell x. The path is generated by repeatedly going through the open list and choosing the cell with lowest $f(x)$.

Completing the Search

To complete the search, at every iteration, we simply choose the cell from our open list with the lowest cost $f(x)$ and do the following steps:

- Remove it from the *open list* and move it to the *closed list*.
- Look at the neighbors of the current cell and add them to the *open list* if they are not already on the list. Specify the current cell as the parent of the cells added to the *open list*.
- If an adjacent cell is already on the *open list*, check to see if the current path to the cell is a better one than the one earlier. In other words we check to see if $g(x)$ of this path is lower than the cost of the earlier path to this cell. If it is, the parent of the adjacent cell is changed to the current cell and both costs $f(x)$ and $g(x)$ are recalculated. If the cost to the current cell is higher than the earlier cost, no action is taken.

The algorithm terminates either when **end** is added to the *closed list* or when **end** is not added to the *closed list*, but the *open list* is empty. In the latter case, there is no feasible path from **start** to **end**. The optimal path, when one exists is computed by tracing back from the **end** to **start**. This is done by looking at the parent cell of each point on the optimal path and retracing the steps back to the start point.

A* algorithm is widely used by the robotics community for obstacle avoidance and generation of safe robot paths. However, it is not directly applicable for underwater path planning, as will be discussed later in this chapter. In the following section, we briefly describe Rapidly exploring Random Trees.

3.1.2 Theory - Rapidly exploring Random Trees (RRTs)

Rapidly exploring Random Trees (RRTs) were introduced by Lavalle (1998) as a tool for robotic path planning. RRTs are a randomized data structure that are used in a wide range of path planning problems. They are specifically designed to easily deal with non-holonomic robotic path planning problems of large dimensionality and

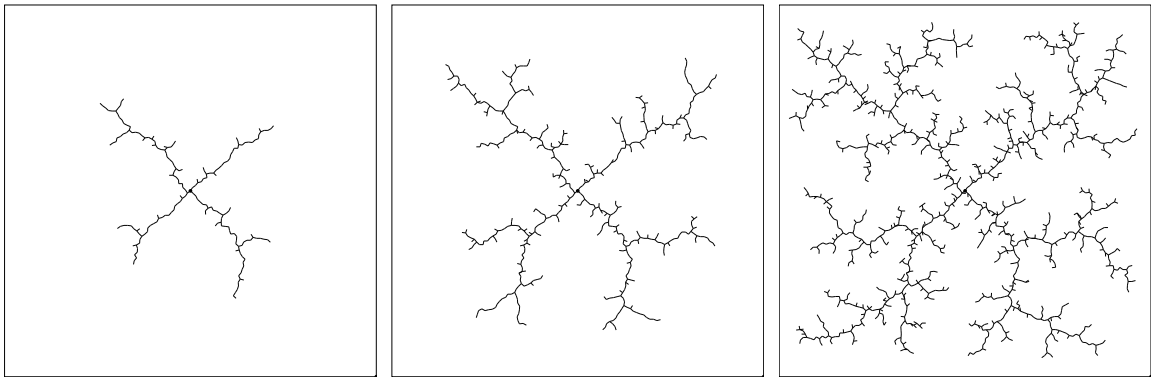


Figure 3-3: Rapidly exploring Random Trees (RRTs) in practice: An RRT uniformly exploring a 2D configuration space of a robot (adapted from (Lavalle, 1998)).

many degrees of freedom. They can efficiently search high dimensional and non-convex spaces.

RRTs work by incrementally building a tree from the start point until the tree reaches the end point. Thus, RRTs search for a path from **start** to **end** by expanding a search tree in the following manner (Akkaya):

1. Identify **start** and **end** configurations for the robot.

2. Initialize the tree with **start** as its root.

3. Randomly pick a point in the robot configuration space. Every point in space has equal probability of being chosen.

4. Identify the vertex in the tree that is closest to the newly generated point in 3.

5. Generate a new tree branch from this vertex by moving a certain fraction of the displacement towards the point generated in 3.

6. This new branch is added to the tree and the algorithm goes back to step 3 until the tree reaches **end**.

The reader is referred to figure (3-3) for a depiction of RRTs exploring the 2-D workspace of a robot.

Advantages of RRTs

1. The expansion of RRTs is heavily biased towards unexplored portions of the search space. Thus, RRTs explore the entire search space thoroughly and efficiently (see figure (3-3)).

2. The RRT algorithm is very simple to implement, even for large degree of system configuration spaces.

3. The major advantage of RRTs is their ability to sample a large search space in a relatively short time. Thus, RRTs can be used to quickly generate feasible robotic paths.

However, the drawback of RRTs is that the path generated may not be optimal. In fact, owing to the probabilistic nature of the algorithm, different paths are generated for every search. Therefore, an RRT alone is insufficient to completely solve a planning problem. It can be a component of other planning algorithms (LaValle), and can be used to reduce the search space. The second limitation is that the algorithm searches for the nearest neighbor vertex to every generated point in space. This can potentially be expensive. We shall further discuss the applications of RRTs in some robotic and underwater path planning algorithms later in this chapter.

3.1.3 Robotic Path Planning Algorithms

Robot motion planning and control is the problem of automatic construction of robot control strategies from task specifications given in high level, human-like language (Belta et al., 2007). A common approach to this problem is based on a three-level process:

1. At the first level, the obstacle-free configuration space of the robot is partitioned into cells and adjacency relations between cells is determined. The result is presented in the form of a graph. (see §(3.1.1)).

2. At the second level, a path on this graph is chosen, for example, using an optimality criterion penalizing the distance traveled, and/or proximity to obstacles.

3. At the third level, an optimization is performed over all the possible trajectories generated in step 2. Robotic controllers are constructed so that this optimal trajectory is followed.

Traditionally, robotic path planning has focussed on generating safe trajectories for the robot, away from hazardous regions and obstacles. Motion planning algorithms for multi-degree freedom systems such as robotic arms have also been extensively studied. Several approaches for solving these problems have been proposed in literature. In addition, cooperative control and coordinated motion of multiple robots have also received a lot of attention. Here, coordination is either in the form of maintaining specific geometric shapes (Leonard and Fiorelli, 2001) or obtaining maximum information from an unknown environment (Bahr et al., 2009, Davis et al., 2009). Even though many prior works have addressed general robotic path planning and its variants, path planning through time dependent fields has received far lesser attention. This task is challenging because the currents directly affect the displacement of the vehicle, making the cost of movement variable, and anisotropic at different points in space (Isern-Gonzlez et al., 2012).

The objective of robotic path planning is different from that of underwater path planning. Robotic motion planning mainly focuses on obstacle avoidance. One of the challenges in these problems is the potentially large number of degrees of freedom of the robot. Due to this, the configuration space of robotic trajectories is very large. There is strong evidence that the solution requires exponential time in the number of dimensions of this configuration space (Barraquand et al., 1996). Reif (1979) has shown that robot path planning is a hard problem. For example, consider the task of navigating a robot arm from one configuration to another. The number of degrees of freedom of the arm is at least four. For more complicated systems, the number of degrees of freedom can be as large as twenty. In fact, RRTs are the only algorithms which are capable of handling such high dimensional problems, till date (Barraquand and Latombe, 1991), since optimality is extremely difficult to achieve in these cases.

Robot motion may also be highly non-holonomic. This means that the number of variables used to describe the configuration of the robot may be smaller than

the number of degrees of freedom. Other complexities in robotic path planning can include field mapping, uncertainties, movable objects and moving obstacles, limited field visibility and other dynamic constraints (Latombe, 1991). Every extension to the basic problem adds in computational complexity (Latombe, 1995). For instance, allowing moving obstacles makes the problem exponential in the number of moving obstacles (Canny, 1988, Reif and Sharir, 1994). Uncertainties in motion and sensing make the problem exponential in the complexity of robot environment (Canny, 1988). Therefore, the requirement of optimality in a robotic path planning algorithm is typically relaxed and replaced by an ability to quickly generate feasible trajectories.

Most of the algorithms for robotic path planning have been developed by extending ideas of dynamic programming based methods such as Dijkstra's algorithm and the A^* algorithm (Rhoads et al., 2010). Most such algorithms are not applicable for dynamic flow environments either because they often lead to infeasible solutions or due to their enormous computational cost when the environment becomes complex. In addition, these algorithms compute discrete paths, i.e. on a grid based representation of the environment. Hence, when extended to a continuous setting, the paths do not remain optimal. Finally, there is a possibility for the algorithms to be stuck in a local optima and not yield globally optimal, continuous trajectories.

A randomized approach to path planning for obstacle avoidance was first developed by Lavalle (1998) (see also Kuffner and LaValle (2000)). Their seminal papers describe the use of Rapidly exploring Random Trees (RRTs) in path planning. As described in §(3.1.2), RRTs use ideas of random sampling and probabilistic completeness to explore the workspace of the robot. They have formed the basis of a variety of path planning algorithms developed in recent years (Yang et al., 2010, Bruce and Veloso, 2002, Melchior and Simmons, 2007, Jaillet et al., 2010, Karaman and Frazzoli, 2011). They are easy to implement and can quickly generate feasible trajectories. As noted earlier, the main advantage of RRTs is their ability to sample a high dimensional space in a short time. This feature however, is not as helpful in the context of long range underwater glider path planning due to the low dimensional nature of the problem.

We do not aim to provide a comprehensive overview of robotic path planning algorithms. We only seek to highlight the differences between the robotic path planning and underwater path planning. Some methods of robotic path planning have been described because of their extensions to underwater path planning.

3.2 Underwater Path Planning

Underwater path planning focuses on AUVs, such as gliders. These autonomous vehicles execute long range sampling missions (order of days) and hence, it becomes important to optimize performance criteria such as energy spent by the AUV or the time of travel, in addition to generating safe and collision free trajectories. The number of degrees of freedom for long range AUVs is two, if motion in vertical direction is neglected. This is because, for long range path planning, the motion of the robot can be considered completely holonomic. If vertical motion is considered, the number of degrees of freedom increases to 3. Thus, the challenge of underwater path planning does not arise from the large dimensionality of the configuration space. The difficulty here, arises due to the following factors:

1. The motion of the vehicle is continuous and is governed by equation (2.1). Hence, the displacement of the vehicle is affected by environmental factors such as the flow-field. Consequently, its trajectory $\mathbf{p}(t)$ is different from its heading direction ($\hat{\mathbf{h}}(t)$).

2. The number of control choices available to the vehicle at each point in its trajectory is infinite. Thus, generation of feasible tracks itself is a challenging problem.

3. The flow-field ($\mathbf{V}(\mathbf{x}, t)$) experienced by the vehicle can be strong, with large spatio-temporal variability.

4. Underwater path planning is usually based on optimizing travel time, energy or information gained by the vehicle.

In what follows, we describe some relevant literature on underwater path planning. Most methods for underwater path planning either fail when the environment becomes complex, or are computationally expensive thus making them unsuitable for real time applications with large number of vehicles. Well established methods in robotic path planning applications have not been designed to handle situations with dynamic environments, such as flow-fields. A recent trend in research on path planning methods has been to develop algorithms which use the dynamic nature of the environment to reduce the energy expended by the vehicle. A closely related problem is to obtain paths which minimize the total travel time of the vehicle when propelled at nominal speed.

Alvarez et al. (2004) propose a genetic algorithm based on Darwinian theories of natural selection for energy optimal path planning in strong ocean currents. A set of feasible paths is generated and these paths are iteratively transformed by using genetic operators like crossover and mutation. The path that minimizes a suitable cost function is chosen. This method can however, lead to suboptimal solutions due to the high dimensionality of the optimization problem.

In (Yilmaz et al., 2008), a path planning scheme based on mixed integer linear programming (MILP) is presented. Their work focuses on the problem of adaptive sampling in the ocean. Adaptive sampling refers to the task of predicting the types and locations of ocean measurements that would be most useful to collect (Lermusiaux, 2007). The usefulness of measurements is governed by an objective function and the path planning algorithm finds a vehicle path along which the line integral of this objective function is optimized. Physical constraints on vehicle motion and obstacle avoidance are translated as linear constraints in the optimization problem. Even though additional constraints on vehicle motion are easily incorporated, the computational effort required can quickly escalate due to the NP-hard nature of the problem, even though the uncertainty fields used in the cost function are assumed fixed and ocean currents are ignored. The posed problem is NP-hard even though uncertainty fields are assumed to be stationary. However, the biggest limitation of their method is that the effect of ocean currents on vehicle motion is ignored. Thus,

the method needs to be modified to incorporate the effect of flow-fields before it can be applied for gliders.

Several researchers have attempted to address underwater path planning using graph search techniques. As explained in §(3.1.1), this involves discretizing the workspace of the robot into a grid and restricting the movement of the vehicle onto this fixed grid. Dynamic programming can be employed as a graph-searching procedure when a cost associated to each arc of the graph is known (Bryson and Ho, 1975). While dynamic programming produces the optimal solution, the computational time needed is proportional to the number of nodes in the graph, which in turn, is dependent on the gridding (finer, coarser) of the solution space and increases geometrically with the dimension of the solution space. In the case of 2-D AUV navigation in space- and time-varying environments, the solution space is three-dimensional. Here, dynamic programming may not be computationally feasible, especially in cases where time-optimal paths are desired.

Instead of dynamic programming, A* method can also be used for the graph search. Although in its original version, A* deals with stationary fields, it can be modified to account for the effects of velocity fields around the vehicle. Applications of the A* search scheme for path planning of AUVs in the ocean is described by Garau et al. (2005) and also by Carroll et al. (1992). In their work, energy optimal paths are calculated in a simulated ocean environment with high spatial variability in the form of different types of eddies. The effect of different heuristic functions ($h(x)$) on the performance of the A* scheme is analyzed. The main drawback of their approach is that the ocean currents are assumed to be steady. For other applications of A* search in AUV path planning using real ocean data, we refer the reader to (Rao and Williams, 2009, Garau et al., 2009).

A major difficulty in the A* approach is in defining a heuristic that works for all types of flow-fields. The computational time of the A* method is highly dependent on the quality of the heuristic cost function (see §(3.1.1)). Some choices of the heuristic may lead to extremely large computational times due to the large size of the *open list* (see §(3.1.1)). Defining an appropriate heuristic is thus, very crucial for the success

of the A* algorithm. In cases where time optimal vehicle paths are desired, the cost function of the A* algorithm is expressed in terms of time of travel. In many cases, the only obvious choice of an admissible heuristic function is

$$h(\mathbf{x}) = \frac{d(\mathbf{x}, \mathbf{x_f})}{\max_{\mathbf{x},t} |\mathbf{V}(\mathbf{x}, t)| + F} \tag{3.2}$$

Here, $d(\mathbf{x}, \mathbf{x_f})$ is the Euclidean distance between the point $\mathbf{x}$ and the end point $\mathbf{x_f}$. $\max_{\mathbf{x},t} |\mathbf{V}(\mathbf{x}, t)|$ is the upper limit on the magnitude of the flow-field. This choice of $h(x)$ is admissible because it always underestimates the time taken by the vehicle to go from $\mathbf{x}$ to $\mathbf{x_f}$. Therefore, this heuristic will guarantee a grid based optimal path. However, this $h(x)$ may be too conservative in practice. It will be particularly bad when the flow field is very strong, but lasts for a short time. In such cases, the A* algorithm approaches Dijkstra's algorithm with this heuristic function (Yigit, 2011). Hence, the algorithm may take a lot of time to compute the optimal solution.

A* method performs reasonably well for simple flow-fields, but is not as good when the flow becomes complex. This occurs because A* uses a grid based representation of the domain and the vehicle trajectory may not always pass through the grid points. To overcome this, non-uniform grids need to be used. Since there is no way of knowing which grid to use beforehand, this method needs several iterations to compute the correct solution. In other words, the inaccuracy arises because of the fixed grid representation. Finally, the computational time for the A* method increases significantly when the flow field becomes time dependent because the algorithm requires an additional search space in this case. To be guaranteed an optimal solution, no branches of the graph search can be trimmed, which makes the worst-case computational time exponential in the size of the search space. We shall discuss the computational cost of the A* method in detail in §(6.6).

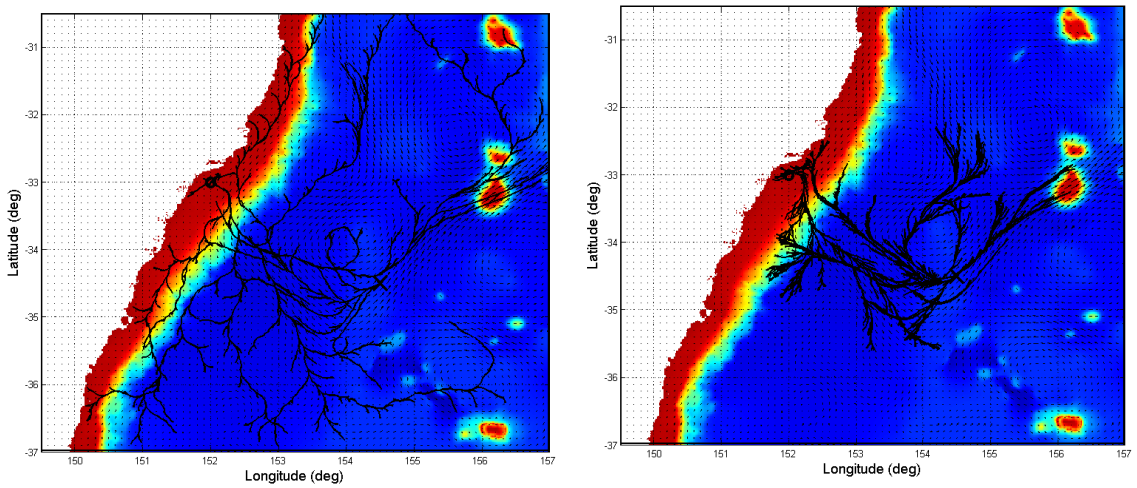
In a related work, Bakolas and Tsiotras (2010a) describe an approximate solution to the steady Zermelo navigation problem by reducing the continuous time problem into a shortest path problem over a directed graph. The domain is divided into sub-polygons and the velocity field is averaged over these polygonal regions, reducing the

problem to a finite dimensional search. The existence of such polygonal subdivisions is assumed to be known. Then a representative point on each polygonal region is computed and shortest paths to adjacent representative points are joined to obtain the complete trajectory. Path planning for unmanned aerial vehicles in steady winds, with a specified limit on their turning rate is discussed in (Techy and Woolsey, 2009).

Several authors have also pursued the idea of Voronoi diagrams to solve the obstacle avoidance problem with generation of safe trajectories for vehicles. In (Garrido et al., 2006), a Voronoi diagram of the environment is constructed and fast marching methods are then used to compute smooth and safe trajectories. Bakolas and Tsiotras (2010b) solve the same problem, but consider the effect of flow fields. Nishida et al. (2007) use a particle tracking method to compute a time dependent Voronoi diagram. However, particle tracking can lead to unresolved contours and overlapping boundaries of the Voronoi. To overcome this, they propose an independent particle tracking method that handle singularities in the Voronoi edges.

As described in §(3.1.2), Rapidly exploring Random Trees (RRTs) have been widely used in robotic path planning and they perform particularly well in situations with dynamic obstacles. This randomized approach to path planning has also been used for underwater vehicles to obtain obstacle free paths (Tan et al., 2004). Rao and Williams (2009) have used RRTs for glider path planning in dynamic flow-fields. It is interesting to note that the RRT growth is inherently biased in the direction of ocean currents. In other words, since the kinematics of the vehicle motion equation (2.1) is respected, the growth of the RRT branches will be biased along the direction of the flow-fields, especially when they are strong. Using this idea, several feasible tracks are generated and the one that minimizes the total time or energy is chosen (see figure (3-4a)). Though continuous tracks are generated, they are still not optimal because the entire search space is not explored.

The results of the RRT approach can be improved by heuristically biasing the growth of RRTs, as shown by Rao and Williams (2009). They describe the usage of a biased RRT algorithm named Iterative k-Nearest RRT (IkRRT), originally proposed by Urmson and Simmons (2003). This bias ensures that the tree growth is



(a) No biasing (b) Biasing included

Figure 3-4: Growth of RRTs in a real ocean velocity field: (a) When no external biasing is applied to the RRTs and (b) With an additional biasing (for details, see (Rao and Williams, 2009)) (figure adapted from (Rao and Williams, 2009)).

extensive enough to ensure optimal solutions are not overlooked, while being small enough to avoid expansion into unnecessary areas (for implementation details, see (Rao and Williams, 2009)). The RRT obtained after implementing this heuristic bias is depicted in figure (3-4b). Since RRTs are based on random sampling, different planning runs will lead to different feasible paths. In figure (3-5), we show the feasible trajectories computed using the above biasing rule. Thus, RRTs show a lot of promise in generating feasible glider trajectories in dynamic currents.

Another class of robotic collision avoidance algorithms that use potential field techniques was introduced by Warren (1990). Since then, it has been widely used by the robotics community and many problem specific developments have been made to this algorithm (see Barraquand et al. (1992)). The key idea of this approach is to introduce an artificial potential field on the obstacles that prevents vehicles from getting very close to them, thus, generating safe paths. The algorithm has the advantage of being inexpensive, thus allowing for easy real-time computations to adapt the vehicle path. However, it has the drawback of producing locally optimal solutions. Potential fields have also been used for underwater path planning in (Witt and Dunbabin, 2008) with a cost function measuring the total drag experienced by

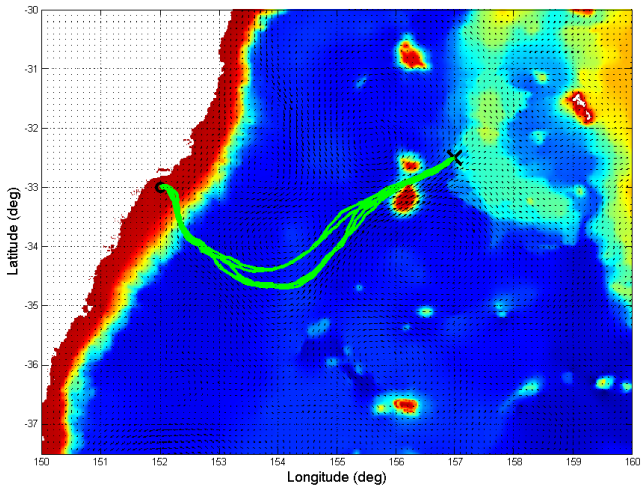


Figure 3-5: Multiple feasible vehicle tracks obtained from using the IkRRT biasing rule. The black circle and the black cross indicate the start point and the end point respectively (for details, see (Rao and Williams, 2009)) (figure adapted from (Rao and Williams, 2009)).

the vehicle, total travel time and any obstacles in the field. After generating a feasible set of tracks, an optimization is performed on these tracks.

Path planning for underwater gliders using a variational calculus approach is described by (Davis et al., 2009). They derive 'ray' equations for routing gliders through steady velocity fields. Under these restrictions, their method is similar to ray tracing for non-dispersive waves. Governing equations for minimal time routes in steady flows are derived and related to Snell's law in optics. They use a 'shooting method' to solve the governing ray equations. This entails repeatedly changing the initial conditions of the equation and solving the equation until the final conditions have been satisfied. They also discuss routing strategies for multiple underwater gliders in order to maximize the mapping skill over the entire field. However, the possibility of time varying flow-fields is not discussed, even though their work is directed towards practical ocean applications.

Jarvis and Byrne (1986) introduce a wavefront expansion algorithm for obstacle avoidance using ideas of the 'distance transform' methodology. Distance transforms have been used for path planning in stationary fields. This approach propagates a distance wave through the domain, from start to goal. The shortest path to the goal

is then traced by following the steepest descent. A comparison of A* search, RRTs and distance transforms is presented in (Jarvis, 2006).

Tsitsiklis (1995) introduces a first order 'fast marching' scheme. This scheme solves the discretized Hamilton-Jacobi equation for a trajectory optimization problem in steady fields (see also (Bryson and Ho, 1975)). This algorithm obtains the viscosity solution through a control-theory based optimality criterion. The fast marching algorithm for continuous trajectory optimization is similar to a continuous version of Dijkstra's algorithm. We describe fast marching methods in detail, in §(4.3.2). Fast marching methods were solved using a higher order numerical scheme by Sethian (1999a). Since this development, fast marching methods have found several applications in path planning in both general robotic applications and underwater applications. The method involves solving an *Eikonal equation* (Sethian, 1999b) to isotropically compute the arrival time function at different points. The optimal path is then computed by a gradient descent on this arrival time field.

A continuous approach to path planning in a field of currents is presented by Petres et al. (2007). They describe a fast marching algorithm for path planning using anisotropic cost functions. Due to the dynamic nature of ocean currents, isotropic cost functions cannot be used. Directional constraints such as those enforced by ocean currents are taken into account using these cost functions. The anisotropy is enforced by introducing a variable speed of propagation at different parts of the domain. A drawback of this scheme is that it only accommodates linear energy cost functions.

Soulignac et al. (2009) improve the scheme proposed by Petres et al. (2007) to yield better results for vehicle motion in such strong currents. The technique uses a 'symbolic wavefront expansion' to calculate shortest time paths and also determines the departure time of the vehicle from the starting point. More recently, Soulignac (2011) introduces a 'sliding wavefront expansion' technique for path planning in strong currents. The algorithm combines appropriate cost functions with continuous optimization techniques to guarantee the existence of a feasible path. Other references for path planning using a wavefront expansion are (Thompson et al., 2010, 2009). Kruger et al. (2007) also use a similar approach to minimize the total energy spent by the

AUV. They discuss path parameterizations, suitable energy based cost functions and optimization techniques to enable efficient generation of optimal paths.

More recently, Choi and How (2010) presented an algorithm for continuous trajectory planning of mobile sensors to reduce uncertainty in some quantities of interest in the future. The objective reward criterion used is the mutual information between the measurement along the continuous path and the verification variables at a future verification time. Vehicle trajectories are predicted based on the maximization of this reward criterion. The reader is referred to (Binney et al., 2010, Smith et al., 2010) and references therein for further works on path planning for information maximization.

Zhang et al. (2008) use Lagrangian Coherent Structures (LCS) to compute near optimal vehicle tracks for underwater gliders. The paper describes a control theory based method that focuses on generation of nearly optimal trajectories based on the B-spline approach. B-splines are a type of dynamic splines and are useful methods to generate feasible solutions, especially for problems pertaining to trajectory generation in control theory. Another control theoretic work that uses LCS for AUV path planning is (Inanc et al., 2005). The path planning problem is formulated as a standard constrained optimal control problem. Near-optimal solutions are obtained by converting the problem into a nonlinear optimization problem and solving it with linear programming methods (Yigit, 2011). Senatore and Ross (2008) describe another approach involving LCS but aim to solve the energy optimal AUV path planning problem.

An approach named 'Case Based Reasoning' was introduced by Vasudevan and Ganesan (1996). This approach aims to produce feasible AUV trajectories. Their paper introduces a computational method that utilizes previously gathered data and information to compute solutions to the planning problem. Trajectories are produced by modifying older trajectories or by synthesizing the acquired experiences with planned trajectories in the operation region (Yigit, 2011). Even though case based reasoning methods may be helpful in well-explored regions, it does not show a lot of promise in poorly mapped regions and when time constraints are in place.

The solution to our minimum time navigation problem in dynamic currents is governed by a Hamilton-Jacobi-Bellman equation (Bryson and Ho, 1975). Rhoads et al. (2010) derive a set of Euler-Lagrange equations that can be solved in order to extract the optimal trajectory. These equations are solved using an 'extremal' field approach. This is equivalent to solving a controllability problem backward in time. Since the arrival time at the target is unknown, the they have to solve a controllability problem for every choice of the final destination time. This approach is not ideal as it requires one to track a family of 1-D curves backward in time, for all choices of the final arrival time. The final time that leads to the correct initial conditions is chosen as the optimal solution. This algorithm can get computationally expensive because of the potentially large number of curves that need to be kept track of.

3.3 Summary

In this chapter, we have summarized relevant literature in robotic and underwater path planning methods. We have highlighted the basic differences between these two planning problems and described several methodologies in literature that solve them. Our goal in this thesis is to derive a methodology that is based on solution of governing differential equations, and provides the globally optimal solution in a computationally efficient manner. In the following chapter, we introduce the reader to the theory of front tracking and level set methods.

Chapter 4

Level Set Method

In this chapter, we introduce the reader to front tracking and level set methods. The path planning methodology that we propose in this thesis, is based on level set methods. Thus, through this chapter, we aim to give the reader a good background for level set methods. We explain theoretical properties of level sets and of their governing equation. We also discuss two different formulations of the level set equation and the advantages and drawbacks of using each one. Specifically, we present the following topics, in order.

- Level set methods and their utility in interface and front tracking
- Fast marching method
- Narrow band level set method
- Viscosity solutions of Hamilton-Jacobi equations and hyperbolic conservation laws

4.1 Front evolution and tracking

The task of tracking dynamic boundaries or interfaces between two media has a variety of applications in numerous fields of engineering, science, graphics and computer vision. For example, several phenomena in physics such as crystal growth, flame

propagation, two-phase flow, crack propagation etc. can be modeled using fronts that evolve with a curvature dependent speed. Applications of front tracking in graphics and computer vision include, but are not limited to image enhancement, noise reduction, shape detection, character recognition and morphology. Evolving interfaces in each of these fields change their shape and position according to some underlying laws of physics. Even though the physics that drives the motion of these interfaces is well understood, the task of tracking the interface can potentially be extremely challenging. In figure (4-1), we show such a front (interface) in two dimensions. In three dimensions, the interface between two media becomes a surface. The speed of evolution of this front, F, can be a function of a number of parameters (Sethian, 1999b). These parameters can be local (geometric properties such as normal, curvature etc.), global (location of the front in space, for example) or of an independent nature. F can also be either steady or time dependent. In general, the physics of the front evolution governs the nature of F. In what follows, we assume that this speed function F is known accurately.

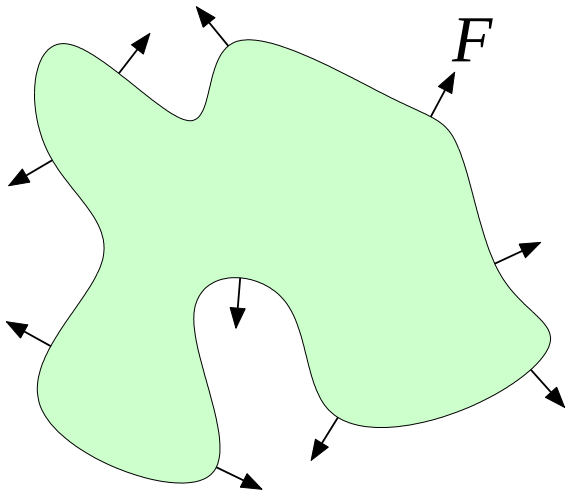


Figure 4-1: A general front in two dimensions moving at speed F (adapted from (Persson, 2010))

The objective of the front tracking problem is to develop schemes that allow us to predict the shape, position and other properties of the front as it evolves with time. Level set methods offer a convenient numerical framework to solve this problem. The

interplay between the speed function F and the shape of the front is naturally handled by level set methods.

The simplest approach, perhaps, to solving the front evolution problem is a 'marker' (or, 'string') method. As the name suggests, this method involves approximating the front by placing a number of 'marker' particles on the front. This *explicit* representation involves approximating the continuous front by a finite number of points assuming that they characterize the behavior of the front. In this explicit representation, the front is parameterized by the marker particles. The motion of these particles in space therefore, approximates the dynamics of the front evolution. Let us denote the position vector of the i^{th} marker particle on the front by $\mathbf{x}_i$. Let $\mathbf{V}(\mathbf{x}_i)$ denote the velocity of the marker particle at $\mathbf{x}_i$. In this explicit representation, the trajectory of each particle is computed according to equation (4.1). Since each particle is followed during the course of time, this is a Lagrangian approach to evolution and tracking a moving manifold.

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{V}(\mathbf{x}_i) \tag{4.1}$$

Even though this Lagrangian approach to front tracking is easy to understand, simple to implement and very fast in practice, it also has several drawbacks (Persson, 2010) (see figure (4-2)):

1. Certain velocity fields in space may cause the marker particles to concentrate in some areas of the front and be highly spread out in other areas. This happens whenever the velocity of the marker particles are highly different from each other. In such a situation, certain areas of the front are very well resolved and some areas have a poor resolution. This leads to a distortion of some parts of the curve. When this happens, it becomes crucial to redistribute the particles so that they are uniformly spread out along the front. This redistribution of particles also leads to additional errors.

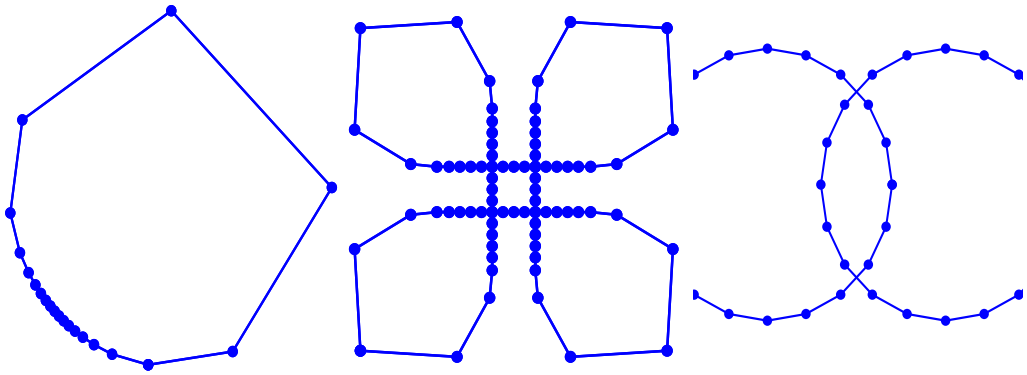
2. The marker particles may distribute themselves so that unrealistic sharp corners appear in the front. At these points, geometric quantities such as normals and

curvature of the curve will have large errors.

3. Topology changes such as front merging and splitting require special treatment. Marker particles may cross over themselves and this entails the removal of some particles from the active set at areas where two fronts have merged.

4. The Lagrangian approach also poses some stability concerns when the speed function F becomes curvature dependent (Persson, 2010).

Level set method addresses all of the above issues naturally and provides a numerical framework for front tracking. This will be described in the following section.



(a) Node Distribution (b) Sharp corners (c) Topology Changes

Figure 4-2: Drawbacks of a Lagrangian particle method for front tracking - (a) Some parts of the font are sufficiently resolved and some parts are not well resolved due to the node redistribution, (b) Unrealistic sharp corners can arise in the front and (c) Topology changes such as front merging and splitting cannot be handled. (adapted from (Persson, 2010))

4.2 Level Set Method

Level set methods overcome the above drawbacks of particle based front tracking approaches. These methods are a class of numerical techniques which were designed to solve problems related to fluid-interface motion. First introduced by Osher and Sethian (1988) to analyze flame propagation problems, level set methods have since then, been used in a variety of fields. They enable tracking of interface in systems

where front evolution is intricately connected to various physical properties of the system. They add dynamics to the front and can capture the interaction between surface motion and external forcing. The key feature of the level set method is that it utilizes an Eulerian perspective on front tracking, instead of a Lagrangian one. In other words, the front is not tracked by following particles on the curve, but instead by embedding it as an iso-contour of a function that resides in one geometric dimension higher than that of the front. This function is defined on a fixed grid and thus forms an Eulerian approach to front tracking.

The *level set* of a function $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$ is defined as the set of points at which the function takes a given constant value. Mathematically, the level set of $f(\mathbf{x})$ is the set, $\{\mathbf{x}|f(\mathbf{x}) = C\}$, where C is a given constant. The original idea behind level set methods is a simple one. Given an open region Ω in $\mathbb{R}^n$ and any hyper-surface (interface) $\partial\Omega$ in $\mathbb{R}^{n-1}$ of co-dimension 1, a smooth function, $\phi(\mathbf{x}, t)$ (at least Lipschitz continuous) is defined on $\mathbb{R}^n$ such that the interface $\partial\Omega$ forms the zero level set of the function $\phi(\mathbf{x}, t)$. The evolution of the front can then be studied by solving an initial value partial differential equation that governs the evolution of the level set function, $\phi(\mathbf{x}, t)$. This higher dimensional embedding is what allows for automatic handling of merging and pinching of fronts and other topological changes. This method is an 'implicit' representation of the front as it does not involve tracking any marker particles. In figure (4-3) we show a picture illustrating the embedding of a front as the zero level set of a higher dimensional scalar field $\phi(\mathbf{x}, t)$.

The front $\partial\Omega$ separates the space into two regions. One of the regions is 'outside' (Ω^+) the front and the other region is 'inside' (Ω^-) the front (see figure (4-3)). The level set function $\phi(\mathbf{x}, t)$ is so defined such that the sign of $\phi(\mathbf{x}, t)$ at any point in space conveys whether the point is inside or outside the front. In this thesis, we will use the convention:

$$\begin{aligned}
\phi(\mathbf{x}, t) &< 0 \quad \text{if} \quad \mathbf{x} \in \Omega^- \text{ (inside)} \\
\phi(\mathbf{x}, t) &> 0 \quad \text{if} \quad \mathbf{x} \in \Omega^+ \text{ (outside)} \\
\phi(\mathbf{x}, t) &= 0 \quad \text{if} \quad \mathbf{x} \in \partial\Omega \text{ (on the front)}
\end{aligned}$$

Some authors use alternate definitions of inside and outside, but we shall stick to above definition for the purposes of this thesis.

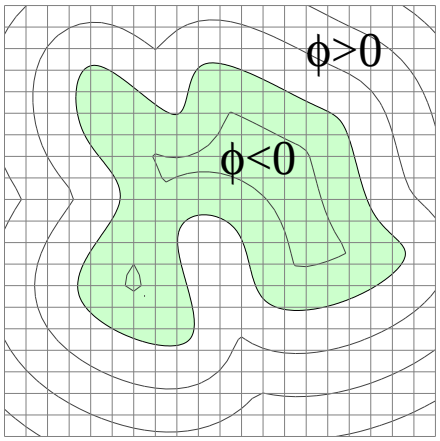


Figure 4-3: A one dimensional front being embedded as the zero level set of a two dimensional scalar field $\phi(\mathbf{x}, t)$. $\phi(\mathbf{x}, t)$ is defined over the entire 2-D domain. The front divides the domain into two regions: outside ($\phi(\mathbf{x}, t) > 0$) and inside ($\phi(\mathbf{x}, t) < 0$)-(adapted from (Persson, 2010)).

In order to track the front under a given velocity field, $\mathbf{V}(\mathbf{x}, t)$, it is sufficient to track the evolution of the level set function $\phi(\mathbf{x}, t)$ under the same velocity field. We now derive the equation that governs the evolution of $\phi(\mathbf{x}, t)$ in this velocity field $\mathbf{V}(\mathbf{x}, t)$. The solution to the level set equation is based on weak solutions to Hamilton-Jacobi equations governing hyperbolic conservation laws (Sethian, 1999b).

Consider an arbitrary iso-contour of $\phi(\mathbf{x}, t)$, say, $\phi(\mathbf{x}, t) = C$. Using chain rule of differentiation, we get:

$$\frac{\partial \phi(\mathbf{x}, t)}{\partial t} + \nabla \phi \cdot \frac{d\mathbf{x}}{dt} = 0. \tag{4.2}$$

Now, $\frac{d\mathbf{x}}{dt}$ is the velocity of any point $\mathbf{x}$ on the the iso-contour, $\phi(\mathbf{x}, t) = C$. This is identical to the velocity field experienced by the point $\mathbf{x}$. From this, we can derive the equation describing the evolution of a front moving normal to itself at constant speed $F(> 0)$ in a stationary environment (i.e. with no external flow-field). The front's motion can be thought of as an internally generated velocity field, $F\hat{\mathbf{n}}$. Since $\hat{\mathbf{n}} = \frac{\nabla \phi}{|\nabla \phi|}$, we get, $F\hat{\mathbf{n}} \cdot \nabla \phi = F\frac{\nabla \phi}{|\nabla \phi|} \cdot \nabla \phi = F|\nabla \phi|$. This gives,

$$\frac{\partial \phi}{\partial t} + F|\nabla \phi| = 0 \tag{4.3}$$

Considering now, the motion of a field $\phi(\mathbf{x}, t)$ solely driven by an external flow-velocity field $\mathbf{V}(\mathbf{x}, t)$ (i.e. $\frac{d\mathbf{x}}{dt} = \mathbf{V}(\mathbf{x}, t)$), the governing advection equation is given by the equation (4.4):

$$\frac{\partial \phi}{\partial t} + \mathbf{V}(\mathbf{x}, t) \cdot \nabla \phi = 0 \tag{4.4}$$

If in addition to the external flow-field of the equation (4.4), the front is also internally driven by its own velocity as in equation (4.3), the advection equation (4.4) becomes:

$$\frac{\partial \phi}{\partial t} + \Big(F_v(t)\hat{\mathbf{h}}(t) + \mathbf{V}(\mathbf{x}, t) \Big) \cdot \nabla \phi = 0 \tag{4.5}$$

Here, as in equation (2.1), $F_v(t)\hat{\mathbf{h}}(t)$ is the velocity relative to the flow, of a hypothetical vehicle located on the zero level set front. The magnitude of this relative velocity is F_v, with $0 \leq F_v(t) \leq F$ and its heading direction is $\hat{\mathbf{h}}(t)$.

In all of the above cases, the motion of the front can be tracked by identifying locations where the field $\phi(\mathbf{x}, t)$ vanishes (Osher and Fedkiw, 2003).

Note that sometimes, the velocity field $\mathbf{V}(\mathbf{x}, t)$ may not be known away from the location of the front. In such cases, any arbitrary velocity field can be used away from the front because we are interested only in how the zero level set front evolves. We observe that equation (4.4) is a Hamilton-Jacobi equation. Weak solutions to this equation allow the formation of singularities and shocks in the level set field. Such mild singularities are typical of correct solutions to these Hamilton-Jacobi equations. We shall further look into Hamilton-Jacobi equations and the ideas of viscosity solutions in §(4.5).

Equation (4.4) is typically solved using a numerical procedure, on a discretized domain. Thus, the accuracy of the solution procedure and the ability to handle subtle features of the front become functions of various numerical parameters such as order of discretization, time step and grid size. Thus, a natural way of accurately computing

quantities of interest emerges. Advances in numerical solution procedures, such as variable grid spacing in meshing and parallel computing may also help in reducing the computational complexity of solving equation (4.4).

The biggest drawback of level set methods arises from the computational expense. Since the front is embedded in a higher dimension, this introduces additional computational effort. However, this problem can partly be addressed by Narrow Band level set methods, as discussed in §(4.4).

We remarked earlier that the level set function $\phi(\mathbf{x}, t)$ has to be at least Lipschitz continuous. The choice of this function $\phi(\mathbf{x}, t)$ though, is somewhat arbitrary. The most common type of function used for this purpose is the *signed distance function*, denoted by $d(\mathbf{x})$. As the name suggests, a distance function, $d(\mathbf{x})$, is the shortest distance from point $\mathbf{x}$ in space to the front. Mathematically,

$$d(\mathbf{x}) = \min_{\mathbf{x_i}} |\mathbf{x} - \mathbf{x_i}|, \text{ for all } \mathbf{x_i} \in \partial\Omega \tag{4.6}$$

A signed distance function, is defined as:

$$\phi(\mathbf{x}) = \begin{cases} d(\mathbf{x}), & \text{if } \mathbf{x} \in \partial\Omega^+ \\ -d(\mathbf{x}), & \text{if } \mathbf{x} \in \partial\Omega^- \end{cases} \tag{4.7}$$

For every point $\mathbf{x_i}$ on the front, the shortest distance between the point and the front is zero, i.e. $\phi(\mathbf{x_i}) = 0$. Clearly, for every point on the front, $\phi(\mathbf{x_i}) = 0$, consistent with the zero level set representation of the front. For all points outside the front, $\phi(\mathbf{x}) > 0$, and for all points inside the front, $\phi(\mathbf{x}) < 0$. The defining property of a signed distance field is that it is the viscosity solution to the following static Hamilton-Jacobi equation (see §(4.5) for details).

$$|\nabla d(\mathbf{x})| = 1, \quad \mathbf{x} \in \Omega \tag{4.8}$$

Signed distance is a preferred choice for the implicit function because it is smooth and maintains fixed gradients in the iso-contours (Sethian, 1999b), especially near

the zero level set. From this point on, we shall refer to the front also the 'zero level set' or simply 'level set front'. We will use all the three terms interchangeably.

We now summarize the advantages offered by level set methods for front tracking (from Sethian (1999b), Osher and Fedkiw (2003)).

1. The first advantage comes from embracing a mathematical point of view for front evolution. Complexities of front motion are highlighted, in particular, the role of singularities, weak solutions, shock formation, entropy conditions and topological changes in the evolving interface.

2. From a numerical perspective, natural and accurate ways of computing delicate quantities emerge, including the ability to build higher order advection schemes, compute local curvature in two and three dimensions, track sharp corners and cusps and handle subtle topological changes of merger and breakage.

3. From an implementation point of view, since the approach is based on an initial value partial differential equation, robust schemes result from numerical parameters set at the beginning of the computation. The error is thus controlled by changing the

   (a) The order of the numerical method

   (b) The grid spacing Δx

   (c) The time step Δt

4. The computational adaptivity, both in meshing and in the computational labor is possible. In addition, the method lends itself to parallelization.

5. In the case of monotonically advancing fronts under certain speeds, fast marching methods based on merging narrow band techniques and sorting algorithms can be devised.

6. Lastly, geometric properties of the front can easily be calculated from $\phi(\mathbf{x}, t)$.

For example, the normal to level set front $\hat{\mathbf{n}}$ can be evaluated as

$$\hat{\mathbf{n}} = \frac{\nabla\phi}{|\nabla\phi|} \tag{4.9}$$

and the curvature of the level set front κ can be evaluated as

$$\kappa = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|} = \frac{\phi_{xx}\phi_y^2 - 2\phi_x\phi_y\phi_{xy} + \phi_{yy}\phi_x^2}{(\phi_x^2 + \phi_y^2)^{3/2}} \tag{4.10}$$

4.3 Boundary Value Formulation of the Level Set Equation

4.3.1 Boundary Value Formulation

The level set equation (4.4) is an initial value partial differential equation. In order to solve this equation, initial conditions on $\phi(\mathbf{x}, t)$ must to be specified. Also, in the regular level set method, equation (4.4) is solved in the entire domain at every time step. For simplicity, let us assume that the front evolves only normal to itself at speed F, and the external velocity field $\mathbf{V}(\mathbf{x}, t)$ is identically zero everywhere. Under the conditions $F > 0$, the initial value formulation of the level set evolution can be converted to a stationary boundary value problem (Sethian, 1999a). We shall prove this statement in §(5.2). This boundary value problem is significantly cheaper to solve than the initial value partial differential equation (4.4).

Consider a closed curve $\partial\Omega$ in the plane propagating normal to itself at speed F. Assuming $F > 0$, the front always moves 'outwards'. The boundary value formulation of the motion of the front uses the idea of 'arrival time'. The arrival time at position $\mathbf{x}$ is defined as the time when the front crosses $\mathbf{x}$. Therefore, the position of this expanding front can be characterized by computing the arrival time, $T(\mathbf{x})$ as it passes through each point $\mathbf{x}$ in space.

The initial value partial differential equation governing the evolution of the front

when it moves normal to itself at speed F is given by equation (4.11).

$$\frac{\partial \phi(\mathbf{x},t)}{\partial t} + F|\nabla \phi(\mathbf{x},t)| = 0 \tag{4.11}$$

The stationary boundary value formulation of this equation is given by equation (4.12) (for proof, see §(5.2)).

$$|\nabla T(\mathbf{x})|F = 1 \tag{4.12}$$

Thus, the front motion is characterized as the solution to a boundary value problem. The boundary conditions for this equation are $T = 0,$ on $\partial\Omega(0)$. The front at any time t is identical to the zero level set of $\phi(\mathbf{x},t)$ at time t, i.e. $\partial\Omega(t) = \{\mathbf{x} : \phi(\mathbf{x},t) = 0\}$. Due to the equivalence between the two formulations, the front at time t is also equivalent to the contour of the field $T(\mathbf{x})$ at level t, i.e. $\partial\Omega(t) = \{\mathbf{x} : T(\mathbf{x}) = t\}$. We

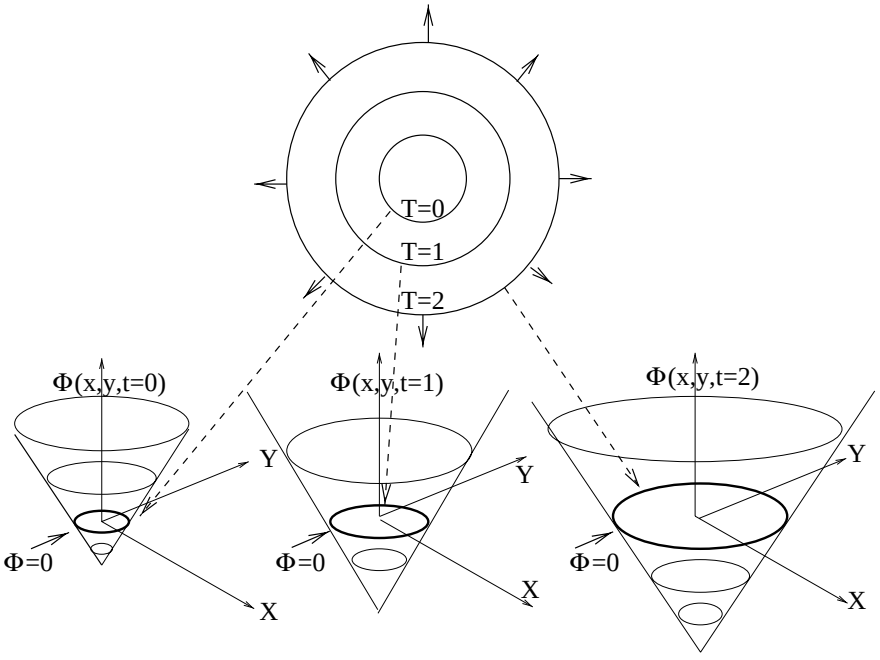


Figure 4-4: Equivalence between the initial value formulation and the boundary value formulation of the level set equation for $F > 0$. The upper plot indicates the shapes of different arrival time contours. The lower plots show the positions of the zero level set of $\phi(\mathbf{x},t)$ at three different times, $t = 0, 1, 2$. In this case, since $F > 0$, the two approaches are identical (adapted from (Sethian, 1999a)).

will depict the equivalence between these formulations with the help of figure (4-4). The upper picture shows the contours of equal arrival times at $t = 0, 1, 2$ obtained

by solving equation (4.12) with $F = 1$ in a 2D case. As expected, the arrival time contours are all circles due to the angular symmetry of expansion. The lower plots indicate the time evolution of the level set function $\phi(\mathbf{x}, t)$. The zero level set at each time has been highlighted. The shapes of the zero level set of $\phi(\mathbf{x}, t)$ are identical to the arrival time contours.

4.3.2 Fast Marching Methods

The boundary value formulation (equation (4.12)) can be solved efficiently by using a class of numerical techniques called Fast Marching Methods. They were introduced by Tsitsiklis (1995) as a continuous version of Dijkstra's algorithm (Dijkstra, 1959). Dijkstra's algorithm is a graph search technique that computes the shortest path between two nodes on a directed graph. The reader is referred to (Bryson and Ho, 1975) for a detailed illustration of Dijkstra's algorithm. Fast marching methods are numerical schemes for computing solutions to nonlinear Eikonal equations and other static Hamilton-Jacobi equations. In the most general setting, these methods solve the following nonlinear Eikonal equation.

$$\begin{aligned} |\nabla u(x)| &= f(x) \text{ in } \Omega, \quad f(x) > 0 \\ u &= g(x) \text{ on } \partial\Omega \end{aligned} \tag{4.13}$$

Fast marching methods compute a continuous viscosity solution to the above Eikonal equation (Tsitsiklis, 1995, Yigit, 2011). Sethian (1999a) proposes an upwind marching technique to solve the above equation based on ideas originally presented in (Tsitsiklis, 1995).

The method starts by numerically approximating the Eikonal equation (4.13). In a 2D problem,

$$|\nabla u(x)| \approx \begin{bmatrix} \max(D_{ij}^{-x}u, -D_{ij}^{+x}u, 0)^2 \\ +\max(D_{ij}^{-y}u, -D_{ij}^{+y}u, 0)^2 \end{bmatrix}^{1/2} = f_{ij} \tag{4.14}$$

Here, $D_{ij}^{-x}u$ denotes the first order backward difference operator etc.

$$\begin{aligned} D_{ij}^{-x}u &= \frac{u(i,j)-u(i-1,j)}{\Delta x} \\ D_{ij}^{+x}u &= \frac{u(i+1,j)-u(i,j)}{\Delta x} \\ D_{ij}^{-y}u &= \frac{u(i,j)-u(i,j-1)}{\Delta y} \\ D_{ij}^{+y}u &= \frac{u(i,j+1)-u(i,j)}{\Delta y} \end{aligned}$$

Similarly, f_{ij} is the value of the function f at node (i,j). Due to the upwind nature of equation (4.14), information propagates only 'outward', i.e. from smaller values of u to larger values. The fast marching method proceeds by building the solution outward from the smallest u value. The reader is referred to figure (4-5) for a depiction of the various steps of the upwind differencing based fast marching method (Sethian, 1999a). The algorithm proceeds in the following steps (from (Sethian, 1999a)):

1. At every step, the algorithm maintains three sets of node points. The *alive* set includes all the nodes at which u is known. All the points which are one grid cell away from the alive set constitute the *close* set. Finally, the *far* set contains points in space which are neither *close* nor *alive* (see figure (4-6)).

2. The algorithm begins by identifying all the nodes that are one grid cell away from the *alive* set. All these points are transferred to the *close* set. Points neither in the *alive* set nor the *close* set are labeled *far*. At the first step, the value of u known is known only at the boundary node (see figure (4-5)a).

3. The algorithm sweeps in a downwind fashion, updating the values of u at all the *close* points by solving the piecewise quadratic equation (4.14) (see figure (4-5)b).

4. Due to the upwind nature of the problem, no node can be affected by a node that has a larger value of u. In other words, no point that is downwind to a chosen node can affect the u value at this node. Thus, among all the updated

nodes in the *close* set, the node with the smallest value of u must be correct (see figure (4-5)c).

5. This node is now added to the *alive* set and the set of *close* points is updated (see figure (4-5)d), and the above sequence of steps repeats (see figure (4-5)e).

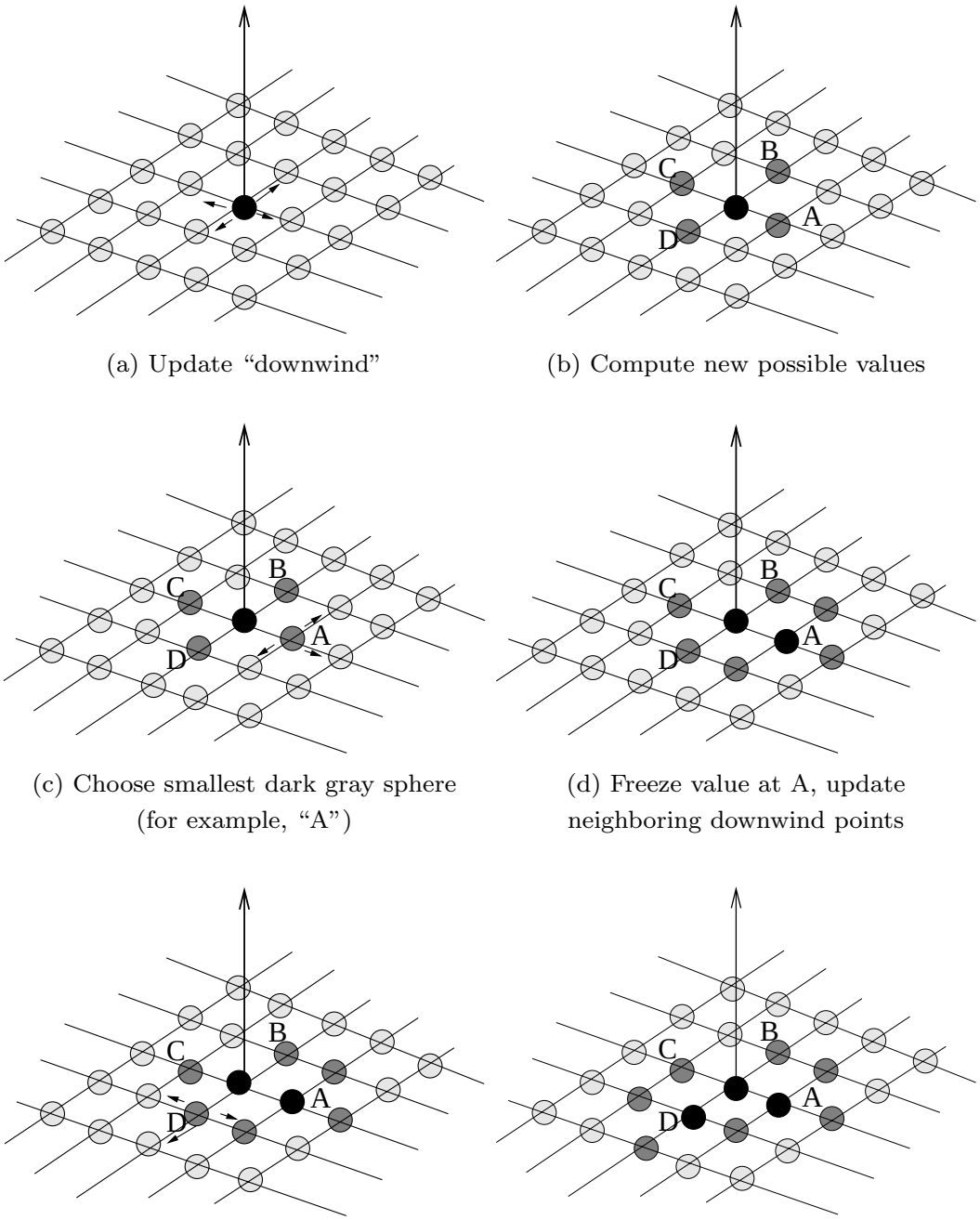
6. The algorithm marches forward until the u is solved at every node point in the domain.

4.4 Narrow Band Level Set

As explained in §(4.2), the major drawback of level set method stems from the computational expense. Level set methods work by embedding the front as the zero level set of a higher dimensional function, $\phi(\mathbf{x}, t)$ which allows us to track changes in topology and to calculate geometric quantities of interest. This embedding however, comes at a significant price - we now track *all* the level sets of $\phi(\mathbf{x}, t)$ and not just the zero level set of interest. Even though the computational expense is still manageable in two dimensional problems, the regular level set approach becomes intractable in higher dimensions. The *narrow band* level set method is a modification to the regular level set method that addresses this issue.

The idea of a *narrow band* approach for front tracking was first proposed by Chopp (1993) and later significantly developed in the context of level set methods by Adalsteinsson and Sethian (1995). The key idea of the narrow band approach arose from the observation that the level set equation (4.4) does not have to be solved far away from the zero level set because the grid points far away from the location of the zero level set have little or no impact on its evolution.

The algorithm works similar to the regular level set method, but only operates in a well-defined neighborhood of the zero level set called the *band* or the *tube*. This band only contains points that are within a certain distance to the zero level set. This distance is called the 'width' of the narrow band. We refer the reader to figure (4-7) for a visual depiction of the tube around the zero level set front. The narrow



(a) Update "downwind"

(b) Compute new possible values

(c) Choose smallest dark gray sphere (for example, "A")

(d) Freeze value at A, update neighboring downwind points

(e) Choose smallest dark gray sphere (for example, "D")

(f) Freeze value at D, update neighboring downwind points

Figure 4-5: Various steps in the update procedure of the upwind differencing based fast marching method.(adapted from (Sethian, 1999a)).

band method therefore, builds an adaptive mesh around the propagating front, i.e. a band of neighboring level sets and performs the computations only on these grid

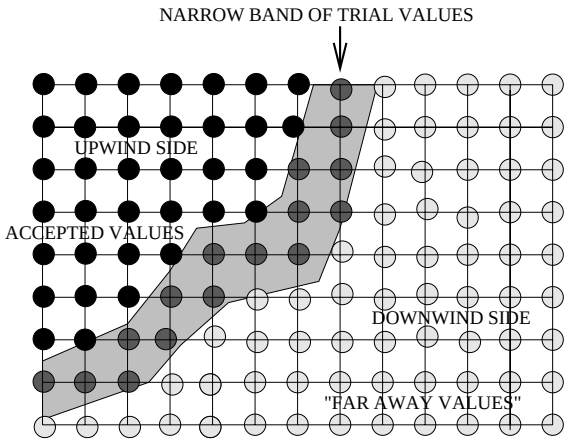


Figure 4-6: The fast marching method maintains three sets of node points: The *alive* set (shaded black) on the upwind side where the value of u is known, the *close* set (shaded dark grey) of trial values, and the *far* set on the downwind side which contains all other points in space (adapted from (Sethian, 1999a)).

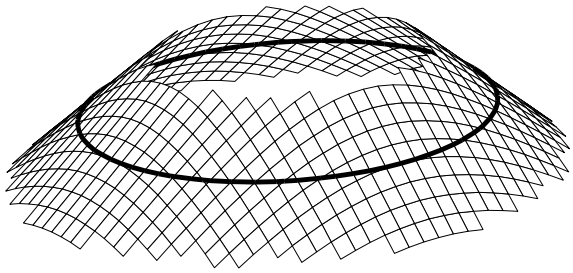


Figure 4-7: Narrow Band level set method - A *tube* or *narrow band* of points is built around the zero level set front of interest and computations are performed only within this tube. (adapted from (Adalsteinsson and Sethian, 1995)).

points. While considerable programming complexities are introduced, the decrease in computational labor is desirable in a lot of applications (Adalsteinsson and Sethian, 1995). Another reason to focus the computation only around the zero level set front is that in some problems, the velocity of propagation may only be known at the front and points close to it. Extending the velocity field at the front to other points in space is called the 'extension problem' and this process can be expensive. By limiting the computation to points near the front, the extension problem is also simplified.

The level set function is typically chosen to be the signed distance function for all points within the narrow band. The various steps of the narrow band level set method can be summarized as follows.

1. Identify the points which are within a specified distance away from the front. This is usually done by first marching a few grid cells away from the front and accurately computing the signed distance of these points from the front.

2. The front is evolved by solving the level set equation (4.4) for points within the band.

3. Update the points inside the band and go to step 1.

We refer the reader to §(6.6) for a detailed discussion on the computational cost of the narrow band level set approach and a standard level set method.

4.5 Viscosity Solutions to Hamilton Jacobi Equations

In §(4.2), we indicated that one of the main difficulties in solving the level set equation (4.4) is that the solution need not be differentiable everywhere, even with smooth initial and boundary conditions. This notion of non-differentiability of the solution is intertwined with the concept of weak solutions to a differential equation. In this section, we introduce the concept of viscosity solutions.

The numerical solution technique used to solve equation (4.4) should take into account the possibility of formation of singularities and sharp corners in the field and must lead to the 'physically correct' non-smooth solution. Thus, the numerical solution to the level set equation (4.4) is based on viscosity solutions to the related Hamilton-Jacobi equation (Fleming and Soner, 2006).

A static Hamilton-Jacobi equation governing a scalar field $u(\mathbf{x})$ is any equation of the form

$$H(u_x, u_y, u_z, x, y, z) = 0 \tag{4.15}$$

Here, the operator H is called the *Hamiltonian* of the system. The Eikonal equation (4.13) is a special case of Hamilton-Jacobi equations, with $H = |\nabla u(x)| - f(x)$. In

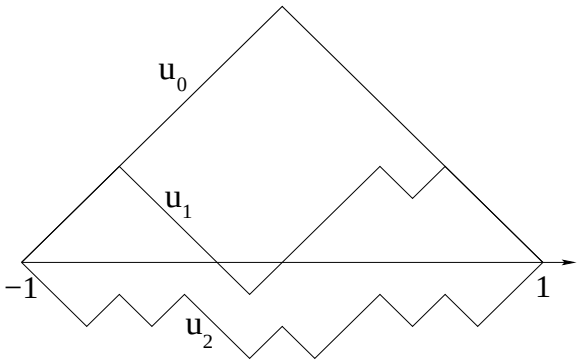


Figure 4-8: Multiple generalized solutions to equation (4.15). The problem admits several solutions and only one solution can be the correct viscosity solution. (adapted from (Bressan, 2011)).

order to understand the concept of a viscosity solution, we first present a simple example in one dimension.

Example: Consider the following static Hamilton-Jacobi equation:

$$|u_x| - 1 = 0, \quad x \in [-1, 1], \quad \text{with} \quad u(1) = u(-1) = 0 \tag{4.16}$$

This is a two point boundary value problem which admits a number of piecewise affine generalized solutions. A function u is said to be a *generalized* solution to a boundary value problem in Ω if it is Lipschitz continuous on Ω and satisfies the boundary conditions. The piecewise generalized solutions to equation (4.16) are plotted in figure (4-8). The signed distance function, $d(x)$ (see §(4.2)) is clearly a solution to this equation.

Instead of computing solutions to equation (4.16), let us compute the solutions to the following equation under the limit, $\epsilon \to 0$.

$$|u_x^\epsilon| - 1 = \epsilon u_{xx}^\epsilon, \quad x \in [-1, 1], \quad \text{with} \quad u^\epsilon(1) = u^\epsilon(-1) = 0. \tag{4.17}$$

ϵ is the numerical viscosity that has artificially been added to the equation. The name is inspired from fluid viscosity which also acts on the second derivative of the fluid speed. The solution $u(x)$ of this equation that results under the limit $\epsilon \to 0^+$ is called the *viscosity solution*. We will show, by means of a contradiction, that the viscosity

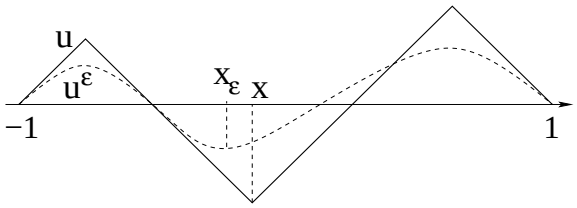


Figure 4-9: The solution to equation (4.17) does not converge to any generalized solution to equation (4.16) besides the signed distance function, $d(x)$. The signed distance field is thus, the correct viscosity solution to this equation (adapted from (Bressan, 2011)).

solution to equation (4.16) is the signed distance function $d(x)$. Let us assume that the solution to equation (4.17) is not $d(x)$, but some other generalized solution $u(x)$ (see figure (4-9)). Every such $u(x)$ has a local minima that lies strictly in $(-1, 1)$. Let us call this point y. If the viscosity solution converges to $u(x)$, we must have $u^\epsilon_x = 0$ at some point y^ϵ in the neighborhood of y. At this point,

$$u^\epsilon_x - 1 = -1 = \epsilon u^\epsilon_{xx}$$

This leads to a contradiction because if y^ϵ is a local minima of u^ϵ, then $u^\epsilon_{xx} > 0$, and ϵ is a small positive number. Therefore, the viscosity solution cannot have a local minima in $(-1, 1)$. The signed distance function is the only solution that satisfies the above requirement. Hence, the signed distance function is the correct viscosity solution to equation (4.16). Even though this discussion is not a completely rigorous proof, it serves to demonstrate the significance of viscosity solutions to hyperbolic equations. In what follows, we provide a mathematical background of viscosity solutions to steady and unsteady Hamilton-Jacobi equations and explain their relation to level set methods.

Consider a steady Hamilton-Jacobi equation of the form

$$G(x, u, \nabla u) = 0 \tag{4.18}$$

Here, $u : \Omega \to \mathbb{R}$ is a scalar function defined on an open set $\Omega \subseteq \mathbb{R}^n$. The set of

super-differentials of u at point x is defined as:

$$D^+u(x) = \left\{ p \in \mathbb{R}^n : \limsup_{y \to x} \frac{u(y) - u(x) - p \cdot (y - x)}{|y - x|} \leq 0 \right\} \tag{4.19}$$

In other words, the hyperplane $y \to u(x) + p \cdot (y - x)$ is tangent from above to the graph of u at point x (see figure (4-10)). Similarly, we can define the *sub-differential* as:

$$D^-u(x) = \left\{ p \in \mathbb{R}^n : \liminf_{y \to x} \frac{u(y) - u(x) - p \cdot (y - x)}{|y - x|} \geq 0 \right\} \tag{4.20}$$

A function u that is Lipschitz continuous in Ω is a *viscosity subsolution* of equation (4.18) if

$$G(x, u(x), p) \leq 0 \quad \text{for every } x \in \Omega, p \in D^+u(x) \tag{4.21}$$

Similarly, u is a *viscosity supersolution* of equation (4.18) if

$$G(x, u(x), p) \geq 0 \quad \text{for every } x \in \Omega, p \in D^-u(x) \tag{4.22}$$

We say that u is a **viscosity solution** of equation (4.18) if it is both a viscosity subsolution and a supersolution.

The above definition can be easily extended for the case of unsteady Hamilton-Jacobi equations as well. A general unsteady Hamilton-Jacobi equation is of the form:

$$u_t + H(t, x, u, \nabla u) = 0 \tag{4.23}$$

A Lipschitz function u is a viscosity subsolution of equation (4.23) if for every $\mathcal{C}^1$ function $\psi = \psi(x, t)$ such that $u - \psi$ has a local maximum at (t, x) there holds

$$\psi_t(x, t) + H(t, x, u, \nabla u) \leq 0 \tag{4.24}$$

Similarly, u is a viscosity supersolution of equation (4.23) if for every $\mathcal{C}^1$ function $\psi = \psi(x, t)$ such that $u - \psi$ has a local minimum at (x, t), there holds

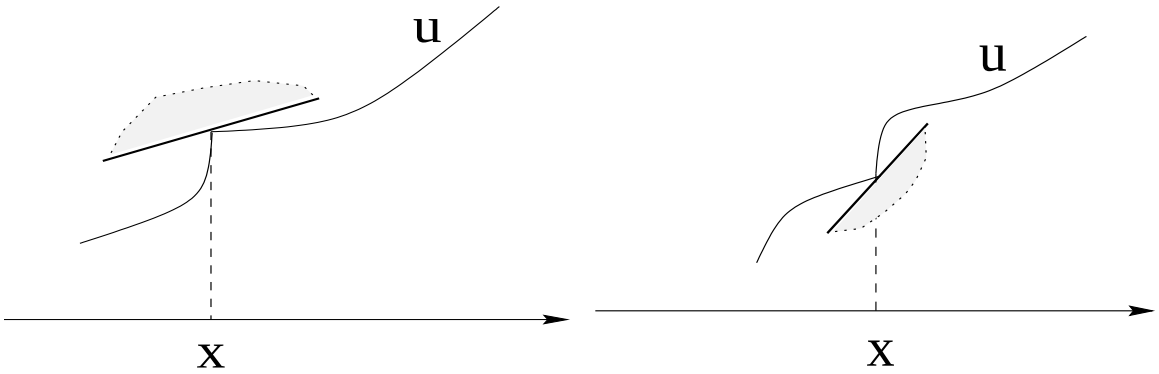
$$\psi_t(x, t) + H(t, x, u, \nabla u) \geq 0 \tag{4.25}$$

A function u is a **viscosity solution** to equation (4.23) iff it is both a viscosity subsolution and a supersolution. Note that in this definition, u is not differentiated anywhere. All the derivatives are taken only for the test function ψ. We state without proof, the following statements (see Sethian (1999a)). The interested reader is referred to (Crandall et al., 1984, 1992, Crandall and Lions, 1983, Bressan, 2011, Evans, 1998) for a better formulation and a proof of the following statements:

1. If u is a smooth solution to the Hamilton-Jacobi equation, then it is a viscosity solution.

2. If a viscosity solution u is differentiable at some point, then it satisfies the Hamilton-Jacobi equation at that point.

3. The viscosity solution is unique, for given initial conditions. Using the above statements, it can be shown that the solution produced by taking the limit $\epsilon \to 0^+$ of $u^\epsilon(x)$ is indeed this viscosity solution. The viscosity solutions are therefore, the unique solutions obtained from the smoothed Hamilton-Jacobi equations.



(a) Super-differential (b) Sub-differential

Figure 4-10: Super-differentials and sub-differentials (adapted from (Bressan, 2011)).

We note that the level set evolution is governed by an unsteady Hamilton-Jacobi equation (equation (4.4)). In order to allow mild singularities such as shocks and formation of sharp corners in $\phi(\mathbf{x}, t)$, we aim to compute the viscosity solutions to

equation (4.4). Numerical solutions to hyperbolic conservation laws use entropy conditions that are similar to those in seen in Hamilton-Jacobi equations (see for example, (Colella and Puckett, 1994, LeVeque, 2002, Sod, 1985)). Thus, numerical solutions to level set equations have been developed using this observation. Crandall et al. (1984) show that a monotone scheme for any hyperbolic conservation law converges to the correct viscosity solution. They use a first order accurate monotone scheme to solve a Hamilton-Jacobi equation. Osher and Sethian (1988) develop a higher order scheme to solve the governing equation using ideas from numerical solutions to hyperbolic conservation laws (Yigit, 2011). Therefore, viscosity solutions to Hamilton-Jacobi equations play a very crucial role in the numerical solution to any level set equation. Finally, we refer the reader to (Sethian, 1999b) for a detailed discussion on convergence and for additional numerical schemes for solving the level set equation.

4.6 Summary

In this chapter, we have presented the theory of level set methods and their advantages in tracking the propagation of fronts. We have derived an initial value formulation and a boundary value formulation for propagating fronts. Fast marching methods and narrow band level set methods have been described. Finally, we introduced the reader to a more theoretical aspect of solving the level set equations. Specifically, we defined viscosity solutions for Hamilton-Jacobi equations and their applications in the context of level sets.

Chapter 5

Path Planning using Level Set Methods

In the previous chapter, we introduced the reader to level set methods and their applications in front evolution and tracking. In this chapter, we describe our level set based path planning methodology. First, we discuss the concept of *reachability* and link it to optimal control theory and Hamilton-Jacobi-Bellman equations. Then, we state and prove a theorem which forms the foundation of our path planning methodology. Later, we present a number of corollaries to this theorem, each illustrating a unique feature of our methodology.

5.1 Control and Reachability

The computation of continuous time optimal paths in a dynamic flow-field is not a trivial problem. The complexity here arises in part, because of the number of control choices available to the vehicle. At every point in its trajectory, the vehicle has an infinite number of heading directions to choose from (see figure (5-1)). For every such heading direction chosen at a time, it will again have an infinite number of heading choices at the next instant. Thus, it is not trivial to predict the sequence of vehicle headings that will lead to the quickest path.

Instead of aiming for the exact solution, approximation solutions can always be

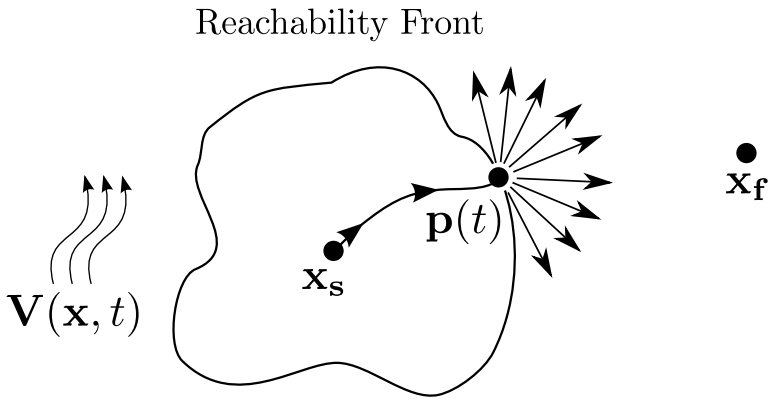


Figure 5-1: Reachability front and possible steering directions of a vehicle: Given the start point, $\mathbf{x_s}$, the end point, $\mathbf{x_f}$, the external flow-field, $\mathbf{V}(\mathbf{x}, t)$ and maximum vehicle speed F, the reachability front at time t represents the edge of the set of points the vehicle can reach within time t. At every point along its trajectory $\mathbf{p}(t)$, the vehicle has an infinite possible heading directions to choose from.

sought after. A class of practical path planning schemes are based on heuristic control decisions for the vehicle. For example, a heuristic steering rule can be to always steer in the direction of the end point. However, such approaches are not guaranteed to be optimal, neither are guaranteed to find even a feasible trajectory. The problem becomes more complicated when the flow fields are dynamic; the heuristic control then becomes a function of the velocity field, to the very least, near the vehicle. One solution to this problem could be to keep track of the vehicle trajectories for every possible control decision choice, and then choose the sequence of headings that leads to the least travel time. This method is however, extremely expensive and requires a great deal of storage.

In the case where there is no external flow-field at all, the problem is much easier to solve, and this has been done by using Fast Marching Methods (Sethian, 1999a). Our aim in this thesis is to develop a rigorous methodology to compute the optimal vehicle headings in the presence of any sort of flow-field, in an efficient manner.

Our approach to path planning is inspired by the problem of computing the *reachability set* from a given starting point. A reachability set is defined as the set of points in space that can be visited by the vehicle until a given time. The boundary of such a set is called the *reachability front*. If we keep track of the reachability front at all

times, we can determine when this front reaches the target. The path traced by the point on the reachability front that reaches the target will be the optimal path we wish to compute.

The reachable (or attainable) set, $\mathcal{R}(\mathbf{x_s}, t)$ at time t, starting from $\mathbf{x_s}$ is the set of all points of the form $\mathbf{y}_{\mathbf{x_s}}(\tau)$, where, $\mathbf{y}_{\mathbf{x_s}} \in \mathcal{S}_{[0,t]}(\mathbf{x_s})$, i.e.:

$$\mathcal{R}(\mathbf{x_s}, t) = \{\mathbf{y}_{\mathbf{x_s}} : \mathbf{y}_{\mathbf{x_s}} \in \mathcal{S}_{[0,t]}(\mathbf{x_s}), \tau \in [0, t]\} \tag{5.1}$$

Here, $\mathcal{S}_{[0,t]}(\mathbf{x_s})$ is the set of all feasible trajectories $\mathbf{y}_{\mathbf{x_s}}(t)$ which start from $\mathbf{x_s}$, and satisfy equation (2.1), with $\mathbf{y}_{\mathbf{x_s}}(0) = \mathbf{x_s}$ (Falcone and Zidani, 2012). We refer the reader to figure (5-1) for a depiction of the reachability front and the infinite number of possible heading directions that a vehicle can take, at every time.

This definition of a reachability set (and front) poses some key questions, which include: If the reachability set exists, can one prove that its evolution is directly linked to that of the the time-optimal path in any dynamic flow? What are the equations that govern the dynamics of this front and the path? How can these equations be solved efficiently? As we shall shortly see, level set methods provide leads for the answers to all of these questions.

Hamilton-Jacobi based approaches have been used in the literature to compute a reachability front (see Mitchell et al. (2005), Bokanowski et al. (2010), McShane (1937), Evans (1998)). Before we proceed, we would like to refer the reader to equations (4.3), (4.4) and (4.5). For a given level set initial condition, equation (4.5) defines a family of level set equations with each member of the family corresponding to a specific choice for the time history of the relative velocity magnitude ($F_v(t)$) and heading of the vehicle ($\hat{\mathbf{h}}(t)$).

The comparison of (4.5) to (4.3) confirms that the heading and magnitude of the relative velocity of the vehicle are the free time-dependent parameters of our problem. Specifically, it raises the following question: should time-optimal paths be those of vehicles driven in a direction normal to the time dependent level set similar to (4.3), even if the level set is externally advected as in (4.4)? We provide a theorem

and its complete proof that answers this question and sets the foundation of our approach. Using the theorem, we can demonstrate that the time optimal solution, if it exists, is indeed obtained from a combination of (4.3) and (4.5). We derive a new level set equation that governs the evolution of the reachability front and the time-optimal paths from the start point, $\mathbf{x_s}$. This theorem is an extension of a theorem provided with a sketch of proof in (Lolla et al., 2012). Remarks on existence, multiple optimal times and optimal control are given after the theorem, along with several other corollaries.

5.2 Theorem

Let $T(\mathbf{y})$ denote the first arrival time at $\mathbf{y}$, i.e. the minimum time by which a vehicle reaches $\mathbf{y}$, if it starts from the origin, $\mathbf{x_s} = \mathbf{0}$ at time $t = 0$. Let the vehicle move in a dynamic external flow field of velocity denoted by $\mathbf{V}(\mathbf{x}, t)$, with a maximum speed F relative to the flow. Consider the evolution of a Lipschitz continuous function $\phi(\mathbf{x}, t)$, governed by the following initial value partial differential equation

$$\frac{\partial \phi(\mathbf{x}, t)}{\partial t} + F|\nabla \phi(\mathbf{x}, t)| + \mathbf{V}(\mathbf{x}, t) \cdot \nabla \phi(\mathbf{x}, t) = 0 \tag{5.2}$$

with the initial condition for ϕ set to the distance from the origin $\mathbf{x_s} = \mathbf{0}$, i.e.

$$\phi(\mathbf{x}, t = 0) = \|\mathbf{x}\|_2 \tag{5.3}$$

Then,

1. $\phi(\mathbf{y}, T(\mathbf{y})) = 0$.

2. $\nexists\ t < T(\mathbf{y})$ such that $\phi(\mathbf{y}, t) = 0$, and, the optimal vehicle heading at time $t > 0$ is the unit outward normal to the zero level set of ϕ, i.e. $\{\mathbf{z} : \phi(\mathbf{z}, t) = 0\}$ passing through the current position of the vehicle: i.e. $\hat{\mathbf{h}}(t) = \frac{\nabla \phi(\mathbf{x}, t)}{|\nabla \phi(\mathbf{x}, t)|}$.

3. The optimal vehicle trajectory can be traced by solving the following equation:

$$\frac{d\mathbf{x}}{dt} = -\mathbf{V}(\mathbf{x}, t) - F\frac{\nabla\phi(\mathbf{x}, t)}{|\nabla\phi(\mathbf{x}, t)|} \tag{5.4}$$

integrating backward in time starting from $\mathbf{x} = \mathbf{y}$ at $t = T(\mathbf{y})$.

4. In the special case that $F > |\mathbf{V}(\mathbf{x}, t)| \ \forall \ \mathbf{x}$, t, then, $T(\mathbf{y})$ satisfies the following Eikonal equation:

$$F|\nabla T(\mathbf{y})| + \mathbf{V}(\mathbf{y}, T(\mathbf{y})) \cdot \nabla T(\mathbf{y}) = 1 \tag{5.5}$$

Proof

Part 1

The condition $\phi(\mathbf{y}, T(\mathbf{y})) = 0$ means that the optimal zero level set front reaches $\mathbf{y}$ at time $T(\mathbf{y})$. To show this, consider such a 1-D front, implicitly represented as the zero level set of a Lipschitz continuous scalar field, $\phi(\mathbf{x}, t)$. The equation describing the evolution of this front in a flow of velocity $\mathbf{V}(\mathbf{x}, t)$ is given by equation (4.4). Now, let us assume that in addition to this flow-field, the front also moves in a direction normal to itself at a constant relative speed, $F(> 0)$. For this relative field, the general governing equation (4.5) becomes the level set equation (5.2).

The initial condition (5.3) ensures that the zero level set is initially, a point located at the origin. The level set is a closed hyper-surface which, at the initial time, has a singularity at the origin. As time progresses, this level set evolves according to (5.2).

Let us introduce an imaginary point vehicle P, that stays on this zero level set front at all times. In other words, this point vehicle P experiences the same flow field $\mathbf{V}(\mathbf{x}, t)$ as the front and has a speed F normal to the front, relative to the flow. Therefore, this vehicle behaves as a fixed 'marker' particle on the front. Let the path traced by P be denoted by $\mathbf{x}_P(t)$. Since P is always located on the zero level set, we

have,

$$\phi(\mathbf{x}_P(t), t) = 0 \tag{5.6}$$

Now, we consider the subset of vehicles P that reach point $\mathbf{y}$. For any path taken by such vehicles reaching $\mathbf{y}$, the vehicles always stay on the front and therefore evolve according to (5.2). Hence, whenever $\mathbf{x}(t) = \mathbf{y}$, i.e. whenever such a vehicle P reaches $\mathbf{y}$ at a time t_y, we obtain using (5.6), $\phi(\mathbf{y}, t_y) = 0$. Among all of such times t_y's, $T(\mathbf{y})$ is by definition, the (yet unknown) minimum possible time and thus simply the first of all such times t_y's. Therefore, at $T(\mathbf{y})$, we also have $\phi(\mathbf{y}, T(\mathbf{y})) = 0$. This completes the proof of part 1 of the theorem. We note that this result would also hold if the evolution of the zero level set was governed by the more general equation (4.5) instead of equation (5.2). This is relevant for Part 2.

Part 2

We now need to show that the minimum of the t_y's, i.e. $T(\mathbf{y})$, is indeed obtained when the front evolves normally to itself at maximum speed F. In other words, we need to show that a sufficient condition for obtaining the fastest arrival time is that the heading of the vehicle at every time is the normal to the zero level set passing through the vehicle location at that time.

We prove this statement as follows. We consider the general set of vehicle motions governed by (2.1). To each of these corresponds one member of an infinite family of zero level set evolutions governed by equation (4.5). We compare these dual equations to the to-be-proven optimal level set equation (5.2) and to its corresponding vehicle relative velocity $F_v(t)\,\hat{\mathbf{h}}(t) = F\hat{\mathbf{n}} = F\frac{\nabla\phi}{|\nabla\phi|}$ which is governed by,

$$\frac{d\mathbf{x}}{dt} = F\frac{\nabla\phi}{|\nabla\phi|} + \mathbf{V}(\mathbf{x}, t)\,, \tag{5.7}$$

where ϕ is given by (5.2). Due to these dual correspondences, we could compare either one of (2.1) and (4.5) to either one of (5.2) and (5.7). This is because each comparison can be extended to the others by duality. Next, we choose to compare the evolution

of the general set of vehicle motions governed by (2.1) to that of to-be-proven optimal level set equation (5.2).

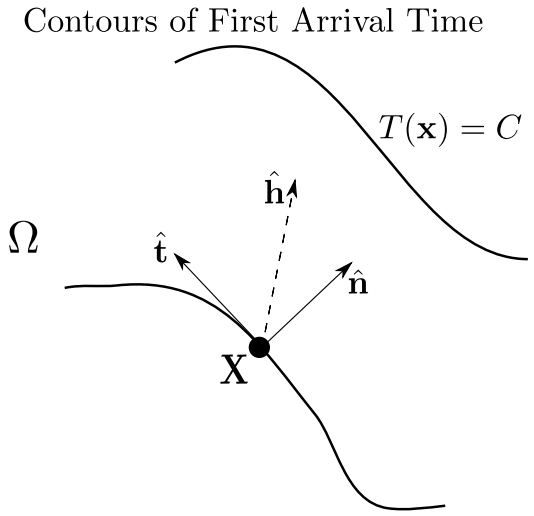


Figure 5-2: Tangential ($\hat{\mathbf{t}}$) and normal ($\hat{\mathbf{n}}$) directions to the zero level set: The general heading direction $\hat{\mathbf{h}}$ can be written as a linear combination of these two directions.

First, we thus consider any vehicle heading direction, $\hat{\mathbf{h}}$, not necessarily aligned with the normal to the front (see Fig. 5-2). This heading vector $\hat{\mathbf{h}}$ can be written as a linear combination of the unit vectors in the tangential ($\hat{\mathbf{t}}$) and normal ($\hat{\mathbf{n}}$) directions,

$$\hat{\mathbf{h}} = \alpha\hat{\mathbf{t}} + \beta\hat{\mathbf{n}} \tag{5.8}$$

where

$$\alpha^2 + \beta^2 = 1, \qquad 0 \leq |\alpha|, |\beta| \leq 1\,. \tag{5.9}$$

It should be noted that α and β are not constants, but are functions of the space and time, i.e. $\alpha(\mathbf{x}, t)$ and $\beta(\mathbf{x}, t)$. However, we simply refer to these variables as α and β with the understanding that they are control variables.

The total velocity of such a vehicle, $\mathbf{U}$, is given by (2.1), i.e. $\frac{d\mathbf{x}}{dt} = \mathbf{U}(\mathbf{x}, t) = F_v\hat{\mathbf{h}} + \mathbf{V}(\mathbf{x}, t)$, where $0 \leq F_v \leq F$ is the speed of the vehicle relative to the flow. Using (5.9), we rewrite (2.1) as

$$\frac{d\mathbf{x}}{dt} = F_v(\alpha\hat{\mathbf{t}} + \beta\hat{\mathbf{n}}) + \mathbf{V}(\mathbf{x}, t)\,. \tag{5.10}$$

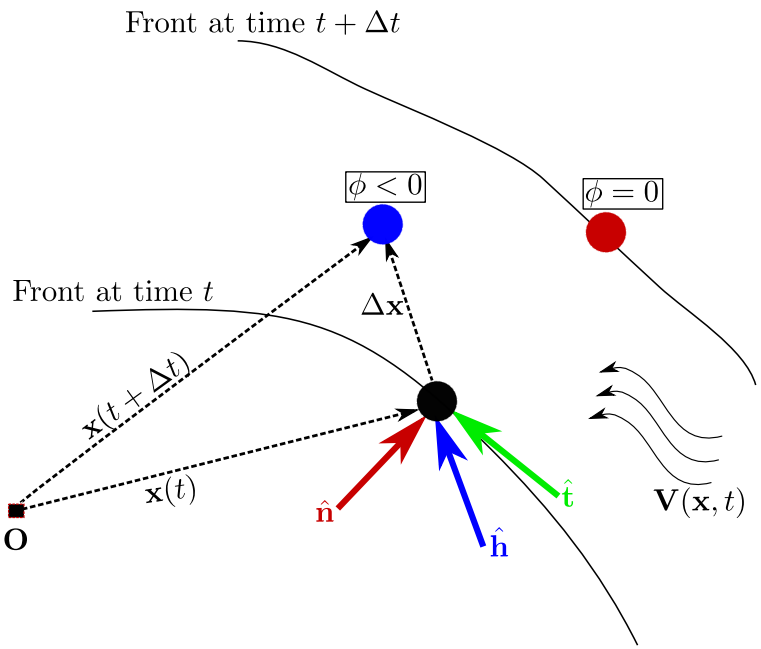


Figure 5-3: A vehicle (black circle) initially at position $\mathbf{x}(t)$ on the zero level set front at time t can be steered in one several directions ($\hat{\mathbf{h}}$). If it is steered in a direction normal to the front (i.e. $\hat{\mathbf{n}}$) and moves at maximum relative speed F, it will lie on the zero level set front at the next instant $(t+\Delta t)$. For all other steering directions $\hat{\mathbf{h}}$ and relative speeds of motion F_v, the vehicle falls 'inside' the front at the next instant.

From (5.10), we will now show that if a vehicle starts on the front, i.e. on the zero level set of ϕ governed by (5.2), all headings $\hat{\mathbf{h}}$ that differ from $\hat{\mathbf{n}}$ will lead the vehicle to be 'inside' of the front, i.e. $\phi < 0$ at the next instant. To see this, we seek the sign of $\phi(\mathbf{x}(t + \Delta t), t + \Delta t)$ under the limit $\Delta t \to 0^+$. First, we have,

$$\lim_{\Delta t \to 0} \frac{\Delta \mathbf{x}}{\Delta t} = \frac{d\mathbf{x}}{dt} = \mathbf{U}(\mathbf{x}, t) = F_v \hat{\mathbf{h}} + \mathbf{V}(\mathbf{x}, t) \tag{5.11}$$

where we defined $\Delta \mathbf{x} = \mathbf{x}(t + \Delta t) - \mathbf{x}(t)$. Clearly, since the vehicle trajectory is assumed to be continuous, this $\Delta \mathbf{x} \to \mathbf{0}$ as $\Delta t \to 0^+$. Consider now the limit:

$$\lim_{\Delta t \to 0^+} \frac{\phi(\mathbf{x} + \Delta \mathbf{x}, t + \Delta t)}{\Delta t} .$$

Since $\phi(\mathbf{x}, t)$ is Lipschitz continuous, $\lim_{\Delta t \to 0^+} \phi(\mathbf{x} + \Delta \mathbf{x}, t + \Delta t) = \phi(\mathbf{x}, t)$. Now, since the vehicle is assumed on the zero level set front at time t, we have, $\phi(\mathbf{x}, t) = 0$.

Therefore,

$$\lim_{\Delta t \to 0^+} \phi(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t) = 0, \quad \text{and} \quad \lim_{\Delta t \to 0^+} \Delta t = 0\,.$$

The conditions of L'Hôpital's rule are satisfied. Using this rule to evaluate the limit, we have:

$$\begin{aligned} \lim_{\Delta t \to 0^+} \frac{\phi(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t)}{\Delta t} &= \lim_{\Delta t \to 0^+} \frac{\nabla\phi \cdot \frac{\Delta\mathbf{x}}{\Delta t} + \frac{\partial\phi}{\partial t}}{1} \\ &= \nabla\phi \cdot \frac{d\mathbf{x}}{dt} + \frac{\partial\phi}{\partial t} \end{aligned} \tag{5.12}$$

where in the numerator, we used the fact that as $\Delta t \to 0^+$, $\frac{\Delta\mathbf{x}}{\Delta t} \to \frac{d\mathbf{x}}{dt}$ which is the total velocity of the vehicle. Substituting the expression for $\frac{d\mathbf{x}}{dt}$ from (5.10) into (5.12), we have;

$$\lim_{\Delta t \to 0^+} \frac{\phi(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t)}{\Delta t} = \frac{\partial\phi}{\partial t} + \nabla\phi \cdot (F_v\beta\hat{\mathbf{n}} + F_v\alpha\hat{\mathbf{t}} + \mathbf{V}(\mathbf{x}, t))$$

Since in our comparison ϕ is governed by (5.2), we have, $\frac{\partial\phi}{\partial t} = -F|\nabla\phi| - \mathbf{V}(\mathbf{x}, t) \cdot \nabla\phi$. Note also that the dot product $\nabla\phi \cdot \hat{\mathbf{t}} = 0$ because $\hat{\mathbf{t}}$ is perpendicular to $\nabla\phi$. Using these substitutions we obtain the expression:

$$\lim_{\Delta t \to 0^+} \frac{\phi(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t)}{\Delta t} = \nabla\phi \cdot \hat{\mathbf{n}}\,(F_v\beta - F) \tag{5.13}$$

$$\lim_{\Delta t \to 0^+} \frac{\phi(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t)}{\Delta t} = |\nabla\phi|(F_v\beta - F) \tag{5.14}$$

From this, we see that the coefficient of the leading order term in the expansion of $\phi(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t)$ is $|\nabla\phi|(F_v\beta - F)$, which is negative either if $\beta < 1$ or $F_v \neq F(\because |\nabla\phi| > 0)$. Physically, this condition states that if a vehicle starts on the zero level set and takes any direction other than the normal to the level set at that point and any speed smaller than the maximum nominal vehicle speed, then it will no longer stay on the zero level set at the next instant. In fact, it will always fall 'inside' the front. Also, if $\beta = 1$ and $F_v = F$, the vehicle stays on the zero level set at all times,

because then, the zero level set and the vehicle experience the same velocity field. It is crucial to note that there is no control for this vehicle that will take it 'outside' the zero level set governed by (5.2). The best a vehicle can do is to stay on that zero level set at all times.

At this stage, we can utilize the dualities between (5.10) and (4.5) and between (5.2) and (5.7). Specifically, to a given vehicle motion (5.10) over Δt corresponds one member of the family of zero level set evolution governed by (4.5) over the same Δt. From the above result, that level set governed by (4.5) over Δt will contain a vehicle marker point remaining within the level set governed by (5.2). This argument can be repeated for all points of the level set governed by (4.5) since they correspond to specific vehicles (5.10): it is only vehicles governed by (5.7) that remain on the level set governed by (5.2). This shows that the level set governed by (5.2) is optimal over Δt.

To complete the proof, we now show that once a vehicle is 'inside' the front (i.e. $\phi(\mathbf{x}(t), t) < 0$), it will not be able to reach 'outside' the front governed by (5.2) at any later time. Let us assume that the value of the level set function at the position of the vehicle inside the front is $\phi(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t) = \gamma < 0$. We define a new field, $\psi(\mathbf{x}, t) = \phi(\mathbf{x}, t) - \gamma$, resulting in $\psi(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t) = 0$. Since ϕ and γ differ only by a constant, their time-optimal evolution equations over Δt are identical, given by (5.2). Therefore, the vehicle is on the zero level set of the ψ field. Repeating the above arguments over successive Δt's, the value of ψ at any point on the vehicle trajectory can never become strictly positive $\psi \leq 0$, or, $\phi = \psi + \gamma \leq \gamma < 0$.

In conclusions, the zero level set governed by (5.2) will reach the goal earlier than any other level set. So, a sufficient condition to obtain a time optimal trajectory for the vehicle is to always stay on that zero level set. In other words, the value of ϕ at the goal will not take a negative value unless it was zero sometime earlier. Finally, the globally optimal trajectory can be obtained from the solution of (5.2). Hence, (5.2) solves the reachability front problem and corresponds to at least one optimal trajectory. This completes the proof of part 2.

Part 3

We have established in parts 1 and 2 of the theorem that in order to obtain the time optimal path, $F_v = F$ and $\beta = 1 \Rightarrow \alpha = 0$. Therefore, from equation (5.10), we get;

$$\frac{d\mathbf{x}}{dt} = F\hat{\mathbf{n}} + \mathbf{V}(\mathbf{x}, t) \tag{5.15}$$

The optimal path can thus be traced by solving the backtracking problem from the goal to the start, according to equation (5.4). This equation is solved backwards in time because the direction of the normal to the zero level set is not well defined at the initial time, due to the singularity at the origin.

Part 4

If the maximum speed of the vehicle (F) is, at all times, larger than the flow field speed, the level set equation (5.2) can be simplified and the resulting equation can be solved very efficiently. Mathematically, the additional assumption is:

$$F > |\mathbf{V}(\mathbf{x}, t)| \quad \forall \ \ \mathbf{x} \text{ and } t\,. \tag{5.16}$$

We now show that if (5.16) holds, (5.2) can be simplified to an equivalent steady boundary value equation (5.5) that governs the fastest or 'first arrival' time $T(\mathbf{x})$ at any location $\mathbf{x}$. In other words, $T(\mathbf{x})$ is the minimum duration in which the vehicle can reach any point $\mathbf{x}$. Note that T is *only* a function of position vector $\mathbf{x}$ and *not* of time t. Hence, $T(\mathbf{x})$ is a time-independent scalar field in space. We aim to *minimize* the first arrival time at $\mathbf{x_f}$, i.e. $T(\mathbf{x_f})$, by choosing a suitable heading direction of the vehicle (we already know from (5.2) that the vehicle has to travel at maximum speed F, regardless of the external flow).

As a preliminary, we write equation (5.2) as,

$$\frac{\partial \phi}{\partial t} + (F + \mathbf{V}(\mathbf{x}, t) \cdot \hat{\mathbf{n}})\,|\nabla \phi| = 0. \tag{5.17}$$

and show that under assumption (5.16), we have, $F + \mathbf{V}(\mathbf{x}, t) \cdot \hat{\mathbf{n}} \geq 0$ and the field $T(\mathbf{x})$ is a continuous single-valued field. To prove these two statements, we write $\mathbf{V} = V_x\hat{\mathbf{i}} + V_y\hat{\mathbf{j}}$ and $\hat{\mathbf{n}} = n_x\hat{\mathbf{i}} + n_y\hat{\mathbf{j}}$, with $n_x^2 + n_y^2 = 1$. Then, we have:

$$\begin{aligned}
(\mathbf{V}(\mathbf{x}, t) \cdot \hat{\mathbf{n}})^2 &= (V_x n_x + V_y n_y)^2 \\
&\leq (V_x^2 + V_y^2)(\underbrace{n_x^2 + n_y^2}_{=1}) \quad \text{[Cauchy-Schwarz Inequality]} \\
&= |\mathbf{V}(\mathbf{x}, t)|^2 \\
&< F^2
\end{aligned}$$

or,

$$-F < \mathbf{V}(\mathbf{x}, t) \cdot \hat{\mathbf{n}} < F \Rightarrow \mathbf{V}(\mathbf{x}, t) \cdot \hat{\mathbf{n}} + F > 0$$

Now, in eq. (5.17), $|\nabla\phi| \geq 0$ and $\mathbf{V}(\mathbf{x}, t) \cdot \hat{\mathbf{n}} + F > 0$. which proves the first part of the claim. Now, using this result in (5.17), since $|\nabla\phi| \geq 0$, we obtain,

$$\begin{aligned}
\Rightarrow \frac{\partial\phi}{\partial t} &= -\left(F + \mathbf{V}(\mathbf{x}, t) \cdot \hat{\mathbf{n}}\right)|\nabla\phi| \\
&< 0.
\end{aligned}$$

Recall that initially, $\phi(\mathbf{x}, t = 0) = \|\mathbf{x}\|_2$, and the zero level set is thus a point at the origin. Since $\frac{\partial\phi}{\partial t} < 0$, the zero level set will, at all times, expand away from the origin. Simply put, this condition ensures that the entire front always 'expands' outwards and never 'contracts'. This in turn, means that the contours of equal first arrival time C, i.e. $\{\mathbf{x} : T(\mathbf{x}) = C\}$, coincide with the zero level set front of $\phi(\mathbf{x}, t)$ at time $t = C$. Since $\phi(\mathbf{x}, t)$ is a continuous field, $T(\mathbf{x})$ will also be a continuous single-valued field in this case. Thus, the optimal path leading to any point $\mathbf{y}$ in space passes through each point along its path only once, at times equal to their respective first arrival times.

With this result, we now derive (5.5) from (5.17). Let us assume that the vehicle is located at point $\mathbf{x}$ at time $t = T(\mathbf{x})$. After an additional time Δt, the vehicle moves to another location in space, say $\mathbf{x} + \Delta\mathbf{x}$, depending on its heading direction. We can

write this as:

$$T(\mathbf{x}+\Delta\mathbf{x}) = T(\mathbf{x}) + \Delta t \tag{5.18}$$

Thus, $T(\mathbf{x}) + \Delta t$ is the first arrival time at the point $\mathbf{x} + \Delta\mathbf{x}$. From part 2 of the theorem, the heading of the optimal path $\hat{\mathbf{h}}$ is normal to the level set fronts, i.e. $\hat{\mathbf{h}} = \hat{\mathbf{n}}$. In addition, under assumption (5.16), the contours of T and of the zero level set of ϕ are identical, and the gradient of $T(\mathbf{x})$ at $\mathbf{x}$ is unique and in the normal direction to the level set, i.e. $\nabla T(\mathbf{x}) = |\nabla T(\mathbf{x})|\hat{\mathbf{n}}$. With this, by definition of the vehicle speed, we have at first order $|\Delta\mathbf{x}| = |\frac{d\mathbf{x}}{dt}|\,\Delta t$. Keeping the vector form of the vehicle's motion (e.g. in 3d in space), this is rewritten as:

$$\Delta\mathbf{x} = |\frac{d\mathbf{x}}{dt}|\,\hat{\mathbf{u}}\,\Delta t\,. \tag{5.19}$$

Now, inserting (5.18) into (5.19) and projecting the result along the vehicle's total velocity $\mathbf{U} = |U|\hat{\mathbf{u}}$, we obtain,

$$|\Delta\mathbf{x}| = |\frac{d\mathbf{x}}{dt}|\,\hat{\mathbf{u}}\cdot(T(\mathbf{x}+\Delta\mathbf{x}) - T(\mathbf{x}))\;\hat{\mathbf{u}} \tag{5.20}$$

Hence, dividing (5.20) by $|\Delta\mathbf{x}|$,

$$1 = |\frac{d\mathbf{x}}{dt}|\,\hat{\mathbf{u}}\cdot\frac{(T(\mathbf{x}+\Delta\mathbf{x}) - T(\mathbf{x}))}{|\Delta\mathbf{x}|}\,\hat{\mathbf{u}} \tag{5.21}$$

If we now take the limit for $|\Delta\mathbf{x}| \to 0$ in (5.21), since $T(\mathbf{x})$ was shown to be continuous and single valued, we obtain,

$$1 = \nabla T(\mathbf{x})\cdot\frac{d\mathbf{x}}{dt} \tag{5.22}$$

$\frac{d\mathbf{x}}{dt}$ the velocity of the vehicle with respect to a ground observer. From part 2 of the theorem, the optimal vehicle velocity is given by (5.7). Thus, inserting (5.7) into (5.22), we obtain,

$$\begin{aligned} \nabla T\cdot(F\hat{\mathbf{n}} + \mathbf{V}(\mathbf{x},T)) &= 1 \\ \text{or,}\quad F|\nabla(T)| + \mathbf{V}(\mathbf{x},T)\cdot\nabla T &= 1 \end{aligned} \tag{5.23}$$

which is the stationary Eikonal equation we wanted to derive. It provides a sufficient condition for obtaining the minimum arrival time at every point in the domain. Physically equation (5.5) means that if the vehicle heads normal to time contours of T, it will reach the destination in the shortest time. This completes the proof. $\square$.

5.3 Remarks and Corollaries

In this section, we state a few remarks and corollaries to the above theorem. Examples corroborating the following statements are presented in chapter 7, along with all other applications of the algorithm.

Existence: For a given problem configuration ($\mathbf{x_s}$, $\mathbf{x_f}$, $\mathbf{V}(\mathbf{x}, t)$, and F), the solution to the level set equation (5.2) can be used to predict whether or not the vehicle can reach $\mathbf{x_f}$ (or any given point in space) within a specified time limit, T_{max}. In some cases, the level set would never reach $\mathbf{x_f}$ in finite time, indicating that it is impossible for the vehicle to reach $\mathbf{x_f}$. In some other cases, the level set may reach $\mathbf{x_f}$, but would take longer than the allowed time, T_{max}. In this case, the vehicle cannot reach $\mathbf{x_f}$ within the time limit. In all other cases, the level set method can compute the time-optimal paths to $\mathbf{x_f}$. We refer the reader to §(7.2.1) for an illustration.

Uniqueness (single and multiple optimal paths): In some situations, there may exist multiple optimal paths to a single end point. This happens when the level set contours are not smooth at the the end point or at any other points along the path. In both cases, the path has multiple time-equivalent optimal paths. Since the level set equation (5.2) admits weak solutions, multiple optimal paths may exist if the end point lies on a region of 'shock' (where, the gradients of ϕ are undefined). The reader is referred to §(7.3.3) for an illustration.

Uniqueness (single and multiple arrival times): As seen in Part 4 of the theorem, the contours of the level set function $\phi(\mathbf{x}, t)$ coincide with those of the arrival time

field, $T(\mathbf{x})$ when the maximum relative vehicle speed F is larger than the flow speed (see figure (4-5)). The arrival time field is possibly multi-valued, because some points may be visited more than once. The first arrival time, $T_1(\mathbf{x})$ is the minimum of all such arrival times. $T_1(\mathbf{x})$ may exhibit discontinuities when the flow-field is strong and adverse. In these cases, multiple arrival times need to be stored, in order to compute the correct optimal trajectory. The Eikonal equation (5.5) is invalid in these cases. We refer the reader to §(7.3.1) for an example.

External Flow Types: The level set equation (5.2) imposes no condition on the type of flow-field $\mathbf{V}(\mathbf{x}, t)$. $\mathbf{V}(\mathbf{x}, t)$ only enforces a kinematic constraint on the vehicle's trajectory. Therefore, equation (5.2) can be used to plan paths, in principle, for compressible flows as well.

Optimal Start Time: The initial conditions in the theorem given in (5.3) indicate that the vehicle starts moving at time $t = 0$. However, in some cases, the vehicle may reach the end point $\mathbf{x_f}$ faster, if it is deployed at a later time, $t_s > 0$. §(7.3.2) discusses an example of such a scenario.

Forbidden Regions: Optimal paths of vehicles moving in dynamic flow fields may be updated/corrected when forbidden or unsafe regions are introduced in the domain. Forbidden regions refer to areas in space which the vehicle must avoid. We discuss an example of forbidden regions in §(7.2.3).

5.4 Summary

In this chapter, we have discussed a theorem that enables us to use level set methods to track the reachability fronts in unsteady flow-fields. This theorem forms the foundation of our path planning methodology. In the next chapter, we describe our path planning algorithm and other numerical details.

Chapter 6

Numerical Implementation and Discussion

The primary goal of this chapter is to describe our level set based path planning algorithm. We discuss various numerical details of its implementation, including the issue of reinitialization of the level set field. In the final section, we provide computational cost estimates of our algorithm and compare it to other algorithms for underwater path planning.

6.1 Algorithm

Our path planning algorithm consists of the following two steps:

1. **Forward Propagation**: In this first step, we expand a wavefront (the reachability front) from the start point ($\mathbf{x_s} = \mathbf{0}$) by solving the governing level set equation (5.2) forward in time, starting from the initial conditions. This equation is solved until the zero level set front reaches the end point ($\mathbf{x_f}$).

2. **Backward Particle Tracking**: Once the level set front reaches the end point, this part of the algorithm tries to identify which points on intermediate level set fronts correspond to the path that terminates at the end point. This is done by solving a particle tracking equation (5.4) backward in time. This step computes

both, the optimal vehicle path and also the sequence of its optimal headings.

6.2 Numerical Schemes

We use a first order accurate upwind scheme to approximate the $|\nabla\phi|$ term at different grid points (Sethian, 1999b, Agarwal, 2009). Let (i, j) respectively represent the grid numbers of the nodes in x and y coordinates. The first order derivatives of ϕ at the coordinates (i, j) are given by:

$$D^{+x}(\phi) = \frac{\phi(i+1,j) - \phi(i,j)}{\Delta x}, \quad D^{-x}(\phi) = \frac{\phi(i,j) - \phi(i-1,j)}{\Delta x} \tag{6.1}$$

$$D^{+y}(\phi) = \frac{\phi(i,j+1) - \phi(i,j)}{\Delta y}, \quad D^{-y}(\phi) = \frac{\phi(i,j) - \phi(i,j-1)}{\Delta y} \tag{6.2}$$

Then, $|\nabla\phi|$ is computed as:

$$|\nabla\phi| = \sqrt{(D^x(\phi))^2 + (D^y(\phi))^2} \tag{6.3}$$

where $D^x(\phi)$ and $D^y(\phi)$ are the upwind derivatives in the x and y directions.

$$D^x(\phi) = \begin{cases} D^{-x} & \text{if} \quad \phi(i-1,j) < \phi(i,j) < \phi(i+1,j) \\ D^{+x} & \text{if} \quad \phi(i-1,j) > \phi(i,j) > \phi(i+1,j) \\ \dfrac{D^{+x} + D^{-x}}{2} & \text{if} \quad \phi(i-1,j) \leq \phi(i,j) \geq \phi(i+1,j) \\ 0 & \text{if} \quad \phi(i-1,j) \geq \phi(i,j) \leq \phi(i+1,j) \end{cases} \tag{6.4}$$

Similarly,

$$D^y(\phi) = \begin{cases} D^{-y} & \text{if} \quad \phi(i,j-1) < \phi(i,j) < \phi(i,j+1) \\ D^{+y} & \text{if} \quad \phi(i,j-1) > \phi(i,j) > \phi(i,j+1) \\ \dfrac{D^{+y} + D^{-y}}{2} & \text{if} \quad \phi(i,j-1) \leq \phi(i,j) \geq \phi(i,j+1) \\ 0 & \text{if} \quad \phi(i,j-1) \geq \phi(i,j) \leq \phi(i,j+1) \end{cases} \tag{6.5}$$

The term $\mathbf{V}(\mathbf{x},t)\cdot\nabla\phi$ is discretized using a second order accurate Total Variation Diminishing (T.V.D.) scheme on a staggered C-grid (Ueckermann and Lermusiaux, 2009, 2011).

1. *Forward Time Integration:* We discretize (5.2) in time using a fractional step method as follows:

$$\frac{\phi^\star-\phi(\mathbf{x},t)}{\Delta t/2} = -F|\nabla\phi(\mathbf{x},t)| \tag{6.6}$$

$$\frac{\phi^{\star\star}-\phi^\star}{\Delta t} = -\mathbf{V}\left(\mathbf{x},t+\frac{\Delta t}{2}\right)\cdot\nabla\phi^\star \tag{6.7}$$

$$\frac{\phi(\mathbf{x},t+\Delta t)-\phi^{\star\star}}{\Delta t/2} = -F|\nabla\phi^{\star\star}| \tag{6.8}$$

Adding these equations, we get:

$$\frac{\phi(\mathbf{x},t+\Delta t)-\phi(\mathbf{x},t)}{\Delta t} = -F\frac{|\nabla\phi(\mathbf{x},t)|+|\nabla\phi^{\star\star}|}{2}-\mathbf{V}\left(\mathbf{x},t+\tfrac{\Delta t}{2}\right)\cdot\nabla\phi^\star \tag{6.9}$$

All terms of the form $|\bullet|$ are evaluated using equation (6.3). During this forward time integration, the zero level set of ϕ represents the reachability front, $\mathbf{r}_\phi(t)$. In order to be able to extract the time-optimal trajectory, we store this front at every time step. The front is extracted from the field of $\phi(\mathbf{x},t)$ using a contour extraction algorithm. For a 2-D problem, the amount of storage required for this is not significant, because the reachability front is a curve in 1-D and is numerically represented by a finite number of points.

2. *Backtracking:* The backtracking equation, (5.4) is solved using a first order forward Euler method. The extracted contour is a piecewise linear curve representing the front at that time. The forward time integration of the level set equation is performed until the first time the front reaches the goal ($\mathbf{x_f}$), i.e. $\phi(\mathbf{x_f},T(\mathbf{x_f}))=0$. However, due to the discrete nature of the numerical scheme, this condition is very difficult to achieve. We almost always overshoot the actual stopping time, by a maximum of one time-step. In other words, a more convenient stopping criterion, (T) for the scheme is the first time when $\phi(\mathbf{x_f},T)\leq 0$.

In Figure 6-1, we can see that the numerical stopping time (T) larger than the actual stopping time $T(\mathbf{x_f})$, with $0 \leq T - T(\mathbf{x_f}) \leq \Delta t$.

Hence the first step in the backtracking step is to project $\mathbf{x_f}$ on the final contour, $\mathbf{r}_\phi(T)$ to obtain a point $\mathbf{x'_f}$ closest to $\mathbf{x_f}$. Starting from $\mathbf{x'_f}$, the optimal path is extracted by solving the discretized form of equation (5.4):

$$\frac{\mathbf{x}(t-\Delta t) - \mathbf{x}(t)}{\Delta t} = -\mathbf{V}(\mathbf{x},t) - F \underbrace{\frac{\nabla\phi(\mathbf{x},t)}{|\nabla\phi(\mathbf{x},t)|}}_{\hat{\mathbf{n}}_w(\mathbf{x}',t)}. \tag{6.10}$$

with, $\mathbf{x}(T'(\mathbf{x_f})) = \mathbf{x'_f}$.

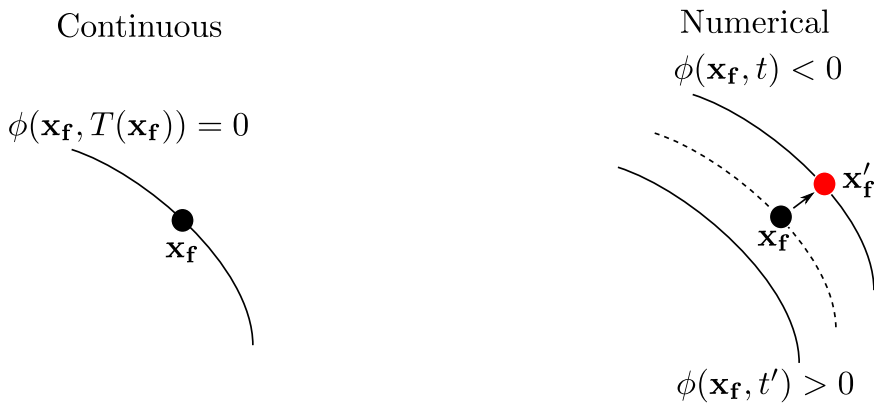


Figure 6-1: Stopping Criterion of Forward Evolution Equation

The outward weighted unit normals to the level set contours, $\hat{\mathbf{n}}_w(\mathbf{x}', t)$ are computed as follows (see Figure 6-2): If the point $\mathbf{x}'$ lies between two points representing the level set contour, the direction of $\hat{\mathbf{n}}_w$ is exactly perpendicular to the line joining the two level set points. If $\mathbf{x}'$ exactly coincides with one of the level set points, the weighted normal $\hat{\mathbf{n}}_w$ is computed as the average of the two normals to the line segments of the level set on either side of the point. Since the external velocity field is completely known, the term $\mathbf{V}(\mathbf{x}, t)$ can be computed. Now, we have everything we need to solve equation (6.10). Due to the discrete nature of the saved level set contours, the newly computed point, $\mathbf{x}(t-\Delta t)$ will not lie exactly on the saved contours $\mathbf{r}_\phi(t-\Delta t)$. Therefore, we need to perform the projection again, to compute the point on $\mathbf{r}_\phi(t-\Delta t)$ closest to $\mathbf{x}(t-\Delta t)$.

The above steps are repeated until we reach a point on the first saved level set contour. The points traced by solving the backtracking equation form a discrete representation of the time-optimal trajectory of the vehicle. Also, the optimal headings of the vehicle are equal to $\hat{\mathbf{n}}_w(\mathbf{x}, t)$.

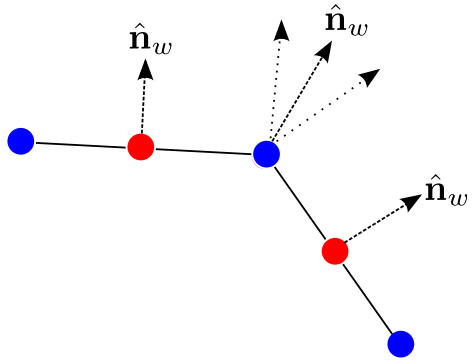


Figure 6-2: Directions of outward normals to discrete level set contours - Blue dots are the discrete points which form the contour, red dots are the points computed by backtracking equation (6.10)

6.3 Reinitialization of the level set function

In this section, we describe the issue of reinitialization of the level set field $\phi(\mathbf{x}, t)$. We first define reinitialization and discuss its importance in the context of level set methods. We discuss reinitialization in cases where there is no external flow-field, and when there exists one. Finally, we describe some techniques for reinitialization which are frequently used in the literature.

Reinitialization of the field $\phi(\mathbf{x}, t)$ is an important issue in the process of solving the level set equation (5.2). $\phi(\mathbf{x}, t)$ is usually chosen to be the signed distance function. A signed distance function has several favorable properties- it is smooth, and maintains uniform gradients in ϕ. As seen in chapter 4, the signed distance function is the unique viscosity solution of the static Eikonal equation, $|\nabla\phi| = 1$. This property of the signed distance function is numerically, very favorable.

Let us consider a scenario where the external flow-field is identically zero every-

where, i.e. $\mathbf{V}(\mathbf{x}, t) = \mathbf{0}$. In this case, the level set equation (5.2) reduces to:

$$\frac{\partial \phi}{\partial t} + F|\nabla \phi| = 0 \tag{6.11}$$

If ϕ is initialized to be the signed distance function, then $|\nabla \phi| = 1$, which gives:

$$\frac{\partial \phi}{\partial t} + F = 0$$

Taking the x-derivative of this equation;

$$\frac{\partial}{\partial t}\frac{\partial \phi}{\partial x} = 0$$

Similarly,

$$\frac{\partial}{\partial t}\frac{\partial \phi}{\partial y} = 0$$

This gives;

$$\begin{aligned} \frac{\partial |\nabla \phi|}{\partial t} &= \frac{1}{|\nabla \phi|}\left(\frac{\partial \phi}{\partial x} \underbrace{\frac{\partial}{\partial t}\left(\frac{\partial \phi}{\partial x}\right)}_{=0} + \frac{\partial \phi}{\partial y} \underbrace{\frac{\partial}{\partial t}\left(\frac{\partial \phi}{\partial y}\right)}_{=0} \right) \\ &= 0. \end{aligned}$$

Hence, $|\nabla \phi| = 1$ for all times. This means that theoretically, the level set equation (6.11), i.e. with no external velocity field, preserves the signed distance property of ϕ. However, in general, the signed distance property of ϕ is gradually lost due to numerical approximation of the various terms. This may cause the field $\phi(\mathbf{x}, t)$ to develop steep gradients at some places and shallow gradients at others. This can potentially result in large errors in the numerical approximation of the gradients, $\nabla \phi$. This directly leads to large errors in the evolution of the front and in the evaluation of its geometrical properties such as directions of normals and curvature. This problem, in general, cannot be alleviated by using a higher order scheme to approximate the gradients, or for the time integration, as shown in (Mulder et al.,

1992). Reinitialization is defined as a process in which $\phi(\mathbf{x}, t)$ is reset to a new scalar signed-distance field with the zero iso-contour being unchanged. The ideal reinitialization process should,

- Modify $\phi(\mathbf{x}, t)$ such that it now satisfies the signed distance property, $|\nabla\phi| = 1$.
- Keep the zero iso-contour of $\phi(\mathbf{x}, t)$ unchanged.

Here, we note that there is no theoretical reason why $\phi(\mathbf{x}, t)$ must be chosen as a signed distance function. Hypothetically, if one could solve equation (5.2) exactly, any Lipschitz continuous field would be a valid choice for $\phi(\mathbf{x}, t)$ because we are only interested in tracking the zero level set of $\phi(\mathbf{x}, t)$ and not its entire field. However, since the governing equations are solved numerically, this inevitably leads to errors, which accumulate with time.

Let us now consider the case where the flow-field in the domain is non-zero. Taking the x derivative of equation (5.2) and setting $|\nabla\phi| = 1$, we get;

$$\frac{\partial}{\partial t}\frac{\partial \phi}{\partial x} + \frac{\partial \mathbf{V}}{\partial x} \cdot \nabla\phi + \mathbf{V} \cdot \frac{\partial}{\partial x}(\nabla\phi) = 0$$

Similarly,

$$\frac{\partial}{\partial t}\frac{\partial \phi}{\partial y} + \frac{\partial \mathbf{V}}{\partial y} \cdot \nabla\phi + \mathbf{V} \cdot \frac{\partial}{\partial y}(\nabla\phi) = 0$$

This gives;

$$\begin{aligned}
\frac{\partial|\nabla\phi|}{\partial t} &= \frac{1}{|\nabla\phi|}\left(\frac{\partial\phi}{\partial x}\frac{\partial}{\partial t}\left(\frac{\partial\phi}{\partial x}\right) + \frac{\partial\phi}{\partial y}\frac{\partial}{\partial t}\left(\frac{\partial\phi}{\partial y}\right)\right) \\
&= \frac{-1}{|\nabla\phi|}\left(\left\{\frac{\partial\phi}{\partial x}\frac{\partial\mathbf{V}}{\partial x} + \frac{\partial\phi}{\partial y}\frac{\partial\mathbf{V}}{\partial y}\right\} \cdot \nabla\phi + \mathbf{V} \cdot \left\{\frac{\partial\phi}{\partial x}\frac{\partial}{\partial x}(\nabla\phi) + \frac{\partial\phi}{\partial y}\frac{\partial}{\partial y}(\nabla\phi)\right\}\right) \\
&\neq 0, \quad \text{in general.}
\end{aligned}$$

This means that even though initially, the field of ϕ is a signed distance function, it does not remain a signed distance function when it evolves according to equation (5.2). Note that this is not a numerical issue, unlike the case when $\mathbf{V}(\mathbf{x}, t) \equiv 0$. A completely accurate numerical procedure would still cause ϕ to deviate from a signed distance function. Since it is crucial to correctly estimate gradients of ϕ near the

zero level set, one needs to either reinitialize the level set field or increase spatial resolution around the front, in order to correctly calculate the gradients. However, reinitialization is quite expensive and a lot of research has gone into dealing with this problem.

The overall computational cost of the level set method can be influenced to a large extent, by the scheme used for reinitialization. Another concern with the level set method is the frequency at which reinitialization should be performed. Numerous techniques have been proposed in literature to make the reinitialization process more accurate and computationally efficient. The simplest technique for reinitialization involves computing the distance of every grid point to the discrete level set front and choosing the minimum of all those values. The computational cost of such a method is $\mathcal{O}(n^3)$, where n is the number of grid points in each direction (Sussman et al., 1994). This method involves an exhaustive computation of all distances which makes it expensive. Another approach for reinitialization involves solving a transient PDE to steady state (Sussman et al., 1994). However, this PDE based approach is known to suffer from the drawback of causing significant displacement of the level set front, leading to incorrect features in the front.

Recently, an approach called constrained reinitialization was presented in (Hartmann et al., 2010a,b) to address the problem of front displacement. The scheme uses a special treatment of cells adjacent to the level set front to make the reinitialization more accurate near the front. This method is known to perform well when the level set function is not far from a signed distance field, thus requiring lesser iterations to converge to steady state. Other improvements to this PDE based reinitialization scheme can be found in (Russo and Smereka, 2000).

A third approach for reinitialization is to solve the Eikonal equation $|\nabla\phi| = 1$ using a Fast Marching Method (Sethian, 1999a). The invariance of the zero level set front is the boundary condition for this method. The computational cost of this method is $\mathcal{O}(N \log N)$, which is a significant improvement over the exhaustive computation of all distances. A comparison of computational costs for different reinitialization schemes can be found in (Oberhuber, 2004).

Instead of performing a reinitialization step at frequent intervals, some researchers have also focussed on trying to maintain the signed distance property of the level set function. One of these approaches is using extension velocities away from the level set front to preserve the signed distance field (Chopp, 2009, Adalsteinsson and Sethian, 1999). This method modifies the velocity field away from the zero level set to prevent $\phi(\mathbf{x}, t)$ from deviating from a distance field. Li et al. (2005) describe a variational formulation for the level set equation that penalizes a cost function when the level set field deviates from a signed distance function. The interested reader is referred to (Jones et al., 2006, Min, 2010) for a good review of various other reinitialization techniques used in practice.

6.4 Narrow Band Approach

Since we are interested only in the evolution of the zero level set front and not in the behavior of the ϕ field away from the front, we can use a narrow band approach (Adalsteinsson and Sethian, 1995) to solve equation (5.2) efficiently. In this approach, (5.2) is solved only within a band of points around the zero level set instead of the whole domain. Due to this, significant reduction in computational effort can be achieved. Although we have implemented a narrow band solver for path planning, all the results presented in this thesis have been obtained using a regular level set solver.

6.5 Choice of level set function ϕ

The choice of the level set function is quite arbitrary. Ideally, the evolution of the zero level set front should not depend upon the choice of the field $\phi(\mathbf{x}, t)$, as long as it is Lipschitz continuous (Evans, 1998, Russo and Smereka, 2000). If (5.2) is solved exactly, any Lipschitz continuous field $\phi(\mathbf{x}, t)$ should lead to the correct time evolution reachability front and therefore, correct optimal vehicle trajectories. However, from a numerical perspective, different choices of $\phi(\mathbf{x}, t)$ lead to different solutions. It is well known that a signed distance function for the level set field has several favorable

properties. The signed distance field is smooth, i.e. does not have any sharp gradients, especially near the zero level set contour. This leads to lesser numerical errors in approximating field quantities such as $\nabla\phi$ and therefore gives a more accurate evolution of the level set front. Therefore, it is generally useful if the level set field is chosen as the signed distance function, at least close to the front, if not in the entire domain.

6.6 Computational Cost

In this section, we attempt to quantify the computational cost of our level set based path planning algorithm and compare it to other algorithms in the literature.

In our implementation, eq. (5.2) is integrated forward in time using a finite volume approach, either by a regular solver, or a narrow band solver. In both cases, since this equation is solved on a grid, the overall computational cost of the algorithm becomes a function of the grid resolution. In this thesis, we deal with path planning in two dimensions, hence equation (5.2) is solved in 2D. Let us assume that there are $\mathcal{O}(n)$ grid points in each direction and a total of $\mathcal{O}(N)$ grid points in the whole domain ($N \sim n^2$).

Starting first with the regular finite volume approach, the level set equation is solved in the entire domain. The computational cost per time step for this is $\mathcal{O}(n^2)$ (Adalsteinsson and Sethian, 1995). The number of time steps (K), that the algorithm runs is directly related to the optimal travel time of the vehicle ($K = T(\mathbf{x_f})/\Delta t$). Since $T(\mathbf{x_f})$ is not known a priori, it is not possible to estimate the total number of time steps (K) without solving the equation (5.2) to begin with. Therefore, we can only provide computational cost estimates of the algorithm per time step.

Reinitialization of the level set also comes at an expense. The simplest reinitialization procedure of computing the distance of every grid point to the level set front is an $\mathcal{O}(n^3)$ operation. However, if a fast marching method is employed to solve the reinitialization equation, the computational cost drops to $\mathcal{O}(n^2 \log(n))$ (Sethian, 1999a). Again, the overall contribution of the reinitialization step towards the com-

putational cost depends on the frequency (i.e. the number of time-steps without any reinitialization) at which the level set field is reinitialized.

The computational costs of solving both, the level set equation and the reinitialization equation can be significantly reduced by using a narrow band approach Adalsteinsson and Sethian (1995). If we assume a narrow band of width d, then the cost of solving the level set equation (5.2) reduces to $\mathcal{O}(nd)$, per time step. The cost of computing the distances to the zero level set for all points inside the narrow band reduces to $\mathcal{O}(nd^2)$. The above estimates of computational costs are tabulated in Table 6.1.

Solver	Level Set Eq. (5.2)	Reinitialization
Regular(Full PDE)	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
Narrow Band	$\mathcal{O}(nd)$	$\mathcal{O}(nd^2)$

Table 6.1: Computational cost estimates (per time step) for solving the level set equation (5.2) and the reinitialization equation: Regular solver and narrow band solver.

It is not so straight-forward, however, to come up with estimates of computational costs of other algorithms that solve the fastest time path planning problem. In addition to the approximate vehicle trajectory ensued by a grid based representation of the domain, the computational costs of the algorithms discussed in chapter 3 of this thesis are also not easy to predict. For example, let us consider the A* method. This method does not involve solving a PDE. The A* algorithm maintains an open and closed list of cells that the robot can visit (see §(3.1.1)). The open list contains all the points which can possibly lie on the optimal path, while the closed list contains all points which have been removed from the open list, i.e. they are not considered any more. The closed list contains all trajectories that have been trimmed from the graph search. In other words, the algorithm maintains a sorted priority queue of path segments along with estimates of the total cost to reach the goal. In underwater path planning, the motion of the vehicle is affected by the flow-field. Since these flow-fields are often of a dynamic nature, the costs associated with traveling across different arcs of the AUV workspace (costs here have units of time) become time-dependent. This

adds to the complexity of the graph search. Specifically, for strong flows, the optimal path to the end point may visit some points in the domain more than once. In order for the A* algorithm to provide the optimal solution, no grid point may be added to the closed list. In other words, no branch of the search tree can be trimmed. To guarantee a globally optimal solution (overlooking grid-based paths), the A* algorithm would require a time that is doubly exponential in the robot workspace, in the worst case. In addition, the heuristic cost function that underestimates the cost to go from the current point to the end point depends on the flow-field. For strong flows, the performance of the A* algorithm crucially depends on the choice of the heuristic. The computational cost of the A* method therefore, depends on a lot of factors.

We note that for some flow-fields, A* method can compute the optimal path faster than our level set based approach. For example, consider a flow-field that is directed towards the end point. In this case, the A* algorithm proceeds along a straight line towards the end point and the optimal path is thus, directly computed. However, analysis is not so simple when the flows become complex.

In addition, the performance of the method depends crucially on the choice of the heuristic cost function chosen. The algorithm maintains a sorted priority queue of path segments along with estimates of the total cost to reach the goal (travel time, in this case). When the flow field becomes time dependent, the costs of traveling across various arcs also becomes time dependent. In this case, to achieve optimality, every path segment may need be pursued, which makes the computational cost of the algorithm exponential, in the worst case. Even though the computational cost per time step is small, the number of time steps needed for optimality may be extremely large.

Randomized and approximate methods like RRTs also suffer from a similar problem in underwater path planning. Even though RRTs are quick in practice but an estimate on the computational cost to yield an optimal solution depends on the nature of the flow-field itself. The algorithm itself, owing to its random nature, is not amenable to generating robust trajectories. This algorithm is only *probabilistically complete*, i.e. exhibits favorable convergence properties in a probabilistic sense.

Therefore, it is very challenging to come up with deterministic estimates of the computational cost of RRTs. The author is not aware of any work that discusses the computational costs of various algorithms for time optimal path planning in dynamic flow currents.

6.7 Summary

In this chapter, we have presented our level set algorithm for time-optimal path planning in dynamic flow-fields. We have described the numerical scheme used for solving the governing equation. We then also discussed the topic of reinitialization of the level set field and briefly commented on how it can affect the solution. Finally, we provided estimates of the computational cost of our algorithm and compared it with two other robotic path planning algorithms.

Chapter 7

Applications

In this chapter, we illustrate our path planning algorithm by means of three sets of examples. The first set (Section 7.1) comprises of examples with simple canonical flow-fields such as jet and vortex flows. These serve as benchmark examples wherein, we compare the solution obtained by our algorithm to respective analytical solutions which are either known, or can easily be computed. In the second set of examples (Section 7.2), we utilize more complex and realistic, numerically simulated ocean flow-fields to highlight the features and capabilities of our path planning algorithm. These flow-fields are highly unsteady and include strong jets and eddies. The environment is, in some cases, also composed of obstacles and forbidden regions. In the final set (Section 7.3), we consider other interesting test cases which serve as corollaries to the theorem presented in §(5.2). Each of these examples is designed to highlight a unique feature of the algorithm. For robustness, many more cases have been studied, but we illustrate only a subset of results here.

7.1 Benchmark Examples

7.1.1 No Flow

Consider a simple example in two dimensions, in which, there is no external flow field, i.e. $\mathbf{V}(\mathbf{x}, t) = 0$ everywhere. We wish to navigate a vehicle from a start point,

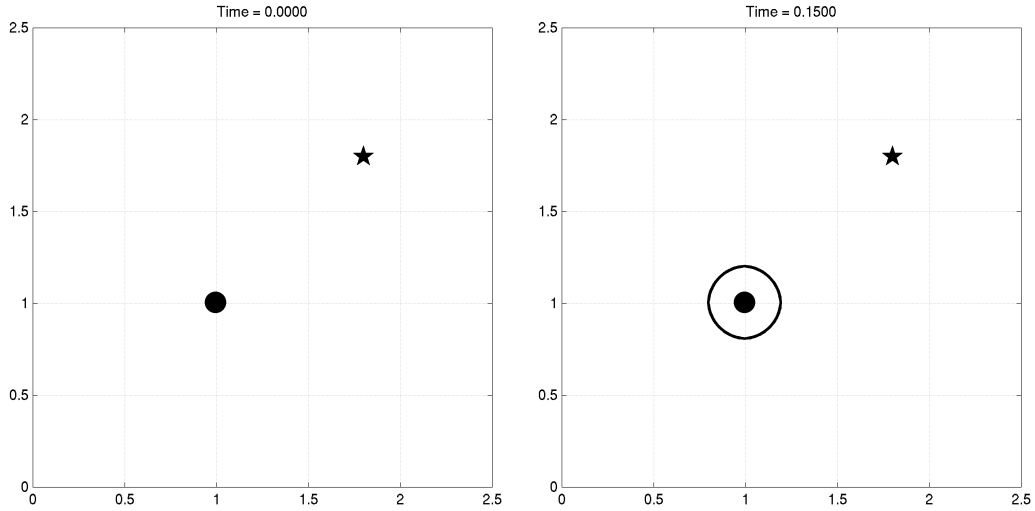
$\mathbf{x_s} = (1, 1)$ to the end point, $\mathbf{x_f} = (1.8, 1.8)$ in the fastest time. The vehicle has a maximum speed of $F = 1$ relative to the flow. The environment of the vehicle has no obstacles and the vehicle is free to visit any point in space. In this case, the shortest path is trivial - a straight line connecting $\mathbf{x_s}$ and $\mathbf{x_f}$.

We use our level set algorithm to compute the optimal trajectory for this problem. As per our algorithm, we solve the forward level set equation (5.2) starting from the initial conditions. We refer the reader to figure (7-1) for snapshots of the zero level set contour (reachability front) at various times. Since there is no flow-field in this example, the shape of the level set front is always a circle, centered at the start point, $\mathbf{x_s}$. As seen from figure (7-1), the algorithm correctly predicts the shapes of the zero level set. The forward level set equation (5.2) is solved until the zero level reaches the end point $\mathbf{x_f}$. This terminates the forward evolution step. The backtracking is performed by solving equation (5.4), starting from $\mathbf{x_f}$ until the vehicle reaches $\mathbf{x_s}$. Since there is no flow field in this example, the optimal heading of the vehicle is always in the outward radial direction from $\mathbf{x_s}$. In figure (7-2), we plot the optimal vehicle trajectory predicted by our algorithm. We can see that our algorithm correctly predicts the the optimal straight line path from $\mathbf{x_s}$ to $\mathbf{x_f}$.

In addition, for this example, the analytical optimal time of travel is equal to $T_{min}(\mathbf{x_f}) = \frac{|\mathbf{x_f} - \mathbf{x_s}|}{F} = 1.1314$. The travel time predicted by our algorithm is equal to 1.1300, which is within 0.15% of the analytical value. Therefore, the predicted solution is within the error limits imposed by the grid resolution (i.e. $\frac{\Delta x}{|\mathbf{x_s} - \mathbf{x_f}|} \approx 0.011$). Thus, the algorithm correctly predicts both the optimal path and the optimal time of travel in this problem.

7.1.2 Optimal Crossing of a Jet Flow

In this second benchmark example, we apply our path planning algorithm to optimally cross a 'jet flow' (Lolla et al., 2012). Consider a flow field (see figure (7-3)) in the form of a uniform jet, from left to right, of constant speed V. This region of the flow field is shaded in gray (see figure (7-3)). There is no flow in the rest of the domain. We wish to determine the minimum time path of the vehicle from the starting location



(a) $t = 0$ (b) $t = 0.15$

(c) $t = 0.75$ (d) $t = 1.10$

Figure 7-1: Snapshots of the zero level set (reachability front) at different non-dimensional times for example §(7.1.1). The start point is marked by a black circle, while the end point is denoted by a star.

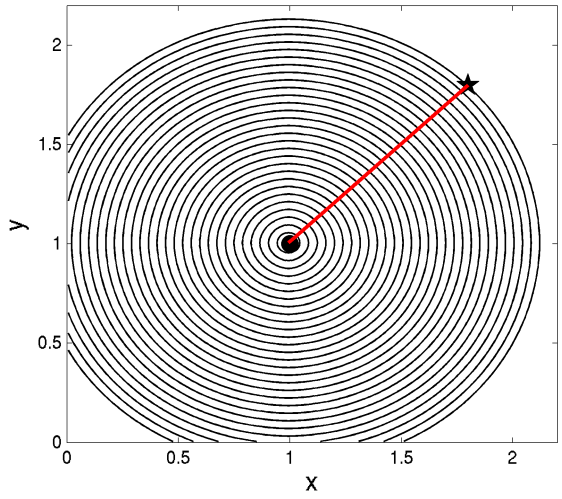


Figure 7-2: Optimal straight line path from start (circle) to end (star) for the example §(7.1.1) computed using our algorithm. The optimal path (shown in red) is overlaid on contours (black circles) of the zero level set at different intermediate times.

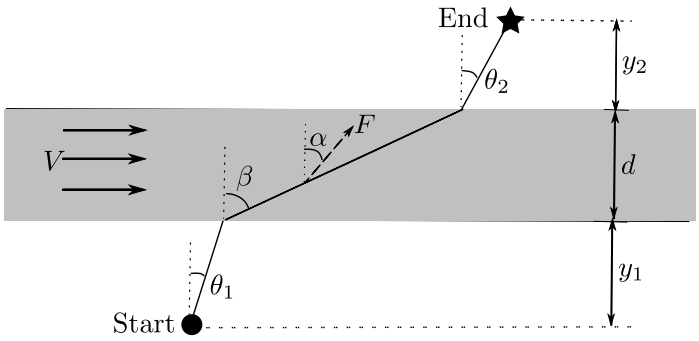


Figure 7-3: Parameters involved in optimal crossing of a jet flow: jet speed V and width d; start (circle), end (star), distances from jet y_1, y_2; vehicle speed in still flow, F and headings $\theta_1, \theta_2, \alpha$; resultant trajectory angle β.

(marked by a filled circle) to the goal (marked by a star). Consistent with our earlier notation, we denote the maximum nominal speed of the vehicle with respect to the flow as F. Let us denote the vehicle heading angles before reaching the flow field, in the flow field, and after exiting the flow region respectively as θ_1, α and θ_2. While the vehicle advances in the jet, it is also advected by the flow. Therefore, in this case, the actual direction of vehicle motion is different from the heading direction. Let us denote the angle formed by the vehicle trajectory in the flow to the vertical by β. Notations for various distance parameters in the problem can be read off from figure (7-3).

As per our algorithm, we propagate a wavefront from the starting position of the vehicle according to the level set equation (5.2) until the zero level set reaches the goal. In figure (7-4), we show the shapes of the zero level set at different times for this example. The level sets are primarily radial expansions outside of the jet and advected to the right in the jet. Therefore, until the zero level set expands from the start point to reach the flow, it remains a circle. After it reaches the horizontal flow, by continuity, this advection elongates the level sets outside of the jet on the downstream side. The kink (shock) in the level sets joining the elongation with the radial expansion below the the jet represents the set of points which can be optimally reached either using the jet or otherwise. As the desired goal is downstream to the jet, the vehicle must make use of this favorable current in order to reach its destination. The forward level set equation (5.2) is solved until the level set front reaches $\mathbf{x_f}$.

Then, we solve equation (5.4) backward in time to calculate the optimal trajectory of the vehicle. The vehicle path computed by our algorithm is shown by discrete points on intermediate level sets in figure (7-5).

Validation

In order to verify the results of the algorithm, we formulate this jet flow problem as a nonlinear optimization problem. The constraints of this optimization problem are obtained as follows.

Let $\mathbf{U}$ denote the velocity of the vehicle in the flow, as seen by a ground observer. We have, componentwise,

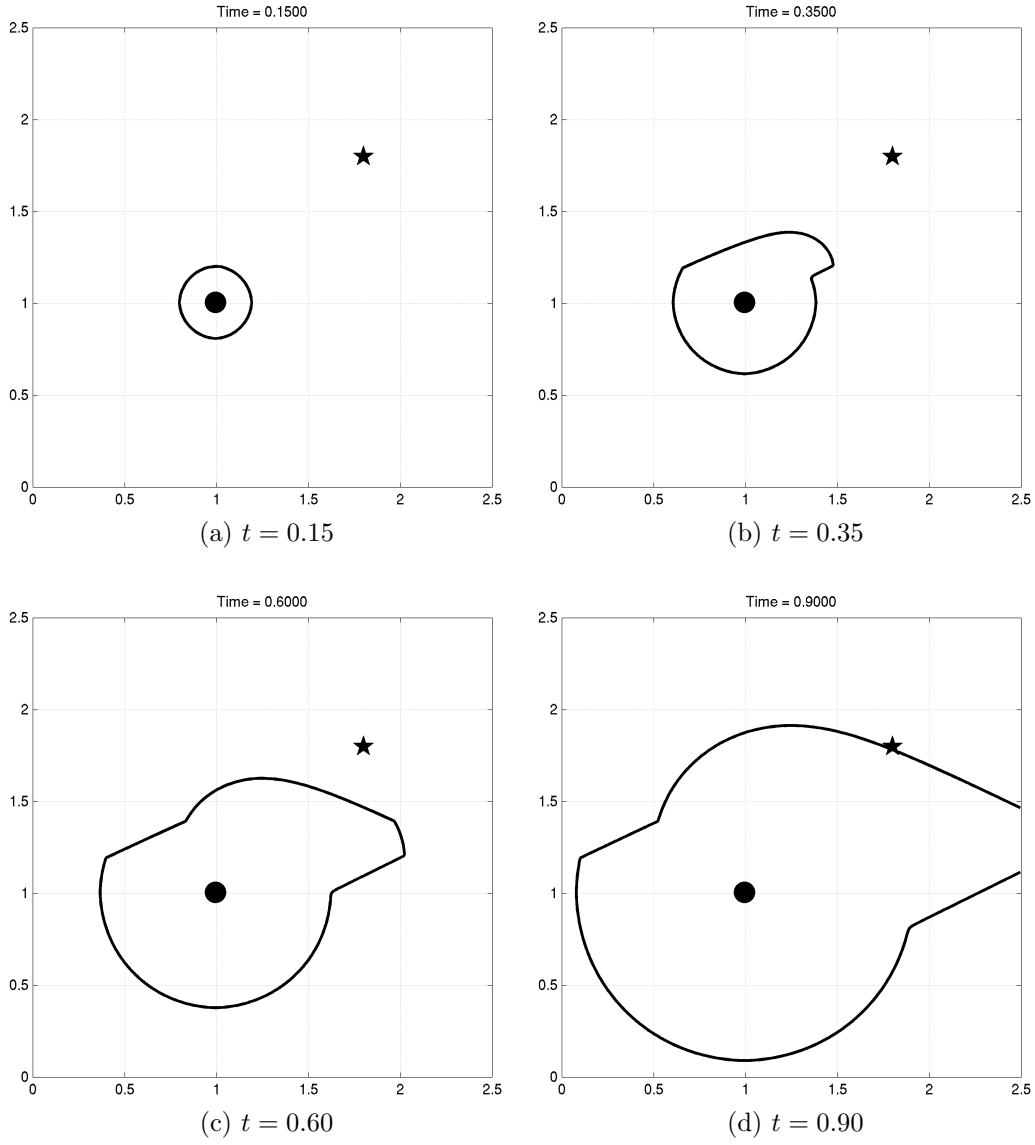
$$U_x = F \sin\alpha + V \tag{7.1}$$

and

$$U_y = F \cos\alpha \tag{7.2}$$

where U_x and U_y are the x and y components of the total vehicle velocity, $\mathbf{U}$. This gives,

$$\tan\beta = \frac{U_x}{U_y} = \tan\alpha + \frac{V}{F}\sec\alpha \tag{7.3}$$



(a) $t = 0.15$ (b) $t = 0.35$

(c) $t = 0.60$ (d) $t = 0.90$

Figure 7-4: Snapshots of the zero level set (reachability front) at different non-dimensional times for jet flow example §(7.1.2). The start point is marked by a black circle, while the end point is denoted by a star.

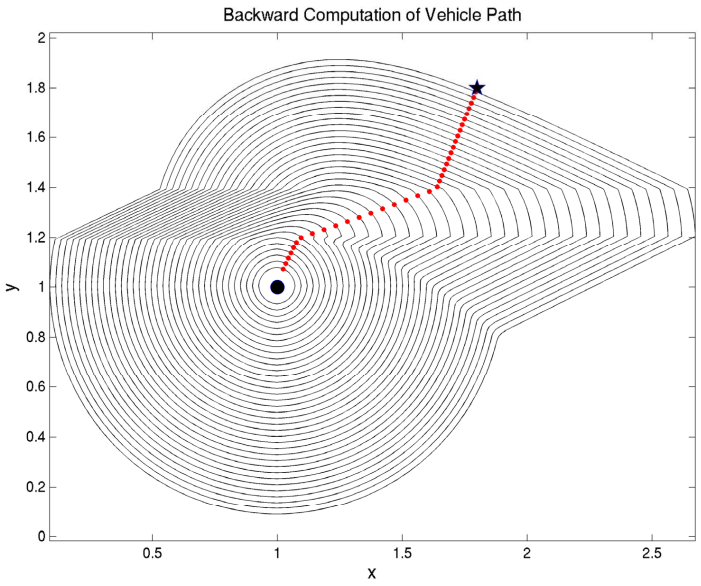


Figure 7-5: Vehicle trajectory predicted by the level set algorithm for the jet flow example in figure (7-3) (§(7.1.2)). The start point is denoted by a circle and the end point by a star. The optimal path (red) is overlaid on level set contours at various intermediate times (black curves).

Let X be the total downstream displacement of the vehicle, i.e. in the x direction. We have, from trigonometry:

$$X = y_1 \tan\theta_1 + d\tan\beta + y_2 \tan\theta_2 \tag{7.4}$$

Finally, the total travel time T can be written as the sum of travel times in each individual region. Hence, the optimization problem we wish to solve is:

$$\min T = \frac{y_1}{F\cos\theta_1} + \frac{d}{F\cos\alpha} + \frac{y_2}{F\cos\theta_2} \tag{7.5}$$

$$\text{s.t. } X = y_1\tan\theta_1 + d\left(\tan\alpha + \frac{V}{F}\sec\alpha\right) + y_2\tan\theta_2 \tag{7.6}$$

and

$$\theta_1, \theta_2, \alpha \geq 0$$

This optimization problem is solved numerically in MATLAB®. We present the results for $F = 1$, $V = 1.2$, $y_1 = 0.2$, $y_2 = 0.4$, $d = 0.2$ and $X = 0.8$ in Table

Table 7.1: Comparison of Results of Level Set Algorithm and Nonlinear Optimization Method

	Level Set Method	Optimization Method
θ_1	22.68°	22.66°
θ_2	22.68°	22.66°
β	70.06°	69.99°
α	45.90°	45.77°
T	0.936	0.937

7.1. From this, we find that the two methods yield the same solution, up to very small differences. These differences in the angles are due to small numerical and truncation errors in the level set computation (e.g. limited grid resolution leads to limited precision in angles).

7.1.3 Rankine Vortex Flow

We have validated our path planning algorithm in the case of zero flow and the case of a simple jet flow. Here, we present the final benchmark example of path planning in a Rankine vortex. This flow field can be characterized in polar coordinates as:

$$\mathbf{V}(r, \theta, t) = v_\theta(r)\hat{\boldsymbol{\theta}} \tag{7.7}$$

Here, $\hat{\boldsymbol{\theta}}$ is the unit vector in the direction of the angular coordinate. $v_\theta(r)$ is a general speed of the flow and depends on the type of vortex. We are interested in computing the fastest time trajectory from the origin, $\mathbf{x_s} = \mathbf{0}$ to the end point $\mathbf{x_f} : (r = R, \theta = 0)$. Let the total travel time of the vehicle be $T(\mathbf{x_f})$ and the fastest travel time be $T_{min}(\mathbf{x_f})$.

Analytical solution for general $v_\theta(r)$

Let the to-be-optimized-speed of the vehicle with respect to the flow be:

$$F_v(t)\hat{\mathbf{h}} = F_r(t)\hat{\mathbf{r}} + F_\theta(t)\hat{\boldsymbol{\theta}}$$

with $\sqrt{F_r(t)^2 + F_\theta(t)^2} \leq F$. The total speed of the vehicle can be written as:

$$\frac{d\mathbf{x}}{dt} = F_r(t)\hat{\mathbf{r}} + [F_\theta(t) + v_\theta(r)]\,\hat{\boldsymbol{\theta}} \tag{7.8}$$

with $\mathbf{x}(t = 0) = \mathbf{0}$ and $\mathbf{x}(t = T(\mathbf{x_f})) = \mathbf{x_f}$. Separating the radial and angular components of the vehicle trajectory, we get;

$$\dot{r} = F_r(t) \quad \text{and} \quad r\dot{\theta} = F_\theta(t) + v_\theta(r)$$

Integrating the first equation, we get

$$R = \int_0^{T(\mathbf{x_f})} F_r(t)dt \;\leq\; \int_0^{T(\mathbf{x_f})} F dt = FT(\mathbf{x_f})$$

Since $F_r(t) \leq F$, we can see that $T(\mathbf{x_f}) \geq \frac{R}{F}$. Hence, R/F is a lower bound for the arrival time, $T(\mathbf{x_f})$. We claim that $T_{min}(\mathbf{x_f}) = R/F$. For this, we need to show that there exists a trajectory that satisfies equation (7.8) and has a travel time of $T(\mathbf{x_f}) = R/F$. Such a trajectory can be generated by inspection, by setting $F_r(t) = F$ and $F_\theta(t) = 0$. For this choice of the vehicle velocity $(F_v(t)\hat{\mathbf{h}})$, we get:

$$\dot{r} = F \tag{7.9a}$$

$$\dot{\theta} = \frac{v_\theta(r)}{r} \tag{7.9b}$$

Integrating the above equations, we get $r(t) = Ft$ and

$$\begin{aligned} d\theta &= \frac{v_\theta(r)}{r}dt = \frac{v_\theta(r)}{r}\frac{dr}{F} \\ \int_{\theta_o}^{\theta} d\theta &= \int_0^R \frac{v_\theta(r)}{Fr}dr \end{aligned}$$

$$\theta = \theta_o + \int_0^R \frac{v_\theta(r)}{Fr}dr \tag{7.10}$$

Here, θ_0 is the initial heading angle of the vehicle. The final value of θ is known from the coordinates of the target point $\mathbf{x_f}$. Therefore, from equation (7.10), θ_0 can

be computed. The control to generate the quickest time trajectory is, therefore,

$$F_v(t)\hat{\mathbf{h}}(t)\Big|_{opt} = F\hat{\mathbf{r}}, \quad \text{with} \quad \theta_0 = \theta - \int_0^R \frac{v_\theta(r)}{Fr}dr \tag{7.11}$$

This solution can also be obtained by using our level set algorithm. The zero level set contours obtained by solving equation (5.2) are all circles centered at the origin because of the angular symmetry of the flow (see figure (7-6)). The only information needed from the forward evolution of the level set to solve the backtracking equation (5.4), is the direction of the normals to the intermediate level set contours. In this problem, we could have guessed the shapes of the contours without solving the forward level set equation (5.2). Since the flow-field has no radial component, all the zero level set contours are circles centered at the origin. The outward normal to the contours is the unit vector in the radial direction, ($\hat{\mathbf{n}} = \hat{\mathbf{r}}$). Using this observation, we can directly solve the backtracking equation to compute the initial heading angle θ_0 (where the normal to the point level set is undefined).

This problem is in fact, almost identical to crossing a uniform jet in the fastest time (discussed in §(7.1.2)). To cross a uniform jet in the fastest time (i.e. reach the opposite bank quickest), one needs to steer perpendicular to the flow at all times, so that the maximum component of the vehicle's speed is directed towards the opposite bank. Similarly, to get away quickest from the center of a vortex flow, one needs to head normal to the streamlines of the flow (i.e. the radial direction). Since the radial direction is undefined at the origin, we have to solve the backtracking equation to compute the initial heading angle (θ_0).

Rankine Vortex Solution

We study the performance of our algorithm when the Rankine vortex flow is defined by,

$$v_\theta(r) = \frac{\Gamma r}{2\pi\sigma^2}$$

Here, Γ represents the total circulation around the origin and σ is the radius of the Rankine vortex. This flow-field models a solid body rotation of the fluid around the

vortex. This is realistic in most practical vortex flows, particularly close to the center. For this example, we use $\Gamma = 20$, $\sigma = 1.5$ and vehicle speed $F = 1$. The coordinates of the end point ($\mathbf{x_f}$) are $R = 1$, $\theta = 0$, i.e. the end point is one unit to the right of the origin. From equation (7.10), the initial heading angle, θ_0 can be calculated.

$$\theta_0 = -\frac{\Gamma R}{2\pi F \sigma^2} = -1.4147 \text{ rad.} \tag{7.12}$$

Hence, the optimal trajectory for this flow-field is:

$$r(t) = Ft, \quad \theta(t) = \theta_0 + \frac{v_\theta(r)}{r}t = \frac{\Gamma(Ft - R)}{2\pi F \sigma^2} \tag{7.13}$$

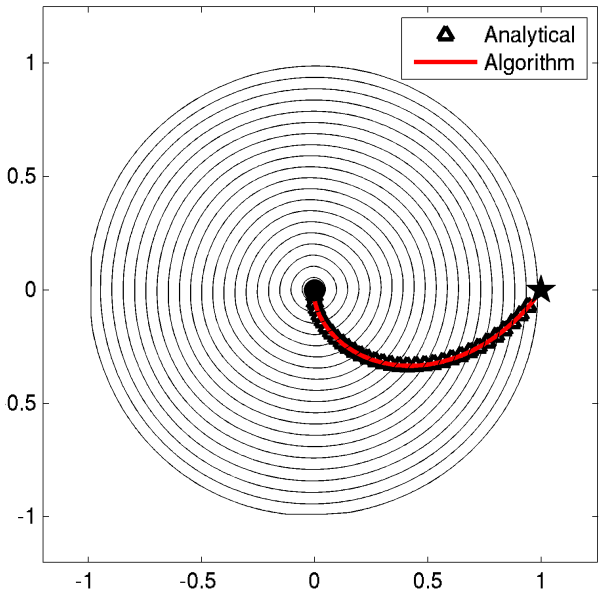


Figure 7-6: Circular level set contours obtained by solving the forward equation (5.2) for a Rankine vortex flow discussed in §(7.1.3). The start point is marked as a black circle at the origin, while the end point is marked as a star. Red: Path given by our algorithm, Black: Optimal path calculated analytically (equation (7.13)). The paths given by the two approaches are identical.

The shapes of the zero level set contours at different times and the optimal vehicle trajectory obtained by solving equation (5.4) are plotted in figure (7-6). The vehicle headings predicted by the level set algorithm are compared against the analytically computed headings in figure (7-7). From these figures, it can be seen that our level

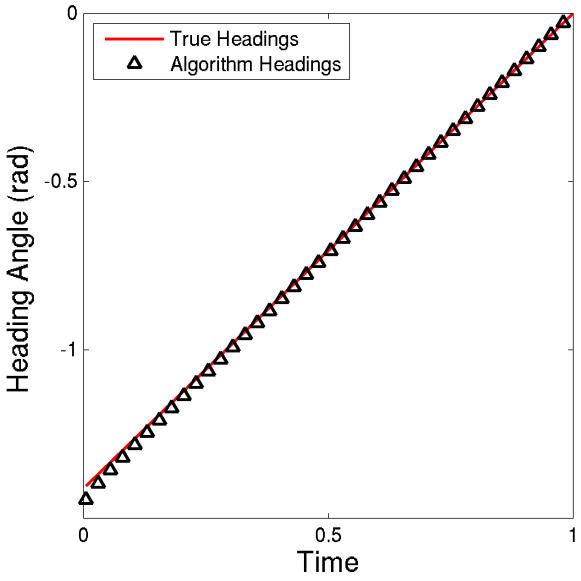


Figure 7-7: Variation of vehicle heading angles (in radians) with time - Black: Heading angles predicted by our level set algorithm, and Red: Heading angles calculated analytically (equation (7.13)).

set algorithm accurately predicts the vehicle headings and the optimal trajectory. Through this example, we emphasize that the only information needed from the solution of the forward level set equation is the time evolution of the zero level set front. If there is an intuitive way of knowing the level set shapes a priori, the forward model does not have to be solved. This is generally not the case and therefore, for most flows, the forward level set evolution equation must be solved.

7.2 Realistic Ocean Flow Examples

In this section, we apply our path planning methodology to more complex and realistic, however numerically simulated ocean flow fields. We use these examples to illustrate certain unique features and capabilities of the algorithm.

7.2.1 Path Planning in a Double Gyre Flow

The wind-driven double gyre flow that we study is modeled using a barotropic single layer-model in a square basin of size $L = 1$ described in detail in (Dijkstra and

Katsman, 1997, Simmonet et al., 2009) (see also Pedlosky (1998), Cushman-Roisin and Beckers (2010)). The intent is to simulate the idealized near-surface double-gyre ocean circulation at mid-latitudes. The mid-latitude easterlies and trade winds in the northern hemisphere drive a cyclonic gyre and an anticyclonic gyre, and the corresponding zonal jet in between. This eastward jet would correspond to the Gulf Stream in the Atlantic and to the Kuroshio and its extension in the Pacific. This idealized flow is modeled by the nondimensional equations of motion

$$\frac{\partial u}{\partial t} = -\frac{\partial p}{\partial x} + \frac{1}{\mathrm{Re}}\Delta u - \frac{\partial\left(u^2\right)}{\partial x} - \frac{\partial\left(uv\right)}{\partial y} + fv + a\tau_x, \tag{7.14a}$$

$$\frac{\partial v}{\partial t} = -\frac{\partial p}{\partial y} + \frac{1}{\mathrm{Re}}\Delta v - \frac{\partial\left(vu\right)}{\partial x} - \frac{\partial\left(v^2\right)}{\partial y} - fu + a\tau_y, \tag{7.14b}$$

$$0 = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}, \tag{7.14c}$$

where Re is the flow Reynolds number taking values from 10 to 10^4, $f = \tilde{f} + \beta y$ is the non-dimensional Coriolis coefficient, and $a = 10^3$ the strength of the wind stress. In non-dimensional terms, we have $\tilde{f} = 0.1$, $\beta = 10^3$. The flow in the basin is forced by an idealized zonal wind stress that is constant in time, given by

$$\tau_x = -\frac{1}{2\pi}\cos 2\pi y$$
$$\tau_y = 0.$$

Free slip boundary conditions are imposed on the northern and southern walls ($y = 0, 1$) and no-slip boundary conditions on the eastern and western walls ($x = 0, 1$). In what follows, we present results for Re = 150, for which we have resolved flow simulations. The dimensionless inertial and viscous boundary layer thickness are $\delta_l = L^{-1}\sqrt{U/\beta_0} = \sqrt{1/\beta}$ and $\delta_M = L^{-1}(A_H/\beta_0)^{1/3} = (1/(\mathrm{Re}\,\beta))^{1/3}$ respectively.

The governing equations (7.14) for the fluid flow are solved using a second order accurate Navier-Stokes solver, which is a component of a modular finite volume framework, implemented in MATLAB® (Ueckermann and Lermusiaux, 2009). This framework uses a uniform, two-dimensional staggered C-grid for the spatial discretiza-

tion. The diffusion operator in the Navier-Stokes equations is discretized using a second order accurate central difference scheme. The advection operator is discretized using a Total Variation Diminishing (TVD) scheme with the monotonized central (MC) limiter Van Leer (1977). The time discretization uses a first-order accurate, semi-implicit projection method, where the diffusion and pressure terms are treated implicitly, and the advection is treated explicitly (for details see Ueckermann et al. (2012), Ueckermann and Lermusiaux (2011)). In figure (7-8), we show a few snapshots of the computed flow-field streamlines, overlaid on a color plot of vorticity, at different non-dimensional times.

Using this example, we wish to:

1. Examine the performance of our algorithm for path planning in a strong and dynamic flow-field.

2. Illustrate an application where the goal is to determine if a vehicle can reach a certain end point within a specified time limit.

Table 7.2: **Double Gyre Flow**: Numerical parameters used in generation of the flow-field (equation (7.14)) and in level set evolution (equation (5.2)).

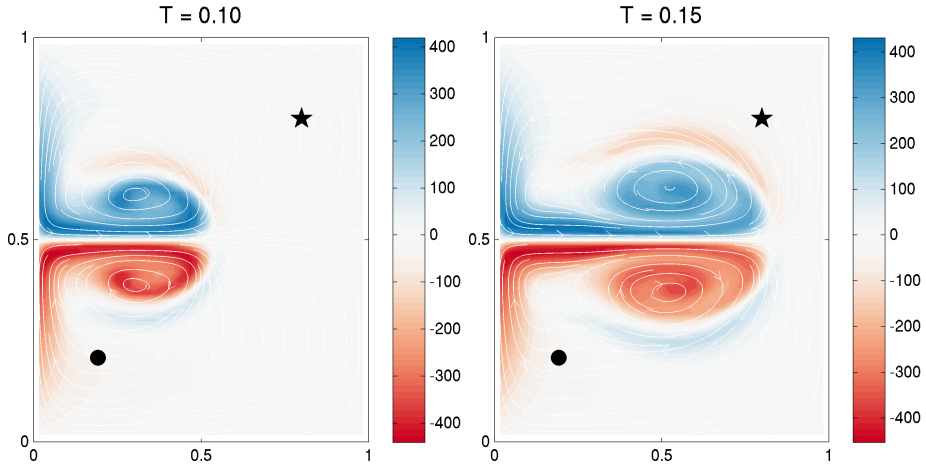
	Parameter	**Value**
Flow Field	Domain Size ($x \times y$)	1×1
	Grid Size ($Nx \times Ny$)	64×64
	Time Step (Δt)	1.0×10^{-4}
	Reynolds Number	150
Level Set	Domain Size ($x \times y$)	1×1
	Grid Size ($Nx \times Ny$)	64×64
	Time Step (Δt)	1×10^{-4}
	Time Offset	1.1
	Vehicle Speed (F)	5
	Initial Conditions	Signed Distance

We are interested in computing the fastest time trajectory from $\mathbf{x_s} = (0.2, 0.2)$ to $\mathbf{x_f} = (0.8, 0.8)$. In order to see more dynamic features in the flow-field, we let the vehicle begin its motion at an offset time, t_s, after the flow-field, i.e. the flow field experienced by the vehicle at the beginning of its motion is the flow field at time

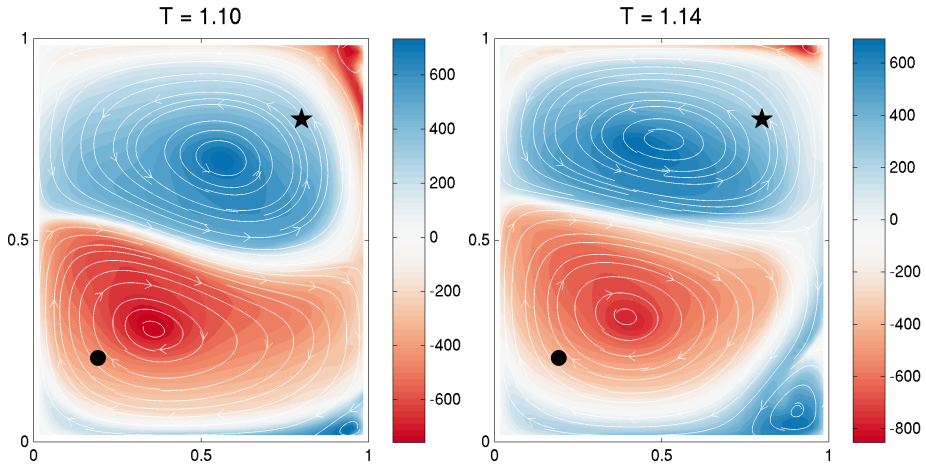
$t = t_s$. In this example, we set $t_s = 1.10$. Various numerical parameters used to solve equation (7.14) and the level set equation (5.2) are shown in Table (7.2). Equation (5.2) is solved according to the numerical scheme described in §(6.2). The level set field is initially chosen to be the signed distance function.

In figure (7-9), we show the time evolution of the zero level set front when the maximum vehicle speed relative to the flow is, $F = 5$. The backtracking equation (5.4) is solved to compute the optimal path. The optimal vehicle trajectory for this example is plotted in figure (7-10). Due to the strong flow-field, the vehicle has to perform two revolutions around the lower eddy before it finds a favorable current that drives it towards the end point.

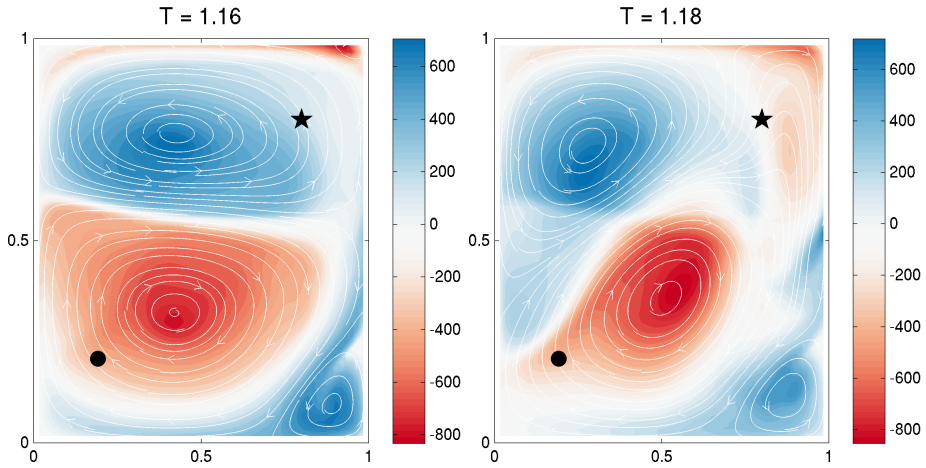
Another important question that arises in path planning is to determine whether a vehicle can reach a given end point within a specified time limit. We study this problem using the double gyre flow-field. For this example, we set $t_s = 0.4$, while all other parameters remain the same as above. If the maximum relative vehicle speed, F is set to 8, the optimal travel time from $(0.2, 0.2)$ to $(0.8, 0.8)$ is computed to be 0.0343 (see figure (7-11)). However, if the relative vehicle speed F is reduced to 6, then the optimal travel time increases to 0.0856, which is more than twice the earlier value. The optimal trajectory in this case is also significantly different. Our level set algorithm can predict if a vehicle can reach the target within a specified time limit. The reachability front at time $t = 0.035$ for $F = 6$ is shown in figure (7-11)c. Since the front has not reached the end point by this time, we can conclude that it is not possible for the vehicle to reach the target within the specified time of 0.035 units. In the general case, the forward evolution equation (5.2) needs to be solved until the zero level set reaches the target, or until the specified time limit, whichever is smaller. If the target is reached within the imposed time limit, the optimal trajectory can be computed by backtracking, and if not, the algorithm terminates by outputting the reachability set (see also §(5.3)).



(a) $t = 0.10$ (b) $t = 0.15$

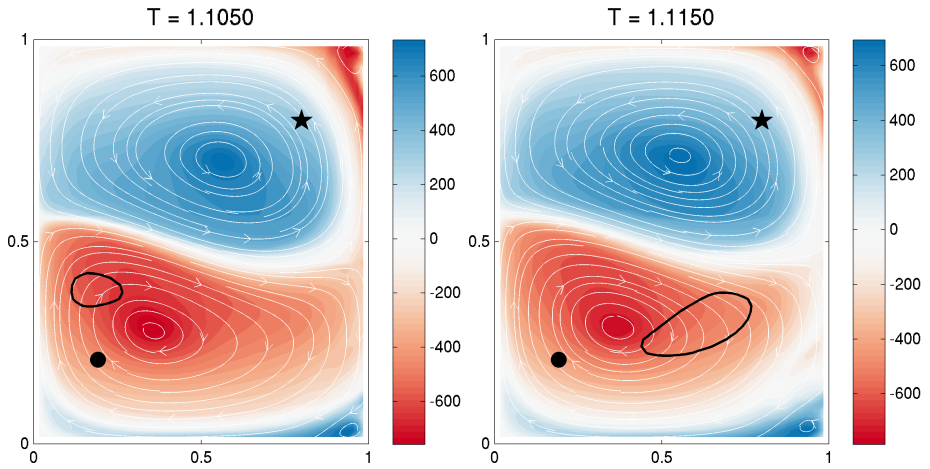


(c) $t = 1.10$ (d) $t = 1.14$



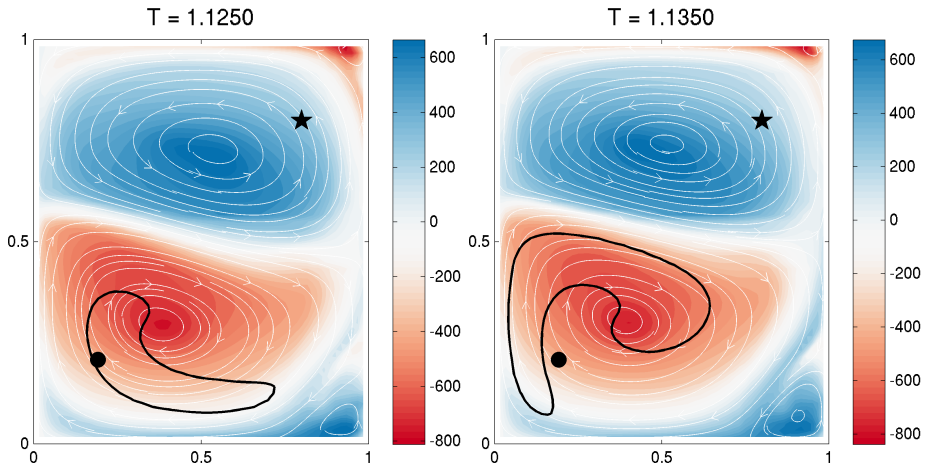
(e) $t = 1.16$ (f) $t = 1.18$

Figure 7-8: Snapshots of double gyre flow-field at different non-dimensional times - streamlines of the flow (white) are overlaid on color plots of vorticity of the flow. The start point ($\mathbf{x_s}$) and the end point ($\mathbf{x_f}$) are marked as a circle and a star respectively.



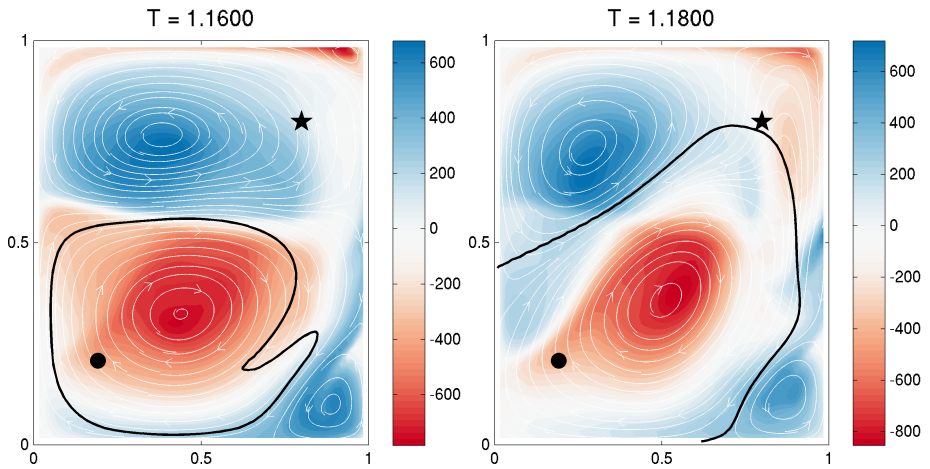
(a) $t = 1.105$

(b) $t = 1.1150$



(c) $t = 1.1250$

(d) $t = 1.1350$



(e) $t = 1.1600$

(f) $t = 1.1800$

Figure 7-9: Time evolution of the zero level set (reachability front) for the double gyre flow field (discussed in §(7.2.1)) for offset time $t_s = 1.10$.

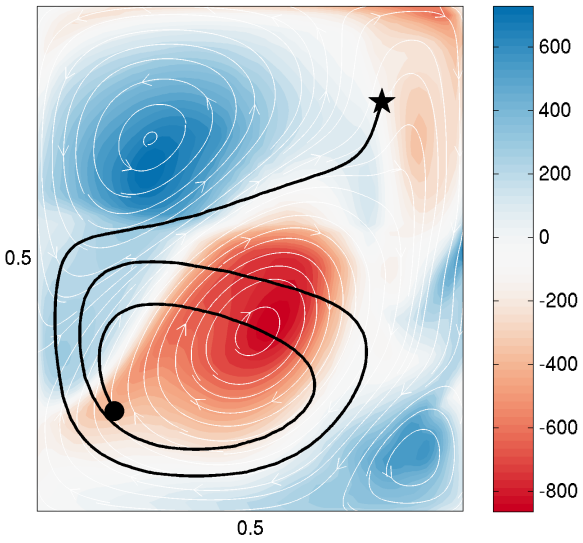
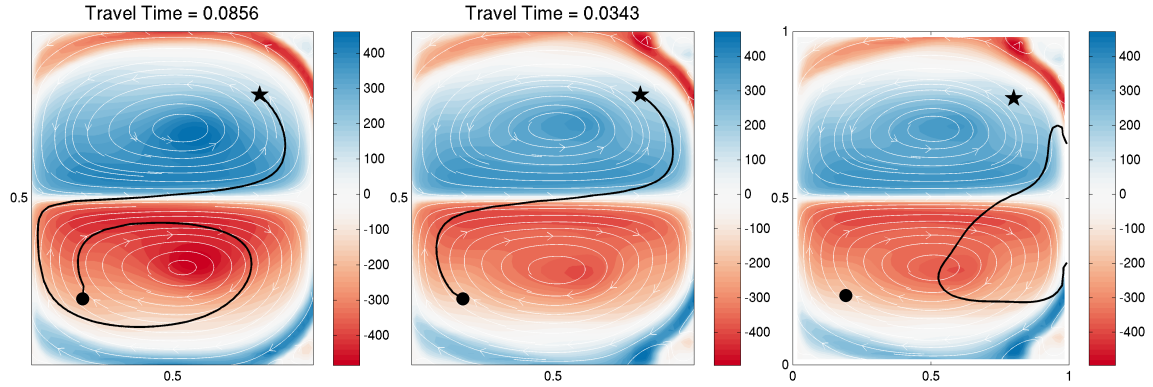


Figure 7-10: Fastest path from $\mathbf{x_s} = (0.2, 0.2)$ (circle) to $\mathbf{x_f} = (0.8, 0.8)$ (star) in double gyre flow-field (§(7.2.1)). The optimal path (black) is overlaid on the final snapshot of the flow-field, colored by vorticity.



(a) $F = 6$ (b) $F = 8$ (c) $F = 6$

Figure 7-11: Fastest time paths for two vehicles in the double gyre flow field overlaid on vorticity-colored plots of the final flow-field. (a) The first vehicle ($F = 6$) takes 0.0856 units of time to reach the end point whereas (b) the second vehicle ($F = 8$) takes only 0.0343 units of time. (c) The reachability front at time $t = 0.035$ for the slower vehicle ($F = 6$).

7.2.2 Flow Past Cylinder/Circular Island

In this example, we present the results of our path planning algorithm in the case of open flow in a frictionless conduit with a circular cylinder obstacle (see figure (7-12)). This is a highly unsteady flow field that exhibits periodic vortex shedding in the

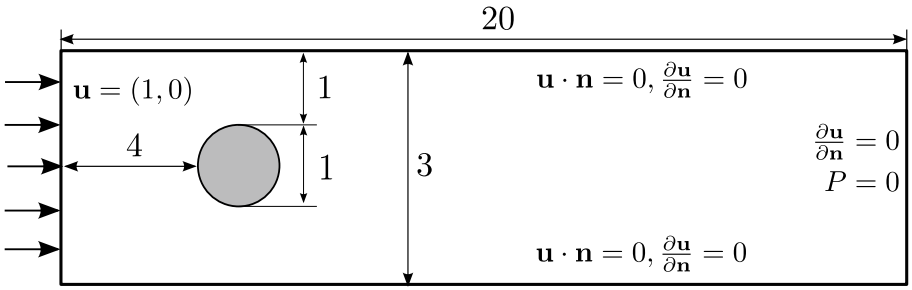


Figure 7-12: Schematic of Flow Past Circular Cylinder Test Case

wake of the cylinder, at appropriate Reynolds numbers. For a real ocean scenario, this corresponds to the flow that results when a stream of ocean current flows past a circular island. The purpose of this example is threefold:

1. To illustrate the performance of our algorithm for path planning in a strong and dynamic flow-field.

2. To demonstrate how obstacles to the flow (and the vehicle) are naturally handled by the algorithm.

3. To illustrate that the algorithm can be parallelized when paths for multiple vehicles have to be planned.

The flow is driven by a deterministic uniform inlet boundary condition (left of domain), with slip velocity boundary conditions at the top and bottom, and open boundary conditions at the outlet (see figure (7-12)). Snapshots of the flow field at different times are shown in figure (7-13). The Reynolds number of the flow used in this example is 1000.

In this example, there are 11 start points (marked by black circles) upstream of the cylinder and 11 end points (represented by colored markers) downstream of the cylinder (see figure (7-13)). Each start point releases a *swarm* of 11 vehicles, one for each different end point. Thus, there are a total of 121 vehicles for which paths have to be planned. The goal for this swarm of vehicles is to reach their respective targets in the fastest time, by utilizing (or avoiding) the multi-scale flow structures in their path. In addition, none of the vehicles should collide with the cylindrical obstacle in the domain, i.e. the paths of all the vehicles should be both, safe and optimal.

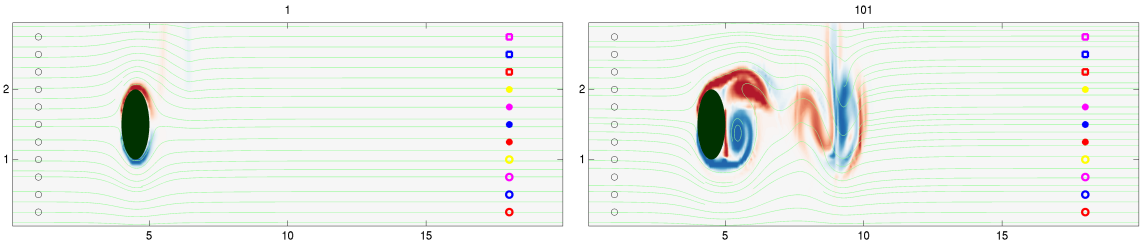
Table 7.3: **Flow past a circular island:** Numerical parameters used in generation of the flow field and in level set evolution (equation (5.2)).

	Parameter	**Value**
Flow Field	Domain Size ($x \times y$)	20×3
	Grid Size ($Nx \times Ny$)	200×30
	Time Step (Δt)	5.0×10^{-4}
	Reynolds Number	1000
Level Set	Domain Size ($x \times y$)	20×3
	Grid Size ($Nx \times Ny$)	200×30
	Time Step (Δt)	5.0×10^{-4}
	Time Offset	0.4
	Vehicle Speed (F)	0.5
	Initial Conditions	Signed Distance

Corresponding to each of the 11 start points, we evolve a level set, according to equation (5.2). The numerical parameters used in generation of the flow field and in solving the forward evolution equation are presented in Table (7.3). The obstacle in the domain is handled by 'masking' out the appropriate region in the domain mesh, i.e. the governing equations are numerically solved at all points outside of the obstacle. For grid points that lie under the obstacle mask, a deterministic boundary condition is imposed on the level set function, ϕ. At all domain boundaries, an open boundary condition is imposed on ϕ.

Figure (7-14) shows the time evolution of level set fronts corresponding to two different start points, overlaid on streamlines of the flow-field, colored by vorticity. We can see that the level set fronts, for all these cases, do not penetrate the obstacle, but 'wrap' around it. This feature of the level set method leads to collision free (safe) trajectories. The level set fronts from each start point are evolved until *every* end point has been crossed. The crossing times of each end point are recorded because backtracking (equation (5.4)) has to be performed from the time each end point is reached. The optimal vehicle trajectories corresponding to each start point, to all the end points are plotted in figure (7-15). As expected, none of the paths pass through the obstacle.

Through this example, we have illustrated the ability of the algorithm to generate collision free vehicle trajectories in addition to predicting time optimal paths. This



(a) $t = 0$ (Initial Time)

(b) $t = 3.00$

(c) $t = 4.800$

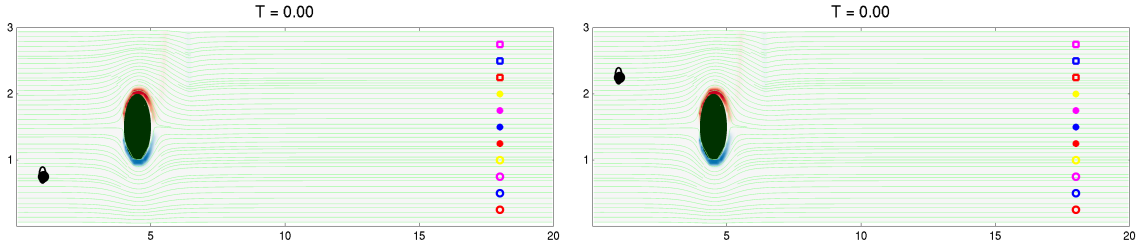
(d) $t = 6.900$ (Final Time)

Figure 7-13: Snapshots of velocity-field for flow behind a circular island (discussed in §(7.2.2)) at different non-dimensional times. The streamlines of the flow are overlaid on color plots of the vorticity.

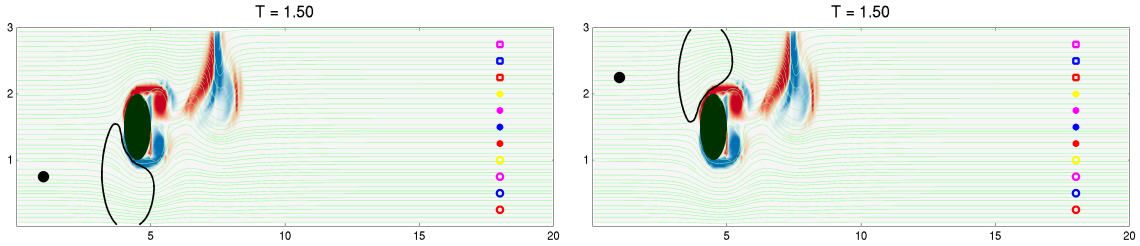
comes at no additional computational expense. Also, the number of level sets that need to be evolved depends on the number of different start points, and *not* on the number of end points. Paths to every end point corresponding to a single start point can be planned by evolving just one level set field. In the case of multiple end points, the level set needs to be evolved until all of the end points have been reached. Thus, this algorithm can efficiently be parallelized to independently compute optimal vehicle tracks from multiple start points.

7.2.3 Sudden Expansion in Coastal Ocean and Fluid Flows

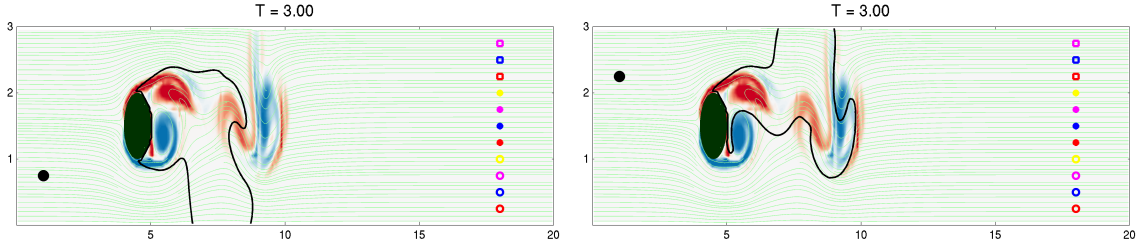
We now apply our path planning algorithm to the third and final realistic ocean flow-field with dynamic jets and eddies, and discuss the results. This example has been published earlier in (Lolla et al., 2012). In this example, we consider a uniform barotropic jet (2D flow in the horizontal plane) exiting a strait or estuary. Such flows commonly occur in the coastal ocean and generally lead to meanders and vortices as the jet exits the constriction. This situation corresponds to a highly unsteady flow



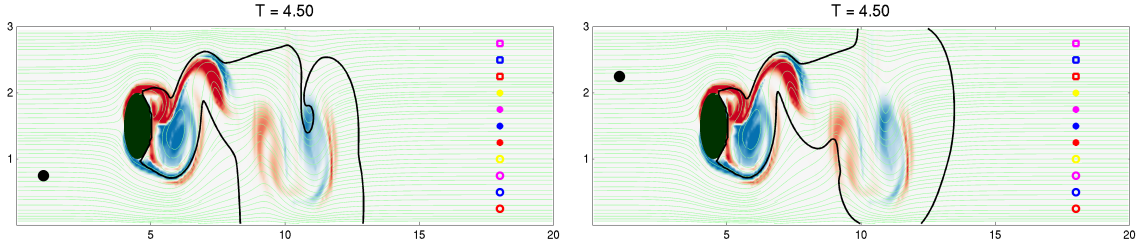
(a) $t = 0$ (Initial Time) (b) $t = 0$ (Initial Time)



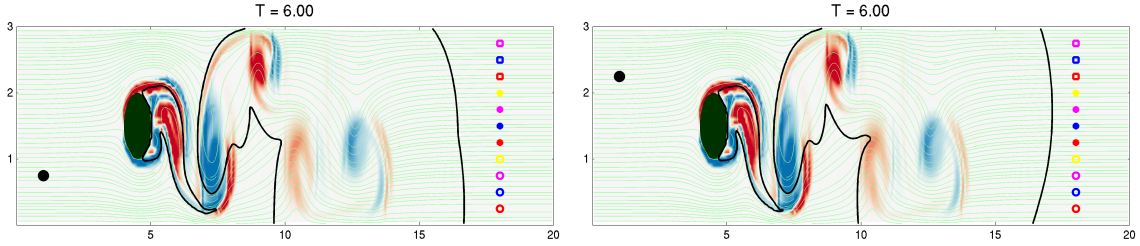
(c) $t = 1.5$ (d) $t = 1.5$



(e) $t = 3.00$ (f) $t = 3.0$

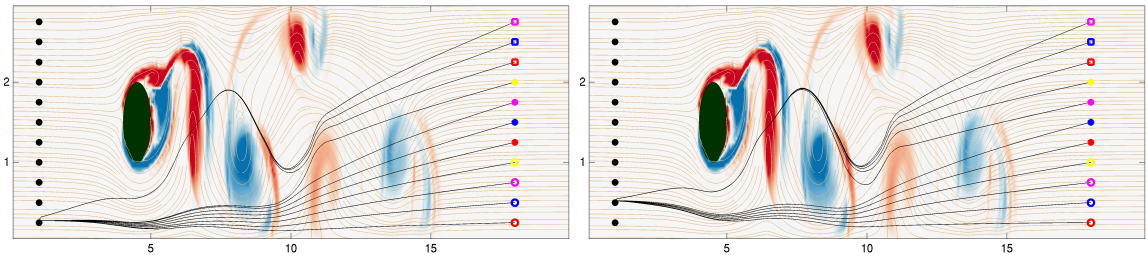


(g) $t = 4.5$ (h) $t = 4.5$



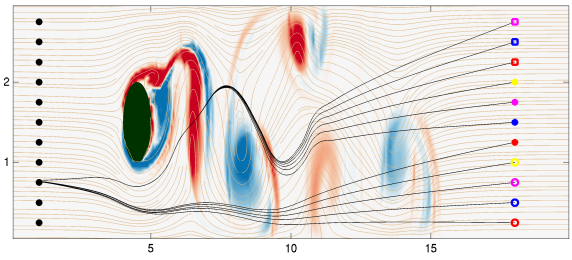
(i) $t = 6.0$ (j) $t = 6.0$

Figure 7-14: Flow past circular island (§(7.2.2)) - time evolution of the zero level set front corresponding to two different start points (marked in black). None of the level set fronts pass through the island, but instead 'wrap' around the island.

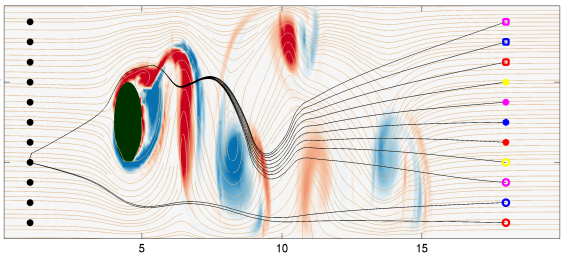


(a) Start point 1

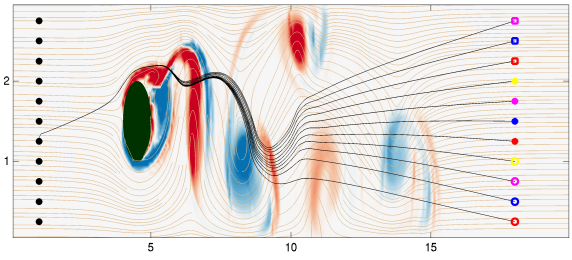
(b) Start point 2



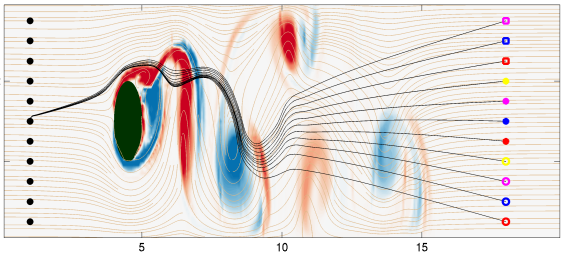
(c) Start point 3



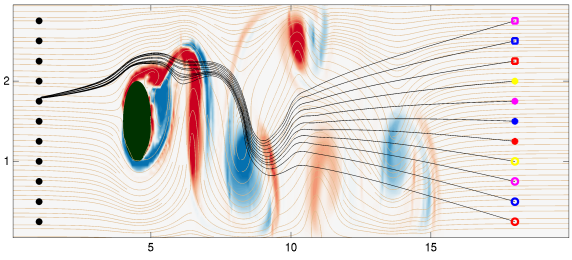
(d) Start point 4



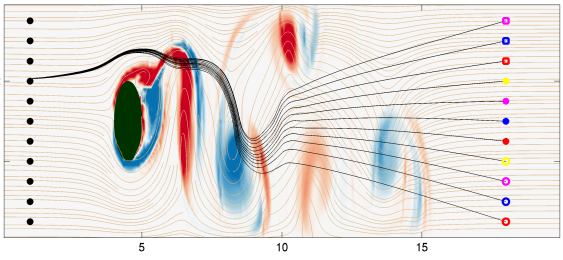
(e) Start point 5



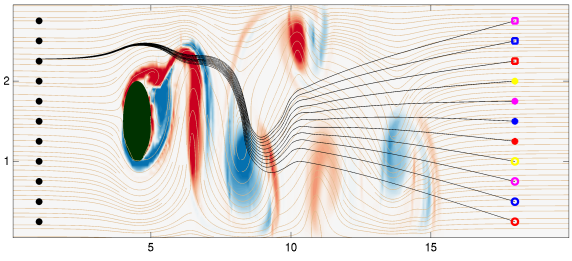
(f) Start point 6



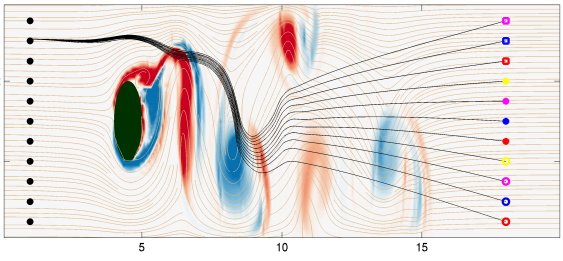
(g) Start point 7



(h) Start point 8



(i) Start point 9



(j) Start point 10

Figure 7-15: Flow past circular island (§(7.2.2)) - Safe and time optimal trajectories corresponding to every start point. As expected, none of the paths pass through the island. All vehicle paths are overlaid on a snapshot of the flow-field at the final time.

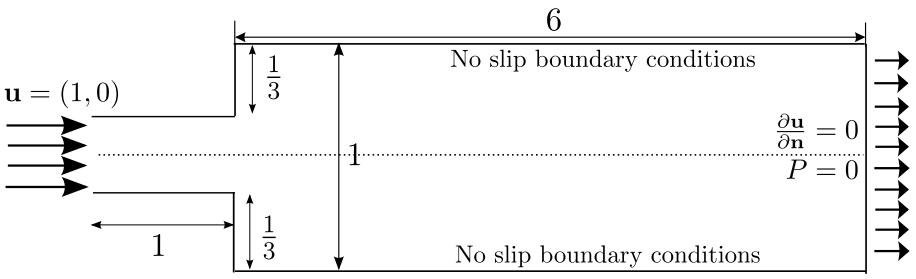


Figure 7-16: Schematic of Sudden Expansion Test Case

field. If the width of the constriction is small enough, effects of the earth's rotation (Coriolis acceleration) can be neglected.

We refer the reader to figure (7-16) for a schematic of the sudden expansion example. The flow is driven by a deterministic inlet velocity with no slip boundary conditions at the top and bottom walls. The numerical parameters used in generation of the flow-field are given in Table (7.4). The reader is referred to figure (7-17) for two snapshots of the flow-field at two successive non-dimensional times. Shown are flow streamlines overlaid on the magnitude (in color) of the velocity field.

Table 7.4: **Sudden expansion in Coastal Ocean:** Numerical parameters used in generation of the flow field and in level set evolution (equation (5.2)).

	Parameter	**Value**
Flow Field	Domain Size ($x \times y$)	7×1
	Grid Size ($Nx \times Ny$)	420×60
	Time Step (Δt)	2.0×10^{-3}
	Reynolds Number	417
Level Set	Domain Size ($x \times y$)	7×1
	Grid Size ($Nx \times Ny$)	420×60
	Time Step (Δt)	2.0×10^{-3}
	Time Offset	0
	Vehicle Speed (F)	0.5
	Initial Conditions	Signed Distance

In what follows, we will consider the scenario of a swarm of underwater vehicles that are released from a fixed point near the exit of the strait or estuary (this start point could correspond to a harbor or larger platform such as a ship or oil rig). The goal for the swarm is to reach a predetermined formation in the open ocean in fastest time, optimally using (or avoiding) the multi-scale flow structures as they

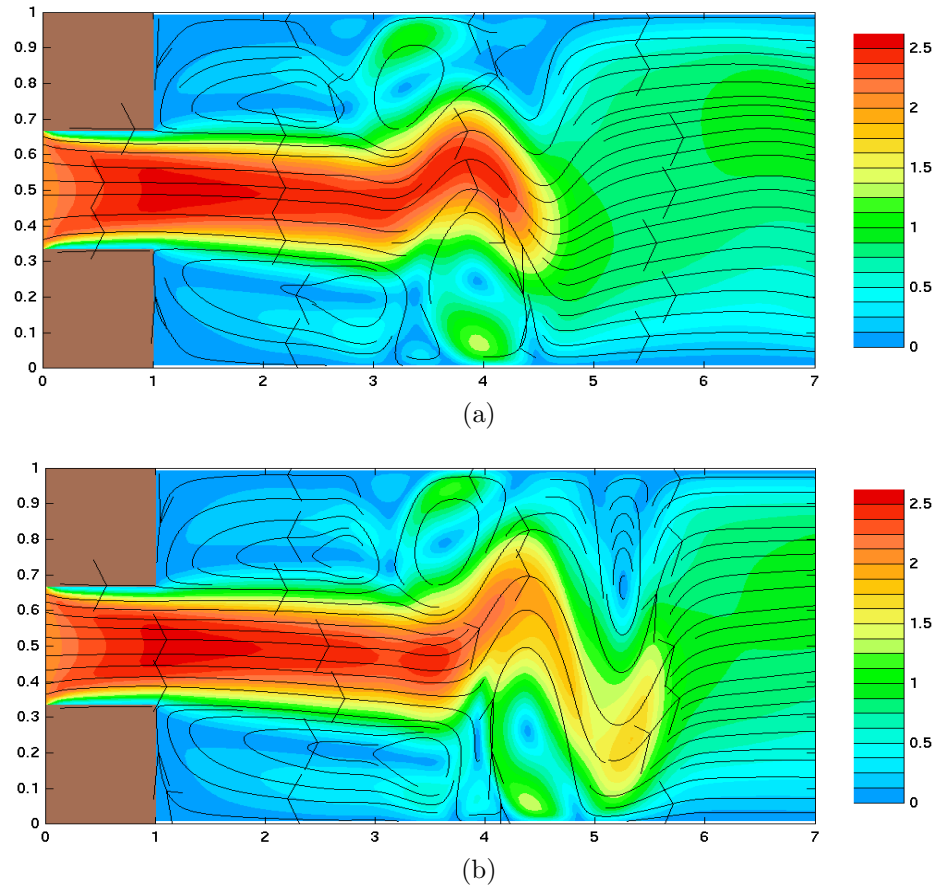
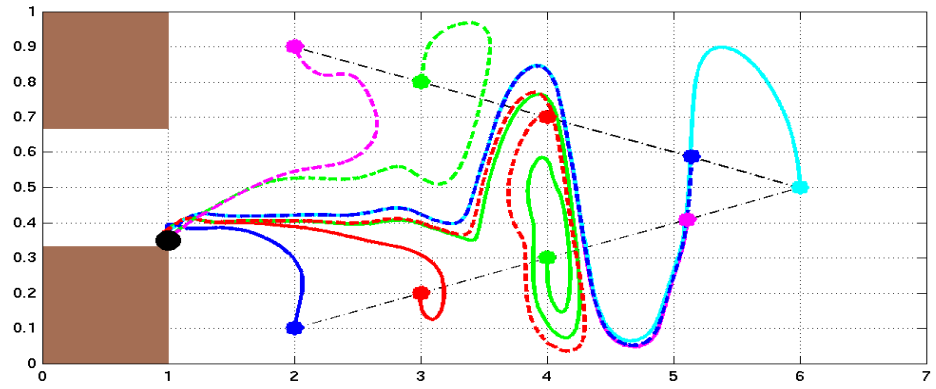


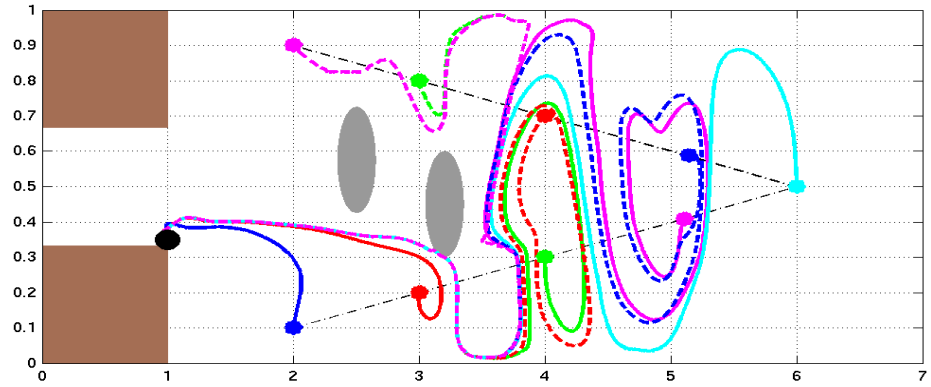
Figure 7-17: Snapshots of the flow field for a jet exiting a strait or estuary (sudden expansion/2D coastal flow) showing color maps of the total magnitude of the flow velocity overlaid with streamlines (a) at the time of initial vehicle deployment and (b) near the final time of vehicle maneuvers in Fig. 7-18a.

occur along the way. The formation can for example be selected based on security, surveillance, pollution monitoring or ocean sampling considerations. In all cases, our methodology will compute the optimal heading time-series for each vehicle based on our predicted time-dependent flow field. The computational cost of the algorithm is overall proportional to the geometric dimensions of the formation pattern.

Another setting where this example can be useful is for the monitoring of the flow in a pipe or channel which encounters a sudden increase in cross sectional area. In that situation, our example would illustrate how mobile sensors released at the junction would have to be navigated to reach a specific formation in fastest time. Such a formation could then be designed to monitor possible pressure drops, release of toxic material, status of pipe wall conditions or other properties.



(a)



(b)

Figure 7-18: Optimal vehicle paths for 9 vehicles deployed from a single point (black dot) in the flow illustrated by Fig. 7-17. Results for two situations are shown: (a) No constraints or forbidden regions: Vehicle paths then take full advantage of evolving jets and eddies to reach their final positions (colored dots) in shortest time. (b) Two forbidden regions: Vehicles are denied access to the gray shaded regions. Our algorithm provides seven new time optimal paths for the paths computed in (a) that are blocked while it correctly leaves unchanged the two paths that are not blocked.

In this example, we set the speed of the vehicles in still flow to $F = 0.5$. The maximum speed $V_{\max}$ of the flow is 2.5 (see Fig. 7-17). The width of the inlet is one third of the total width of the channel (see figure (7-16)). The Reynolds number is

$$\mathrm{Re} = \left(\frac{h}{2}\right)\frac{U_{\max}}{\nu} = 417 \quad \text{with} \quad h = \frac{1}{3} \quad \text{and} \quad \nu = 10^{-3}.$$

In our scenario, we enforce that the swarm of vehicles take a triangle shape at final time, as shown in Fig. 7-18a. The vehicles are released at the lower edge of the inlet. Fig. 7-18a shows the optimal paths of the vehicles computed using our path planning

algorithm. From Fig. 7-17b, we can intuitively see that to reach the tip of the triangle in shortest time, the vehicle must ride along a favorable current. For the four end-points that are closest to the inlet, the vehicles clearly utilize the upper and lower re-circulation eddies. Overall, we find from Fig. 7-18a that the algorithm correctly predicts the shapes of the optimal paths.

7.2.4 Ocean flows with 'forbidden' regions

We now consider the situation where the swarm of vehicles cannot enter specific regions, either because of safety, hazardous conditions, security or naval considerations. We refer to these regions as *forbidden* regions because they cannot be entered by vehicles but they have *no effect on flow fields*, i.e. currents are not affected by them.

To implement the forbidden regions in the forward level set evolution, we replace the right-hand sides of equations (6.6-6.8) with zero for grid points inside the forbidden regions. In the backward calculation, we only need to mask $\mathbf{V}(\mathbf{x}, t)$ in equation (6.10) with zeros in the forbidden regions since the level sets have evolved from the modified forward algorithm have normals that correctly go around the forbidden regions. Handling such regions is thus straightforward, which is a major advantage of our approach.

Path planning with forbidden regions is illustrated in figure (7-18b). The physical setup is as in figure (7-18a), but we prevent the paths from entering the two regions shown in gray. Collectively, these two regions block seven of the nine optimal paths of figure (7-18a). The new optimal paths for these seven vehicles all ride the lower edge of the main jet, just skirting the bottom of the second forbidden region. They then ride down one eddy and up an adjoining eddy, figure (7-17a), to rejoin the main jet behind the forbidden regions. The two paths from figure (7-18a) that did not pass through the forbidden areas remain unaffected by the forbidden areas.

We now present several interesting examples to bring out more subtle points about the algorithm and its give an idea of the wide range of its applicability.

7.3 Corollaries

In this section, we present examples that also serve as corollaries to the theorem discussed in §(5.2). Every example to be discussed here has been designed to bring across an important feature of our methodology.

7.3.1 Discontinuity in Arrival Time

Consider a one dimensional problem in which we want to navigate a vehicle from $x = 0$ to $x = 4$ in the fastest time. Using the notation in theorem §(5.2), we let the maximum vehicle speed in still flow be $F = 1$ and let the external velocity field be $\mathbf{V}(x,t) = -2\sin(\pi t)\hat{\mathbf{i}}$ (see figure (7-19)). This is an oscillating velocity field, which can exist in compressible fluid flows. We choose this oscillating flow-field because it conveys very well, the point we wish to focus on.

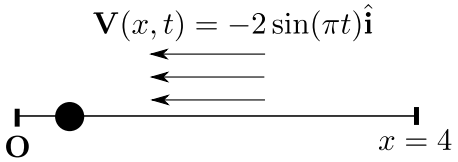


Figure 7-19: 1D Flow Field and Domain

Since the vehicle motion is restricted only to the x axis, the vehicle has only two heading choices at any time- it can either head to the right or the left. As per the theorem in §(5.2), the reachability front is traced when the vehicle moves at maximum speed relative to the flow. Thus, the reachability front in this case consists of only two points. The first point is obtained when the vehicle always heads to the right at speed F relative to the flow, while the other is obtained when it always heads to the left. Since the end point is to the right or the start point and the flow-field is uniform, we can argue that the optimal trajectory is realized when the vehicle always moves with velocity $F\hat{\mathbf{i}}$ relative to the flow. Therefore, the time optimal trajectory of

this vehicle governed by:

$$\frac{dx}{dt} = F + \mathbf{V}(x,t) \cdot \hat{\mathbf{i}} \tag{7.15}$$

$$= 1 - 2\sin(\pi t) \tag{7.16}$$

Integrating this equation with initial condition $x(0) = 0$, we get:

$$x(t) = t + \frac{2}{\pi}\left(\cos(\pi t) - 1\right) \tag{7.17}$$

This continuous path is plotted in blue in figure (7-20). From this trajectory, we can obtain the first arrival time field ($T_1(x)$) for this 1-D problem. Mathematically,

$$T_1(y) = \min_t \{x(t) = y\}$$

The (optimal) first arrival time field $T_1(x)$ is plotted in red, in the figure (7-20). From this plot, we can clearly see the discontinuity in $T_1(x)$ at points $x = 0.08$ and $x = 2.08$. This happens because at certain times, the vehicle experiences a strong flow current adverse to its rightward motion. The vehicle is forced to reverse its trajectory until a favorable (rightward) current advects it again towards the end point. Due to this strong adverse current, the vehicle visits some points in its optimal path *more than once*. This allows us to define the *second* arrival time field, $T_2(x)$ as,

$$T_2(y) = \min_t \{x(t) = y \;\; \& \;\; t > T_1(y)\}$$

Thus, in this example, the optimal path to $x = 0.08^+$ ends up visiting the point $x = 0.08$ twice. Mathematically,

$$\lim_{x \to 0.08-} T_1(x) \neq \lim_{x \to 0.08^+} T_1(x) = \lim_{x \to 0.08^-} T_2(x)$$

Therefore, the first arrival time field $T_1(x)$ is not necessarily continuous at all points in the domain. Due to this, the gradient of $T_1(x)$ at these points is unbounded. Under these circumstances, equation (5.18) is invalid. We need to keep track of later

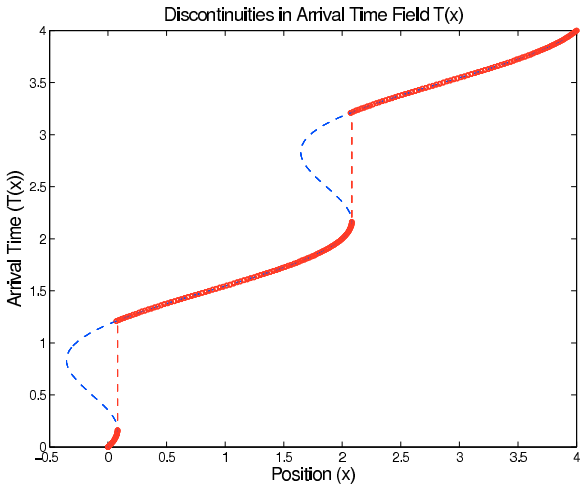


Figure 7-20: Discontinuities in the first arrival time field, $T_1(x)$ (plotted in red) caused by adverse the flow-field for the corollary discussed in §(7.3.1). The maximum flow speed is 2, which is larger than the vehicle speed F. The optimal vehicle path is plotted in blue. The start point is $x_s = 0$ and the end point is $x_f = 4$. The optimal path to the end point visits some points more than once, causing the discontinuity in the first arrival time field.

arrival times (in addition to the first arrival time) at these points when the velocity field is strongly adverse.

Solving the forward level set equation (5.2) gives the correct optimal solution, even with strong adverse flow currents because the zero level set front always corresponds to the reachability front. By keeping track of the reachability front, the algorithm automatically records multiple arrival times at various spatial points. Hence, the algorithm is applicable to weak and strong flow fields alike.

Now, let us consider the same 1-D example discussed above, but with a flow velocity, $\mathbf{V}(x,t) = -0.95\sin(\pi t)\hat{\mathbf{i}}$. The maximum flow speed in this case is equal to 0.95, which is not adverse to the vehicle motion. The trajectory of the vehicle in this case is plotted in blue in figure (7-21). The first arrival time field $T_1(x)$ is superposed in red. In this case, both the vehicle trajectory and the first arrival time curves are identical. This is expected because the vehicle does not experience strong adverse currents along its path. In this case, the $T_1(x)$ field is continuous everywhere and the first arrival time field is governed by equation (5.5).

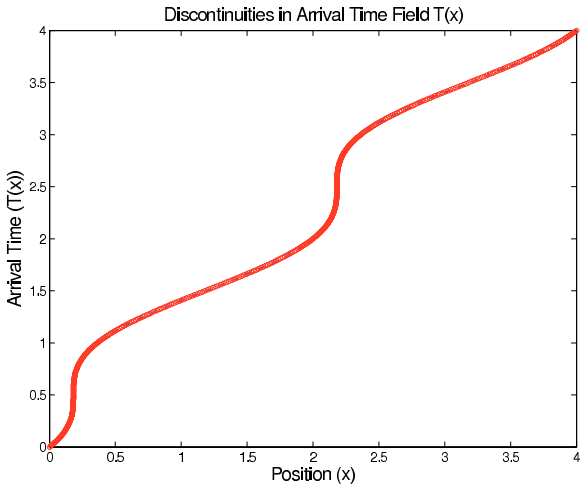


Figure 7-21: Optimal trajectory of the vehicle (blue) and the first arrival time field $T_1(x)$ for the corollary in §(7.3.1). The maximum flow-field speed is 0.95. The first arrival time field always coincides with the optimal trajectory because the flow is not adverse to the motion of the vehicle.

7.3.2 Determination of Starting Time

Our level set approach for path planning can also be used to determine *when* vehicles must be deployed, in addition to their sequence of headings, to reach their targets in the quickest time. In all the examples seen until now, we have assumed that the vehicle starts its motion at time $t_{start} = 0$, from the starting point ($\mathbf{x_s}$). However, if the vehicle is allowed to start at a later time (unknown a priori) it may be able to arrive at the target sooner than if it starts at $t_{start} = 0$. This can happen if the vehicle experiences strong adverse currents at the start which take it away from the target ($\mathbf{x_f}$). In such cases, it may be better for the vehicle to be deployed (from a ship, for example) at a later time when the adverse current has passed. We now present a simple example in 1-D where this situation occurs, and how our level set approach can be used to determine the optimal starting times. We use the same 1-D example that was discussed in §(7.3.1). The flow field is given by

$$\mathbf{V}(x,t) = -2\sin(\pi t)\hat{\mathbf{i}}$$

The maximum vehicle speed, F is set to 1. The starting point of the vehicle is the origin, $x_s = 0$, while its end point is $x_f = 2$. As seen earlier, the optimal trajectory

for the vehicle is governed by:

$$\frac{dx}{dt} = 1 - 2\sin(\pi t).$$

Let us assume that the vehicle is deployed at a variable start time $t_{start} \geq 0$, so that $x(t_{start}) = 0$. Our goal now, is to minimize the first arrival time at $x_f = 2$ by a suitable choice of t_{start}. Integrating the above equation and setting the limits, we have,

$$x(t) - 0 = (t - t_{start}) + \frac{2}{\pi}\left(\cos(\pi t) - \cos(\pi t_{start})\right) \tag{7.18}$$

This family of trajectories can be computed for different values of $t_{start} \geq 0$. The arrival time at $x_f = 2$ can consequently be computed for each of the trajectories. Some sample trajectories corresponding to starting times $t_{start} = \{0, 0.5, 0.833, 1.5, 2\}$ are plotted in figure (7-22).

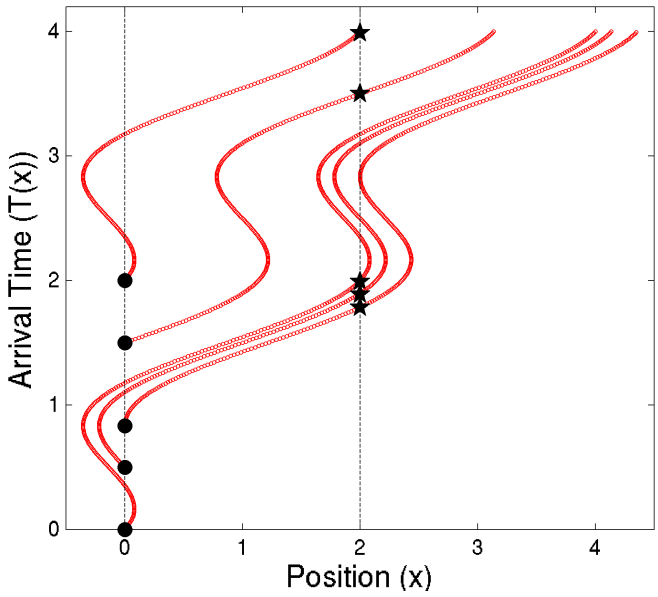


Figure 7-22: Sample trajectories of vehicle, for different starting times, t_{start}, (denoted by filled circles). The first arrival time for each trajectory is marked by filled stars. As observed, an earlier starting time does not necessarily lead to a quicker arrival time at the destination. The arrival time corresponding to $t_{start} = 0.834$ is the smallest.

From this figure, we can see that the trajectory corresponding to $t_{start} = 0$ reaches the target *later* than the one corresponding to $t_{start} = 0.5$. This happens because there

is a strong flow in the negative x direction at $t = 0$. Due to this, the vehicle is forced to reverse its path until a favorable (rightward) current appears, which can advect it towards the target.

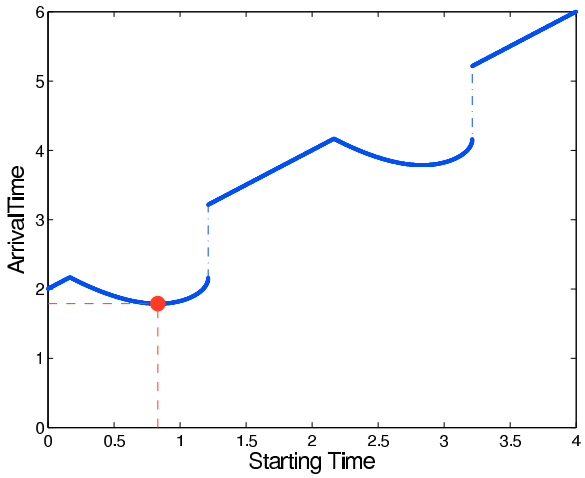


Figure 7-23: Plot of first arrival times at $x_f = 2$ against different starting times, t_{start} for the corollary in §(7.3.2). The minimum arrival time is obtained for $t_{start} = 0.834$. The corresponding arrival time is marked in red.

If the vehicle is deployed at a later time ($0.5 \leq t_{start} \leq 0.834$), the spatially uniform flow becomes favorable at t_{start}, and the vehicle thus does not travel any longer than necessary. The vehicle can move in the positive x direction when the flow speed becomes less than F, which happens at $t = 1 - \frac{1}{\pi}\sin^{-1}(0.5) \approx 0.834$. In this case, this is the optimal starting time, to reach $x_f = 2$. In figure (7-23), we plot the arrival times at $x_f = 2$ as a function of the starting time, t_{start}. This curve clearly shows that the quickest arrival time is for $t_{start} \neq 0$.

Our level set methodology can be used to compute the optimal starting time t_{start} of the vehicle. This can be done by keeping track of the reachability front corresponding to several starting times of the vehicle. Instead of one reachability front, we can now track an ensemble of fronts, each for one choice of the starting time. The starting time corresponding to the level set front that reaches the target fastest, is the optimal starting time. Once this is known, the optimal path can be calculated by solving the backtracking equation. Although this approach requires solving an ensemble of forward level set evolution equations, it is not still not very

expensive due to the low computational cost of the algorithm (see §(6.6)).

Our algorithm also lends itself to easy implementation of heuristics to decide when to evolve new level set fronts in order to reduce the computational cost of solving this problem. For example, one admissible heuristic could be to evolve new level sets whenever the flow at the start point is favorable (directed towards the goal).

7.3.3 Multiple Optimal Paths

In some situations, for a given problem configuration $(\mathbf{x_s}, \mathbf{x_f}, F, \mathbf{V}(\mathbf{x}, t))$, there may exist multiple trajectories with the same travel time. We present such a scenario in this example. We will show in this example that even though two end points are very 'close' to each other in space, the quickest paths to these points can be very different. Theoretically, for the point limit between the two, there are two possible optimal paths. In fact, such point locus of multiple optimal paths can also be more general (e.g. lines in 2D flows, surfaces in 3D flows etc.). We consider the example of a jet flow in a 2-D domain, as described earlier in §(7.1.2) and also in Lolla et al. (2012).

In this example, two identical vehicles ($F = 1$) start at the same position $\mathbf{x_s} = (1, 1)$ and at the same time, $t_{start} = 0$. Their destinations are $\mathbf{x_f^1} = (2, 0.8)$ and $\mathbf{x_f^2} = (1.95, 0.75)$ respectively. The distance between these end points is much smaller than the length of the vehicle trajectory. The fastest time paths of these vehicles are plotted in figure (7-24). From here, we can see that even though $\mathbf{x_f^1}$ and $\mathbf{x_f^2}$ are nearby each other in space, the optimal paths to these points are quite different. One of the trajectories is a straight line from start to end, and is not affected by the jet flow. The second path however, makes use of the jet flow to minimize the travel time. The travel times of both paths however are identical, and equal to 0.982.

The level set equation (5.2) admits weak solutions (see §(4.5)). Thus, the level set contours may develop sharp corners for certain flow fields. At these sharp corners, the normal to the level set front is not defined. We see such a behavior in this example. There exists a 'shock' line or locus formed by the level sets to the target points on which, multiple optimal paths exist. This line has been marked in figure (7-24). In

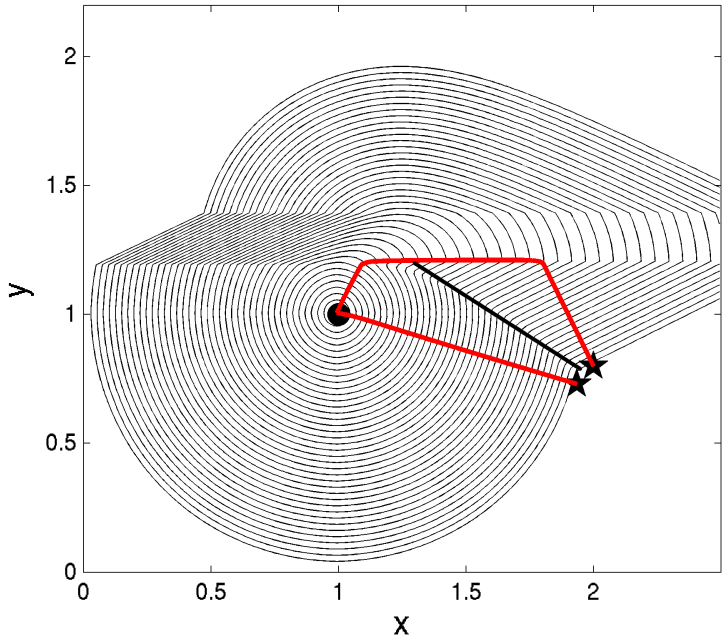


Figure 7-24: Time-optimal paths (red) from $\mathbf{x_s} = (1, 1)$ to two different end points, $\mathbf{x_f^1} = (2, 0.8)$ and $\mathbf{x_f^2} = (1.95, 0.75)$ overlaid on intermediate level set contours. Even though the two end points are very close to each other in space, the optimal paths leading to these points are widely different - one of the optimal paths uses the jet, while the other does not. The line marked in thick black indicates the set of points to which multiple optimal paths exist.

fact, several other similar examples can be constructed in which there exist multiple optimal paths to certain end points. The level set algorithm thus, can predict the existence of such points and compute optimal paths leading to them.

7.3.4 Validity for Compressible Flows

Our methodology enforces no conditions on the nature of the external flow-field, except requiring it to be different from an infinite impulse function. Since the velocity field imposes only a kinematic constraint on the vehicle's motion, the algorithm can be used, in principle, for path planning in a compressible flow as well.

7.4 Summary

In this chapter, we have illustrated a number of examples for our path planning algorithm. We have, first, presented a few benchmark examples where the analytical solution can be easily computed. For these examples, we have examined the performance of our level set algorithm. After establishing the validity of the methodology, we presented the results of the algorithm for three realistic ocean flow-field scenarios. In each example, we have highlighted a different feature of our methodology. Finally, we have discussed a number of interesting examples, which serve as corollaries to the theorem proved in §(5.2). In the following chapter, we discuss how our level set approach can be augmented to optimize coordinated path planning for a swarm of vehicles navigating through a dynamic flow-field.

Chapter 8

Coordinated Path Planning

In the previous chapters, we have described our level set method based path planning algorithm for underwater gliders navigating in optimal time through dynamic flow fields. We have illustrated a number of simple benchmark test cases validating our approach and have also applied our algorithm to more realistic, simulated ocean test cases. In this chapter, we describe two approaches, both based on level set methods, for coordinated control of multiple autonomous vehicles in dynamic flow fields. In this context, coordination refers to formation and maintenance of specific geometric shapes by multiple vehicles navigating together in the ocean. We aim to maintain specific geometric patterns for the vehicles and also minimize their travel time. First, we briefly explain the advantages of coordinated path planning and shape formation before describing relevant prior works in the literature that address this problem. We then, illustrate two novel approaches for shape formation and maintenance, using level set methods.

8.1 Background and Review

8.1.1 The Need for Coordination

Several areas of engineering and computer science have adopted the idea that *swarms* of coordinating agents can solve a lot of complex problems. A *swarm* is defined as a

collection of a number of agents, each of which performs a simple task, but the actions as whole produce a complex behavior (Hinchey et al., 2007). The idea of using swarms in science and engineering has partially been inspired by the behavior of higher-order animals such as colonies of ants, flocks of birds or packs of wolves. These animals exhibit complex social structures to coordinate with each other in order to achieve a common goal. Nature-inspired swarm technology deals with complex problems that may be very difficult to solve using traditional single-agent based approaches. Thus, scientists have tried to extend similar ideas of coordination to areas of learning, robotics, ocean engineering and many fields of computer science. In these fields, the term *swarm* refers to a large number of simple components which may work on the Earth's surface, underwater or in the air.

Coordination amongst a fleet of robotic agents is necessary in order to improve the overall performance of most robotic missions. Several missions (underwater or otherwise) use swarms of autonomous vehicles to accomplish various difficult and complex tasks in a cooperative fashion. Multi-robotic systems are versatile and efficient in exploration missions, military surveillance and cooperative manipulation tasks. There are several reasons why deploying multiple robots can be potentially more useful than using individual units (Balch et al., 1999). These include:

- Distributed Action: Multiple robots can be in several places at the same time.
- Inherent Parallelism: Multiple robots can do, perhaps different things at the same time.
- Divide and Conquer: Certain problems are amenable to be broken down into several smaller problems, which can then be allocated to a smaller group of robots.
- Often, for multi agent missions, the tasks assigned to individual robots become much simpler than when a single robot executes all the tasks.

Research on multi-robot applications has focused mostly on cooperative manipulation, navigation and planning, collaborative mapping, exploration and formation

control.

Since the main focus of this thesis is towards oceanic and underwater applications, the uses of coordination among underwater vehicles are of most relevance to us. Multi-vehicle missions have much to offer a variety of underwater applications. As underwater vehicles become more reliable and affordable the simultaneous use of several vehicles recently became a viable option and multi-vehicle deployments will become standard in the upcoming years (Bahr et al., 2009). Control of vehicle swarms will not only make possible entirely new types of missions which rely on cooperation, but will also allow each individual member of the group to benefit from navigation information obtained from other members. With sensors to measure the environment and coordination that is appropriate to critical spatial and temporal scales, the group can perform important tasks such as adaptive ocean sampling (Fiorelli et al., 2004, Bhatta et al., 2005, Leonard et al., 2010).

Since the ocean is characterized by a wide range of spatial and temporal scales, multi-vehicle tasks will also be associated with a range of characteristic scales. For example, when an AUV group must function as a communication network, the critical spatial scale may be small since there may be limits to how far apart adjacent vehicles can be, without compromising on the transmission of data. In some other sensing applications, ocean features may change quickly, and thus, smaller temporal scales may drive the mission (Fiorelli et al., 2004). Thus, spatial and temporal scales of the sampling mission may be one of the criteria that can be used to determine the control and coordination requirements for the vehicles (see also Bhatta and Leonard (2002)).

When each vehicle is equipped with a sensor to make measurements, the group of vehicles serve as a mobile sensor network. This network may serve to sample physical and biological variables in the ocean. Even for these applications, the range of relevant spatial and temporal scales may be very large. For example, sampling in a relatively large area may help in observing large-scale ocean processes such as upwelling and relaxation (Haley et al., 2009). Sampling in smaller areas of the ocean may help in identifying smaller scale features such as eddies and algal blooms. Thus, multi-AUVs and cooperative control have much to offer for ocean applications. For smaller scale

missions, vehicle speed, communication efficiency and pattern formation of AUVs may be of greater interest. For larger scale missions, vehicle to vehicle communication may be impractical, and one may be interested in optimizing endurance of the swarm. Therefore, cooperative control of autonomous vehicles has been gaining increasing importance in a number of fields and is partly, the focus of this thesis. An excellent review of design of mobile sensor networks for data collection is given in (Leonard et al., 2007).

Centralized and Decentralized Control

There are two main types of planners for coordinated vehicle motion namely,

1. Centralized planner

2. De-centralized planner

In a centralized planning scheme, all of the participating agents communicate with a central agent to report their information and new measurements. The central planner gathers this available information to produce coordinated plans for all agents, which are then redistributed to the team. The entire fleet of vehicles receives its instructions from a central server that is in charge of all the planning. No imperfections in communication are assumed. These types of systems are useful since they place much of the heavy processing requirements safely on the ground or aboard a ship, making the robots smaller and cheaper to build. Generating a coordinated plan using a centralized approach can be computationally intensive, but otherwise it is relatively straight forward because the central planner has access to all information. On the other hand, agents must consistently communicate with a fixed location, reducing the possible mission ranges that the swarm can handle, as well as creating a single point of failure in the mission.

In order to overcome the drawbacks of centralized planning schemes, some types of decentralized methods have been proposed in the literature. These methods usually work by placing a planner on each agent in order to increase the mission range, as well as remove the single point of failure. Decentralized algorithms (Amato et al.,

2011) generally make use of consensus algorithms to converge on an agreement before performing the assignment. These consensus algorithms allow the swarm to perform a wide variety of tasks, even in highly dynamic and uncertain environments. A decentralized coordination controller is more robust and flexible than a centralized controller.

8.1.2 Pattern Formation

In this thesis, we focus on a particular type of coordination between multiple vehicles, i.e. pattern formation. This is a fundamental problem in cooperative control and involves having a group of robots organize themselves into global formations or patterns (Swaminathan and Minai, 2005). The pattern formation problem can be described as a cooperative behavior between various robots that allows the robots to move in a stable configuration. These include simple patterns like circles, triangles, squares, or a line formation.

Pattern formation control has drawn significant attention for many years now. Controlling the formation of air and ground vehicles is becoming an important research topic in robotics and controls community (Bendjilali et al., 2009). The ability of mobile robots to navigate autonomously in a stable configuration and avoid obstacles is central to various applications in both military and civilian domains. These applications range over transportation, exploration, surveillance and rescue. Various multi-robot configurations are also used for distributed sensing and coverage. Pattern formation enables convenient establishment of a coordination between participating agents.

Pattern formation is also helpful for modular robots. Modular robots are a class of robotic systems composed of many identical, physically connected, programmable modules that can coordinate to change the shape of the overall robot. Thus, by transforming its shape, a modular robot can adapt to many tasks, from different modes of locomotion like climbing and crawling to performing robotic structures such as stairs and bridges (Yu and Nagpal, 2008). Pattern formation control is of great importance in these units as well.

In the context of underwater applications, formation of pattern among the vehicles allows for consistent estimation of features in the flow-field, such as gradients of the ocean current speeds. Single vehicle sampling missions will provide local field data such as velocity of the ocean current, temperature, salinity etc. at various times and at different points along the vehicle's trajectory. However, multiple vehicles that maintain certain patterns will enable the estimation of other quantities of interest such as gradients of velocity fields, temperature and salinity distributions *at one fixed time*. Some patterns may be more useful than others to identify certain features in the ocean.

The following are some other areas where pattern formation among participating autonomous agents can potentially be useful (Swaminathan and Minai, 2005).

- Surrounding an object or feature in the environment - This may involve formation of a uniform distribution of robots in a given area for protecting the area or for surveillance.
- Removal of mines and bomb disposal, in military applications.
- Election of a leader or in follow-the-leader situations, with possible military applications.
- Gathering to share information or for some other collaborative task.
- Formation of sensing grids.
- Carrying, moving and assembling objects.
- Aiding emergency response and decongestion of traffic on highways.

8.1.3 Literature Review

In this section, we briefly discuss some approaches for pattern formation that have been proposed in literature (Swaminathan and Minai, 2005).

1. **Behavior based approach**: In a behavior based approach, several desired behaviors (collision avoidance, formation keeping, target seeking etc.) are prescribed to each robot. The final robot action is derived by weighting the relative importance of each behavior. The theoretical formalization and mathematical analysis of this approach is challenging and this is the main drawback of the approach. The other drawback is that the method does not guarantee the convergence of the formation to the desired configuration.

2. **Virtual structure approach**: Virtual structure approaches considers the robot formation as a single virtual rigid structure so that the behavior of the robotic system is analogous to a physical object. Desired trajectories are not assigned to individual robots, but to the unit as a whole.

3. **Leader-following approach**: In the leader-follower approach, one of the robots is designated as the leader, while the others are the follower robots. The leader moves along a trajectory which can be pre-computed, while the followers are to maintain a desired posture (distance and orientation) to the leader. The main criticism to this approach is that the method does not tolerate leader faults and exhibits poor disturbance rejection features. However, despite these drawbacks, it is very scalable and easy to implement.

Various methods are discussed in the literature to solve various problems in formation and cooperative control. Arkin and Balch (1998) develop a shape-formation algorithm using behavior based approaches. This method is highly flexible for various tasks such as guiding the robots through uncertain and dynamic environments using local sensor information only. The robots are initially positioned randomly in a defined area. During this formation, the robots are also desired to coordinate among themselves to avoid collisions and keep the desired distance between each other. The main drawback of the approach is the difficult mathematical analysis and the convergence to a particular pattern cannot be guaranteed.

A potential field approach for pattern formation that is quite similar to the classical potential field is suggested by Reif and Wang (1999) and modified by Dudenhoeffer

and Jones (2000) to control a distributed autonomous multi-robot system. Here, artificial force laws are defined between pairs of robots or robot groups. The authors call the method *social potential field*, since it deals with interaction between multiple agents. Bendjilali et al. (2009) discuss a method for modeling, initialization and control of formation of mobile robots that involves a combination of classical guidance laws and kinematics rules. Using relative vehicle kinematics, they are able to derive governing differential equations that enable the modeling and control of dynamic robot formations. They finally use a leader-follower based method to plan individual robot paths. They illustrate the formation of a straight line pattern, circle and a general polygon formation in stationary environments.

Consolini et al. (2007) describe another work that makes use of leader-follower approach for pattern formation of non-holonomic robots. They discuss how different geometric patterns of the robots affects the admissible trajectory of the leader by placing bounds on the maximum curvature of the leader path. The main difference from other leader-follower based approaches is the the angle between the leader and the followers is measured in the follower reference frame, instead of the leader's. Sisto and Gu (2006) also utilize a leader-follower approach, but consider the effect of noisy communication between various robots in the formation and develop a fuzzy control algorithm for formation control. Ren and Sorensen (2008) describe a formation control architecture which requires communication only between neighbors. An extended consensus algorithm is applied to estimate the group trajectory in a distributed manner. Then, a consensus based formation control strategy is then applied for vehicle level control.

The use of graph search techniques coupled with leader-follower approaches for pattern formation was first shown by Desai (2002). After planning the path for the leader, their algorithm computes control for the follower robots using ideas of nonlinear control theory and graph theory. Sabattini et al. (2011) and Fierro et al. (2002) describe another use of artificial potential techniques for decentralized robotic controllers in order to maintain a polygon shape. They prevent the occurrences of local minima in the solution by using Lyapunov functions (Zhang and Leonard, 2006).

Another work that makes use of artificial potential functions for pattern formation is (Leonard and Fiorelli, 2001). The main feature of this artificial potential function is that it is constant beyond a certain distance $d > 0$. This leads to a distributed control law for each vehicle and this does not depend on the position and velocity of all other vehicles in the formation. However, this method has the drawback of not being able to guarantee stabilization of the system of multiple vehicles to a *unique* desired formation. The pattern formed by the vehicles is a function of the number of vehicles and their initial conditions, which is highly undesirable. Yang and Zhang (2010) also use a similar approach for shape formation.

A different approach is proposed by Belta and Kumar (2001). They describe a geometric algorithm for pattern formation of under-actuated robots. The problem of optimal trajectory generation of the robots is cast in terms of designing optimal curves on an Euclidean Lie group. This leads to paths that minimize energy of translations and rotations, while maintaining a fixed pattern. Olfati-Saber and Murray (2002) propose a framework for formation stabilization of multiple autonomous vehicles in a distributed fashion. They pursue an idea of *structural potential functions* for distributed control stabilization of multiple vehicles. Ogren and Leonard (2003) describe an algorithm for pattern formation in the presence of obstacles in the domain.

The use of implicit shape functions for pattern formation can be seen in (Chaimowicz et al., 2005). The desired shape is set as the zero isocontour of a suitable function and robots are then controlled by a gradient descent technique augmented with robot to robot repulsion to spread along the zero function isocontour. Esin et al. (2008) describe another implementation of using implicit function representations for the desired shape of formation. Shape formation is controlled by using potential fields generated from implicit polynomial representations and the control for keeping the desired shape is designed by using Elliptical Fourier Descriptors (EFD). Coordination between adjacent neighbors is modeled by linear springs connectors. The authors claim that this approach offers more flexibility in the formation shape and is suitable to a wide range of swarm sizes. For results of a virtual deployment of a pattern of vehicles, the reader is referred to (Paley et al., 2008, Zhang et al., 2007).

Derenick and Spletzer (2007) describe a method to convert the problem of pattern formation into a convex optimization problem. Their inspiration comes from the realization that the operation of a team that maintains a specific pattern is inherently a constrained resource allocation problem. However, several other authors have raised scalability concerns of such an optimization based approach (see Kalantar and Zimmer (2006), Liu et al. (2011)). For other papers on pattern formation, the reader is referred to (Fredslund and Mataric, 2002, Yang and Zhang, 2012, Zhang, 2007, Swaminathan and Minai, 2005, Paley et al., 2006).

In all of the above described works on pattern formation, the motion of the robot is not affected by the dynamic nature of the environment. Since underwater gliders are easily susceptible to strong flow currents, we should also consider the effect of the ocean currents on the motion of gliders and on the control for their pattern formation. This motivates us to utilize our level set approach for pattern formation because it accounts for the dynamic flow-field, computes time-optimal paths and can also deal with obstacles to the vehicle's motion in a straight-forward manner. In what follows, we present two approaches for pattern formation based on level set approaches.

8.2 Pattern Formation using Local Control

In this section, we present a *local control approach* for pattern formation for underwater gliders that respects the kinematics of the motion of individual members of the swarm. This method is inspired from a virtual structure approach for pattern formation (see §(8.1.3)).

8.2.1 Methodology

The local control approach presented in this thesis works by assigning a virtual *center of mass* to the pattern we wish to maintain. This center of mass (henceforth denoted by $\mathbf{C}$) is the geometric center of mass of the pattern whose members are the underwater gliders we wish to navigate in a cooperative fashion. For example, the center of mass and the members of an equilateral triangle pattern are shown in figure (8-1).

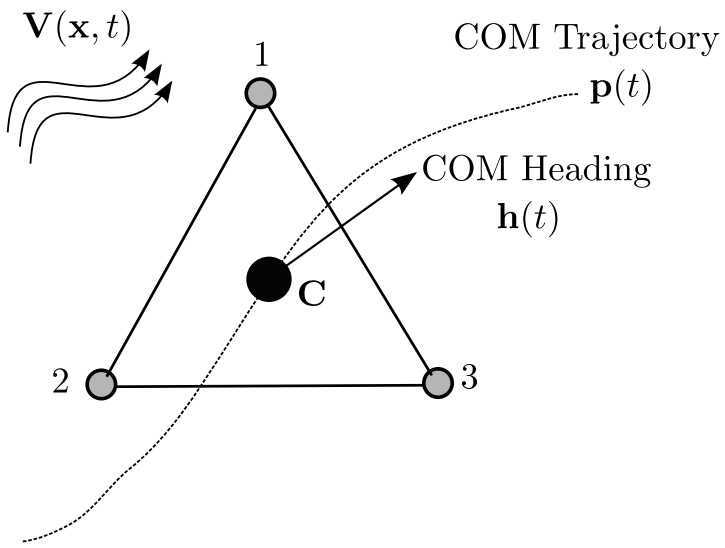


Figure 8-1: Center of Mass (**C**) of an equilateral triangle pattern of vehicles. The vehicles are located at the vertices of the triangle. The trajectory of **C**, $\mathbf{p}(t)$ is computed along with its sequence of headings $\hat{\mathbf{h}}(t)$ using the level set path planning algorithm.

It should be noted that **C** does not represent a vehicle. It is only a virtual point that is characteristic of the pattern we desire to maintain. This local control algorithm is a first order 'predictor-corrector' approach.

The local control algorithm works in the following steps.

1. Using the start point, $\mathbf{x_s}$, the end point, $\mathbf{x_f}$, the maximum vehicle speed F, and the velocity field, $\mathbf{V}(\mathbf{x},t)$, the optimal path of the center of mass **C** is planned by solving equation (5.2) and equation (5.4).

2. The optimal path $\mathbf{p}(t)$ and the heading sequence $\hat{\mathbf{h}}(t)$ of the center of mass **C** are stored.

3. Let us focus on one member of the swarm, say vehicle 1 (see figure (8-1)). As a first approximation, we let vehicle 1 move along the heading direction of the center of mass **C** . In other words, we assume that the heading of vehicle 1 at time t is equal to $\hat{\mathbf{h}}(t)$.

4. Using this heading direction, we predict the position of vehicle 1 one time-step later (i.e. at time $t+\Delta t$) by numerically solving equation (2.1). Let us call this

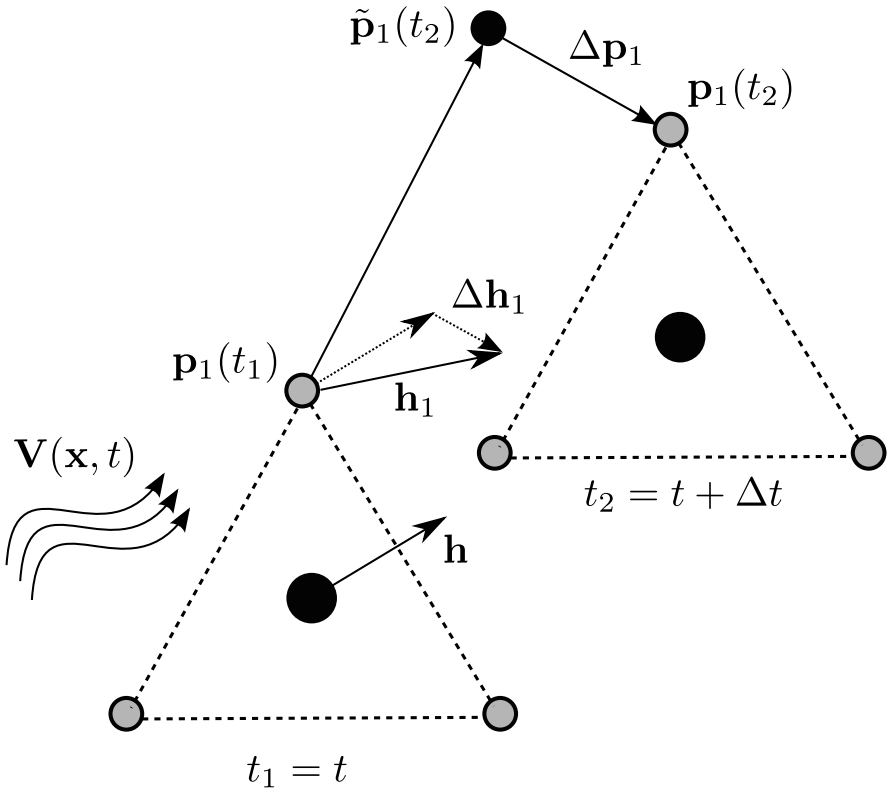


Figure 8-2: Heading of vehicle 1, $\mathbf{h}_1$ computed by adding a correction $\Delta\mathbf{h}_1$ to the heading of the center of mass $\mathbf{C}$, $\hat{\mathbf{h}}$.

position $\tilde{\mathbf{p}}_1(t + \Delta t)$. This is the 'prediction' step of the algorithm. We refer the reader to figure (8-2) for a visual depiction of the motion.

5. The trajectory of $\mathbf{C}$, $\mathbf{p}(t)$ is known. Assuming that the size and orientation of the pattern does not change, the desired position of vehicle 1 at time $t + \Delta t$ can be calculated. Let us denote this position as $\mathbf{p}_1(t + \Delta t)$.

6. If there is no velocity field, i.e. $\mathbf{V}(\mathbf{x}, t) = 0$, then $\mathbf{p}_1(t + \Delta t) = \tilde{\mathbf{p}}_1(t + \Delta t)$. In fact, if the velocity field near vehicle 1 is identical to the velocity field near the center of mass $\mathbf{C}$, then $\mathbf{p}_1(t + \Delta t) = \tilde{\mathbf{p}}_1(t + \Delta t)$.

7. However, in the general case, $\mathbf{p}_1(t + \Delta t) \neq \tilde{\mathbf{p}}_1(t + \Delta t)$. Therefore, we need to add a correction to the heading $\mathbf{h}_1(t)$ of vehicle 1. This correction can be computed by first calculating $\Delta\mathbf{p}_1 = \mathbf{p}_1(t + \Delta t) - \tilde{\mathbf{p}}_1(t + \Delta t)$.

8. The correction to the heading $\Delta\mathbf{h}$ can be calculated as

$$\Delta\mathbf{h}_1 = \frac{\Delta\mathbf{p}_1}{F\Delta t} \tag{8.1}$$

9. The above correction is added to the center of mass heading $\hat{\mathbf{h}}(t)$ to compute the new heading direction for vehicle 1. This completes the 'correction' step.

$$\mathbf{h}_1(t) = \Delta\mathbf{h}_1 + \hat{\mathbf{h}}(t) \tag{8.2}$$

Since the heading vector should have unit magnitude, we normalize this vector to get:

$$\hat{\mathbf{h}}_1(t) = \frac{\mathbf{h}_1(t)}{|\mathbf{h}_1(t)|} \tag{8.3}$$

10. This new heading direction, $\hat{\mathbf{h}}_1(t)$ is used to numerically solve equation (2.1) and evolve the trajectory of vehicle 1.

11. The trajectory of every other vehicle in the pattern is evolved using a similar procedure (starting from step 3).

Thus, the algorithm predicts the headings of every member of the group at each time, until the center of mass of the pattern reaches the end point. Next, we illustrate an application of the above pattern formation algorithm for a realistic ocean scenario and then discuss the advantages and drawbacks of the approach.

8.2.2 Application

We apply the above described local control pattern formation algorithm to the same flow past a circular island example discussed in §(7.2.2). We aim to compute paths of three different groups of vehicles which have to maintain square, triangle and a line pattern (see figure (8-3)). In addition, we want to compute collision free paths for these vehicles which respect the kinematics of motion, i.e. advection due to the flow field. We refer the reader to §(7.2.2) for specifications of the example (such as details of the flow field, numerical parameters used to solve the level set equation and

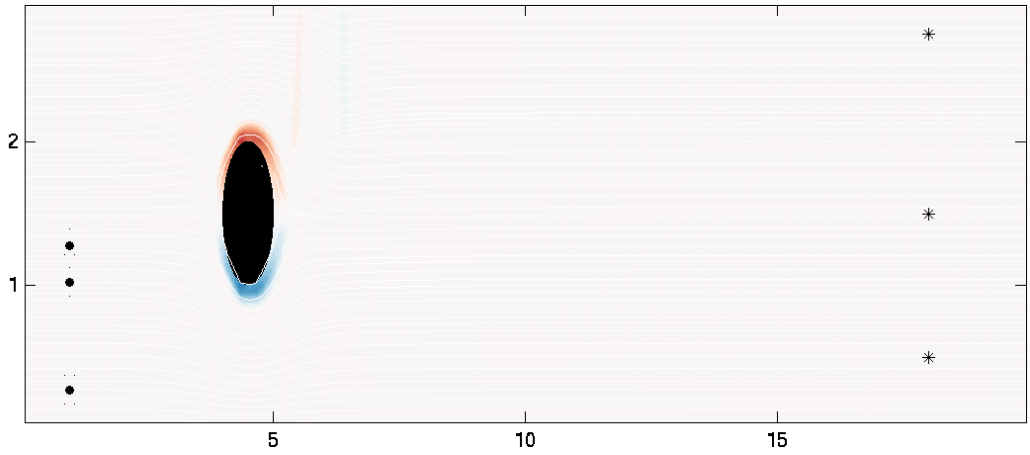
for flow-field generation). The start points of the vehicles are upstream of the island and are marked in filled circles. The end points are far downstream of the island and marked as stars. In figure (8-3), we show the time evolution of each vehicle path along with the position of the center of mass **C** of the respective pattern. As seen from the results, the algorithm performs adequately well in maintaining the specified geometric patterns for all the three groups of vehicles.

8.2.3 Discussion

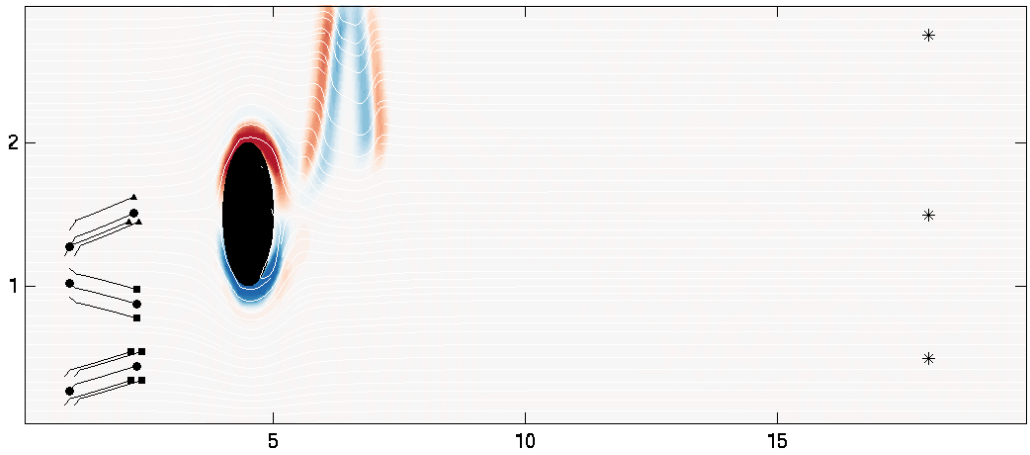
The local control algorithm presented here is simple to implement and computes obstacle-free paths for the participating agents. Another advantage of this method is that the computation of the path of the center of mass and of the pattern formation is decoupled. This allows the center of mass to be optimized for varied criteria, e.g. time optimal, obstacle avoidance, uncertainty reduction etc., while still aiming to maintain a specific pattern. Of course, this decoupling can be a disadvantage too: if the flow fields are very different for the different vehicles in the formation (e.g. the pattern is large-scale compared to the ocean scales), then, the decoupled solution will be too far from the coupled optimal (e.g. time optimal) and/or may not be able to maintain the desired pattern. The first-order correction to the heading is only effective when the velocity field at the center of mass **C** is close to that experienced by the vehicles. This may not be the case for larger sized formations. This can occur even if second order or other corrector schemes are employed (instead of first-order scheme employed here). In addition, the path of the center of mass may completely diverge from those of the vehicles, in some cases. In such cases, the center of mass **C** may not characterize the shape of the pattern any more. These drawbacks lead us to the next approach on pattern formation, which we present in the following section.

8.3 Shape Formation using Level Set Method

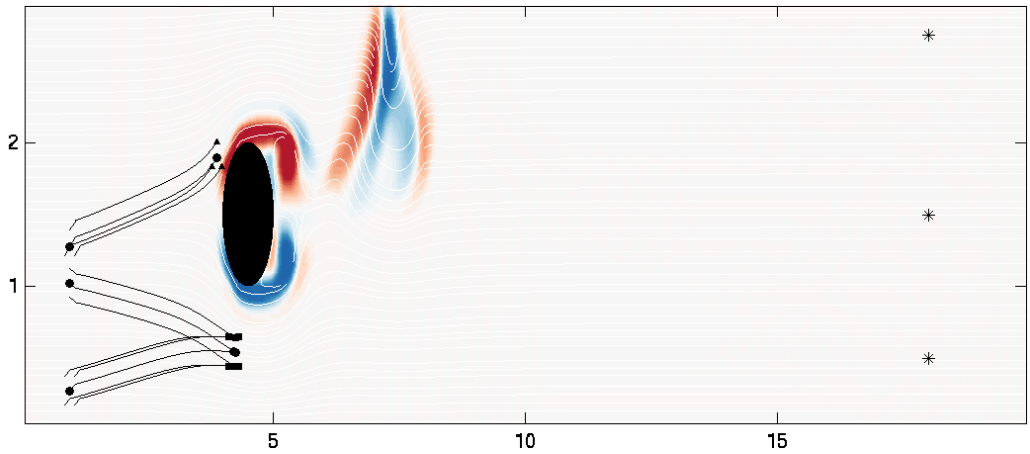
In this section, we present a pattern formation algorithm based on level set methods. This algorithm is developed using a 'leader-follower' approach, discussed earlier in



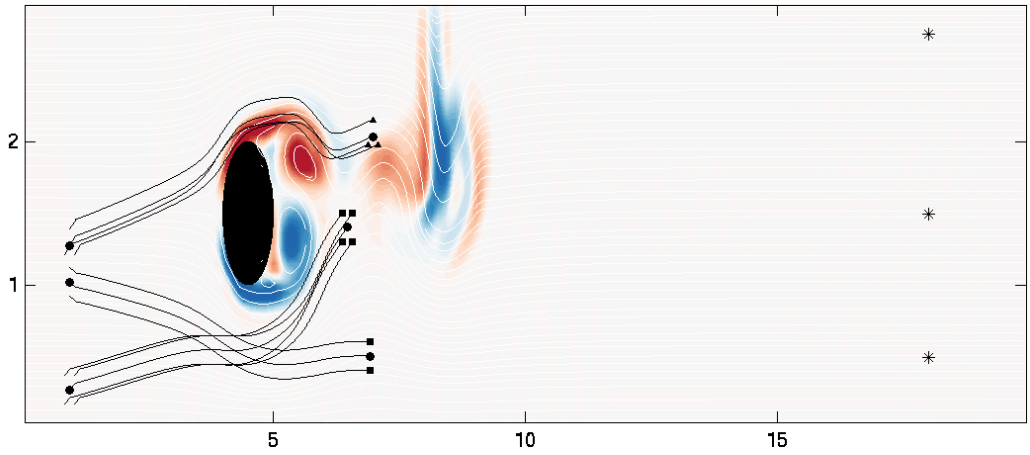
(a) $t = 0.000$



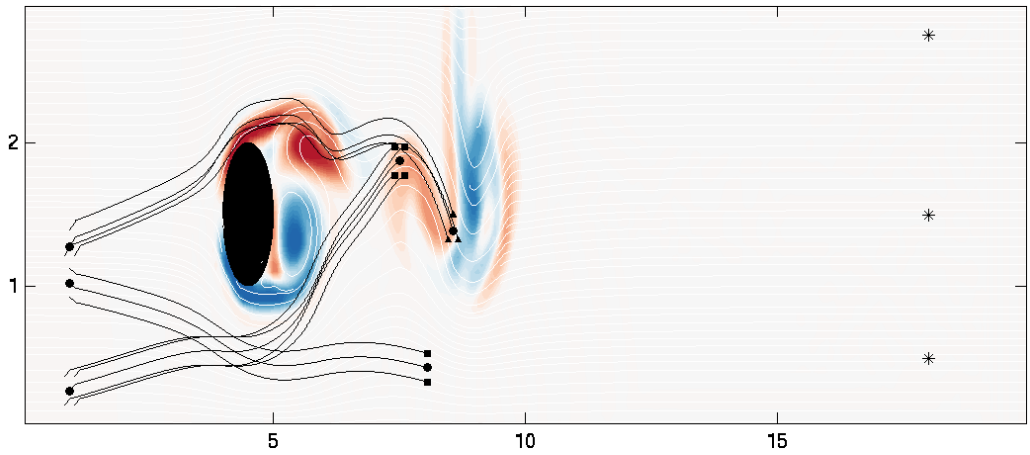
(b) $t = 0.500$



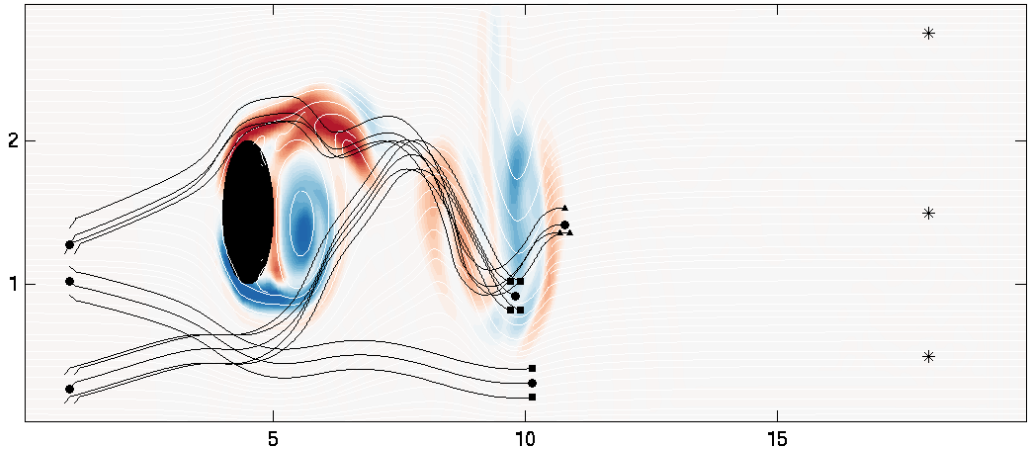
(c) $t = 1.300$



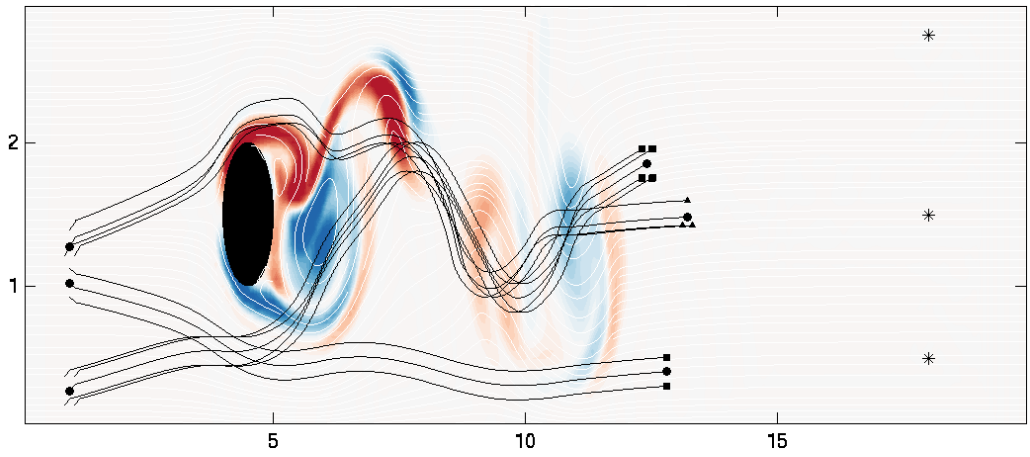
(a) $t = 2.250$



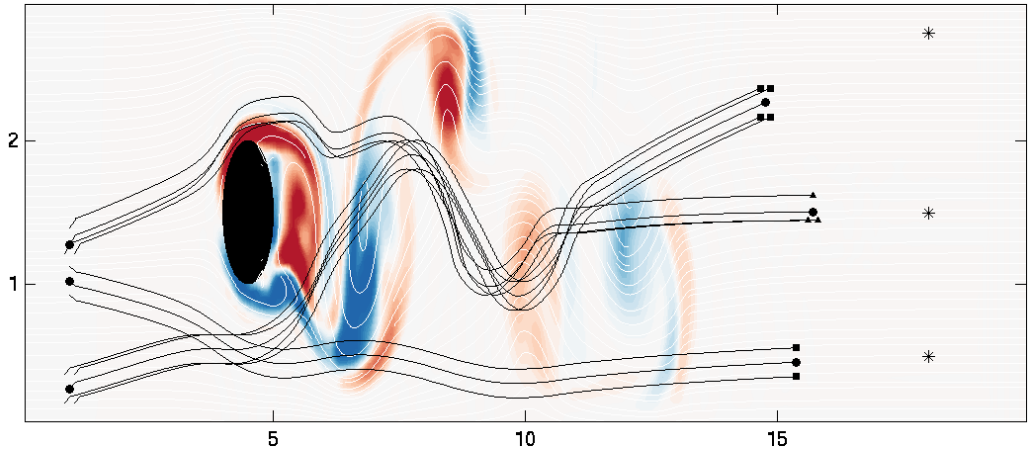
(b) $t = 2.750$



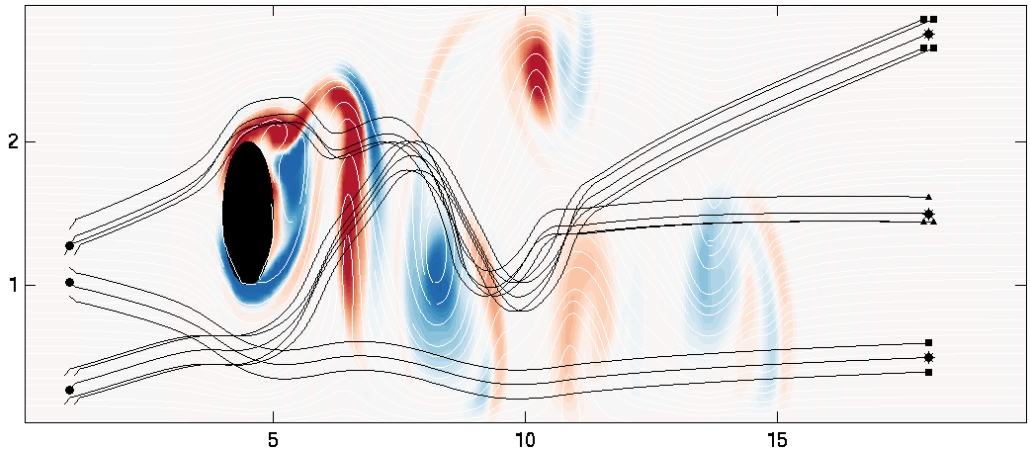
(c) $t = 3.50$



(a) $t = 4.50$



(b) $t = 5.50$



(c) $t = 7.00$

Figure 8-3: Time evolution of trajectories of three different groups of vehicles in the flow past circular island example - The vehicles maintain a square formation, a line formation and a triangle formation. Vehicle paths are overlaid on plots of the velocity field colored by vorticity.

§(8.1.3). In what follows, we first describe the algorithm and then apply it to a simulated ocean flow-field. We then discuss the advantages and the drawbacks of the algorithm.

8.3.1 Methodology

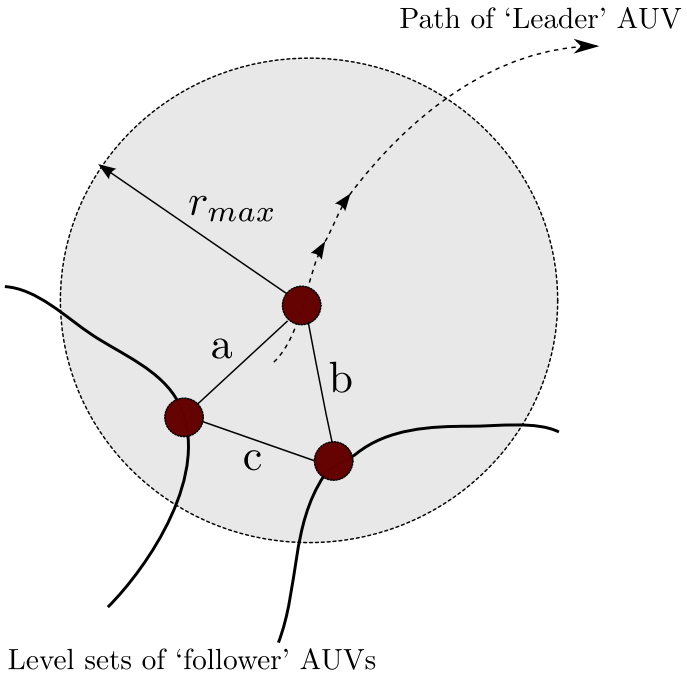


Figure 8-4: The *shape check* operation for maintenance of an equilateral triangle pattern for three vehicles. The vehicles are marked as brown circles. The uppermost vehicle is the 'leader' while the others are the 'follower' vehicles. In the *shape check* operation, the algorithm extracts points from the level sets of the follower vehicles by optimizing an objective function. The radius r_{max} can be increased to allow for larger size patterns.

The level set based 'leader-follower' approach for pattern formation consists of the following steps.

1. One of the vehicles in the group is randomly designated to be the 'leader' while all the other vehicles are the 'followers'. The path of this 'leader' vehicle is first computed by solving equation (5.2) and equation (5.4). The optimal trajectory of the leader $\mathbf{p}(t)$ and the headings $\hat{\mathbf{h}}(t)$ are stored. This part of the algorithm is identical to the local control approach.

2. Let us denote the travel time of the leader as T_l. We divide the interval $[0, T_l]$ into N smaller disjoint sub-intervals (called sub-periods), $I_i = [t_i, t_{i+1}]$, where $t_1 = 0$ and $t_{N+1} = T_l$. At the end of the time corresponding to each sub-interval I_i, we perform a *shape check* operation for the members of the pattern. N is the number of shape check operations performed in total.

   (a) At the initial time $t = t_1 = 0$, we solve the forward level set equation (5.2) corresponding to each of the follower vehicles. This is done by evolving separate level set fronts for each of the follower vehicles from their respective starting positions. These equations are solved until the end of the first sub-period, i.e. until time t_2. Note that we do not yet solve the backtracking equation (5.4), since we do not know the intermediate end points for the follower vehicles.

   (b) The *shape check* operation that is performed at the end of every sub-period computes the intermediate end points for each of the follower vehicles. We refer the reader to figure (8-4). In this figure, we show how a triangle pattern can be maintained by the leader and two follower vehicles. The forward level set equation (5.2) gives us level set reachability fronts corresponding to every follower. Recall that the reachability fronts outline the furthest any vehicle can go in a flow-field. From these reachability fronts, we have to extract the intermediate end points of the vehicles. For this, we perform an optimization by doing a local search (here, assuming we look for an equilateral triangle pattern):

$$\min |a - b| + |b - c| + |c - a| \tag{8.4}$$

   for points on the level sets within a distance r_{max} away from the position of the leader vehicle (see figure (8-4)). This local search is suitable for maintaining a triangular pattern because it minimizes the difference of the triangle sides. This optimization gives us points on the level set fronts which every follower vehicle must reach.

(c) Once the intermediate points for the follower vehicles are known, the backtracking equation (5.4) is solved to compute the trajectories of the follower vehicles during that sub-period.

3. After the paths of each follower vehicle are computed in a particular sub-period, we proceed to the next sub-period, by going back to step 2(a). This process is repeated until we reach the end of the travel period of the leader.

This algorithm, therefore, predicts the headings and the trajectories of the pattern members by solving several smaller sub-problems successively. The quality of the geometric pattern is verified at each shape check step. In the next section, we present an application of this algorithm for a wind-driven double gyre example.

8.3.2 Applications

In this section, we examine the performance of the level set based pattern formation methodology for a realistic ocean-scenario. The example we use for this purpose is the wind-driven double gyre flow-field, which was earlier described in §(7.2.1). In this example, we wish to navigate three vehicles through the flow-field such that they maintain an equilateral triangle pattern during the course of their trajectories. We refer the reader to §(7.2.1) for various details of the flow field and numerical parameters used to solve the level set equation. The results of our pattern formation algorithm are shown in figure (8-5). In this figure, we show the time evolution of the trajectories of each of the three vehicles overlaid on streamlines of the time-dependent flow-field. The rightmost vehicle in each panel is the leader of the group. As seen from these results, the algorithm performs well in maintaining an equilateral triangle pattern.

An interesting point to note, from this example, is that the size and orientation of the triangle are not fixed. We can see that as the group of vehicles approaches the center of the domain (where the horizontal flow gradient $\frac{\partial u}{\partial y}$ is not large), the triangle size reduces. Away from the center, (where $\frac{\partial u}{\partial y}$ becomes larger), the triangle size increases. This happens because, whenever the vehicles in the pattern experience

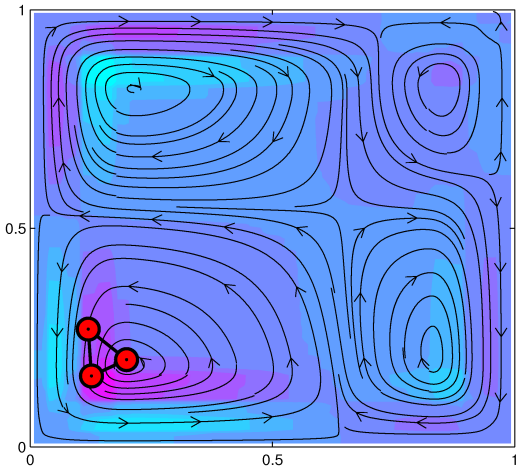
largely different flow fields, maintaining a smaller pattern size is not always possible. The algorithm gives preference to maintaining the shape of the pattern over the size of the pattern. As a result, the size of the pattern needs to be increased, in order to maintain the equilateral triangle shape. Therefore, the algorithm adjusts the size and orientation of the pattern based on the flow gradients experienced by the vehicles.

8.3.3 Discussion

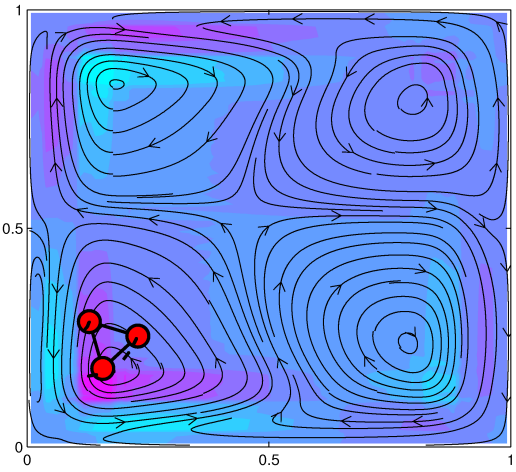
The pattern formation algorithm described in this section overcomes several of the drawbacks of the local control based approach. The size of the pattern can be variable. The local control approach does not allow for variable pattern sizes, nor guarantees the formation of any specific pattern. However, since the level set pattern formation algorithm involves computing feasible intermediate end points for the follower vehicles from their level sets, the desired shape is always maintained. The radius of the search r_{max} can be increased to expand the search space if the desired pattern is not feasible at any time. This algorithm adapts the triangle size and orientation to length scales of the flow. However, the local control approach performs well only when the size of the pattern is smaller than length scales of the flow, but fares poorly when the pattern size increases.

In this algorithm, we have assumed that each vehicle must navigate at maximum speed F, relative to the flow. However, this condition is not absolutely necessary. Earlier in this thesis, the focus was on computing time optimal paths of the vehicle, and this required the vehicle to navigate at maximum nominal speed F relative to the flow. In the present problem, the focus is more on maintaining geometric patterns than time-optimality. Hence, we can relax the time-optimality requirement of the paths and allow vehicles to head at speeds lesser than F, if it helps maintaining more accurate patterns. As a starting point however, we can let the vehicles move at speed F relative to the flow.

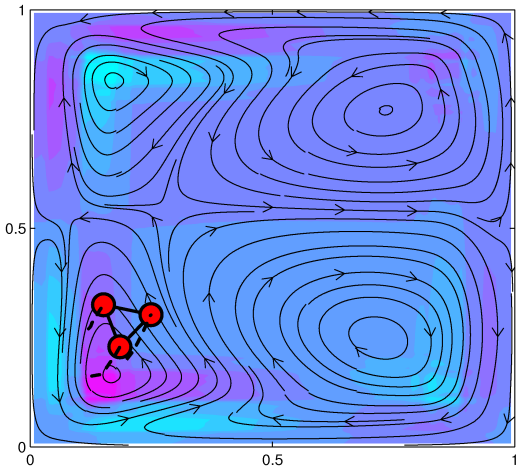
In some situations, the vehicles may need to change their formation pattern during the course of their trajectories. For example, four vehicles, originally moving in a square pattern may need for change their pattern to a triangle after some time. This



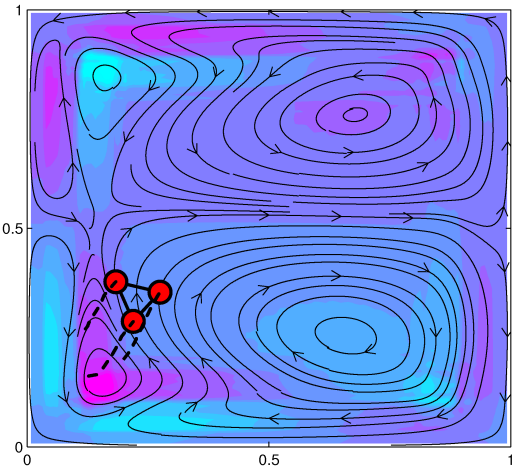
(a) $t = 0.000$



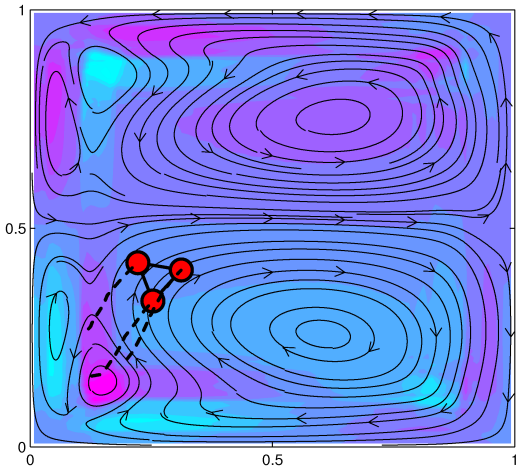
(b) $t = 0.01$



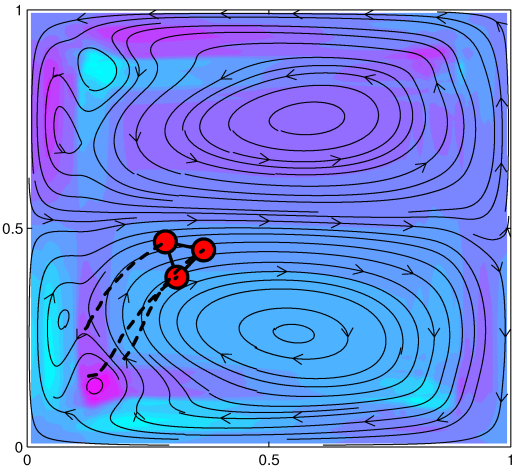
(c) $t = 0.02$



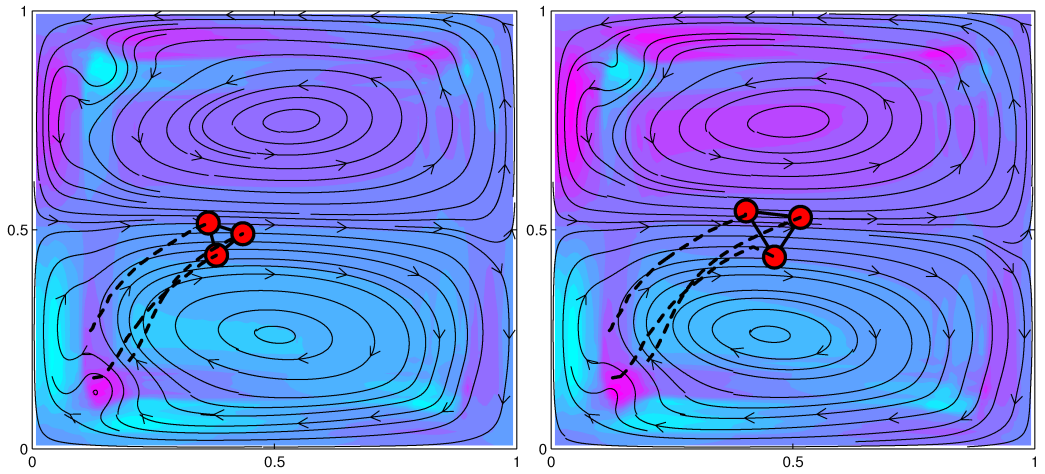
(d) $t = 0.0.03$



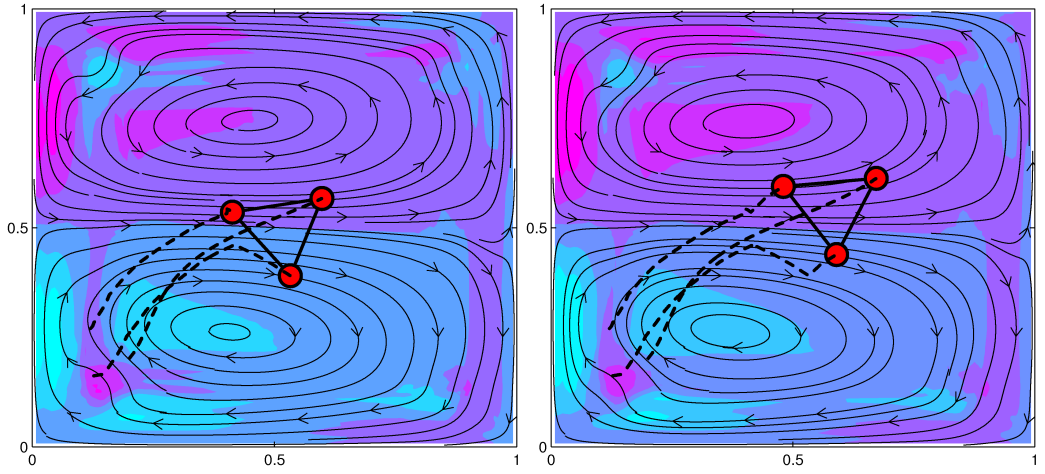
(e) $t = 0.04$



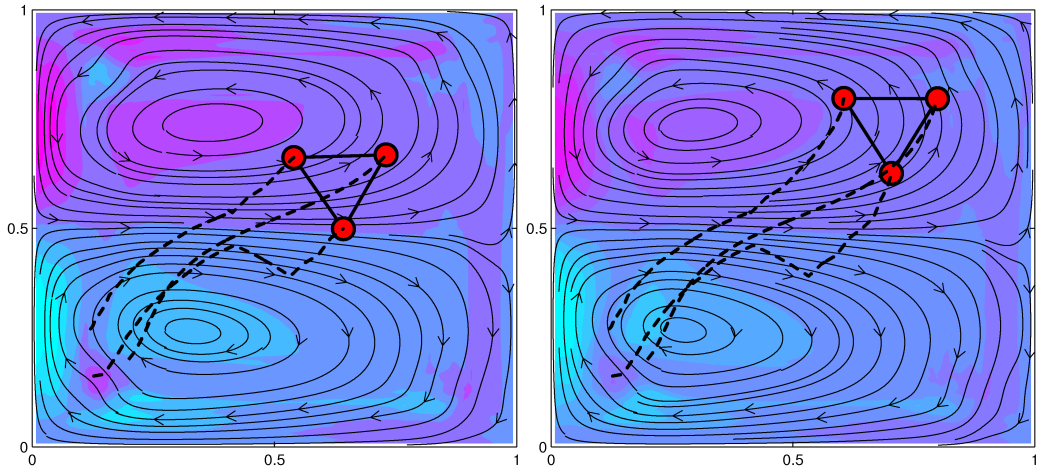
(f) $t = 0.05$



(a) $t = 0.06$ (b) $t = 0.07$



(c) $t = 0.08$ (d) $t = 0.09$



(e) $t = 0.10$ (f) $t = 0.12$

Figure 8-5: Paths of three vehicles which maintain a triangle pattern in a wind-driven double gyre flow-field. The size of the triangle is a function of the gradients in the flow-field.

can easily be incorporated in the level set pattern formation algorithm by appropriately modifying the cost function during the shape check operation. The shape check routine will compute updated intermediate points for the vehicles so that the new formation is realized.

8.4 Summary

In this chapter, we have presented two predictor-corrector methods for pattern formation of multiple vehicles moving together in an arbitrary flow-field. The first, local control approach is a virtual structure method for pattern formation which involves defining a virtual 'center of mass' of the pattern and using its planned path to compute and correct the headings of the vehicles in the pattern. The second approach presented is based on a 'leader-follower' approach for pattern formation. This approach was shown to overcome some of the drawbacks of the local control approach.

In the next and final chapter, we conclude by summarizing the main contributions of this thesis and provide some directions for future work.

Chapter 9

Conclusions and Future Work

In this final chapter, we first summarize the main contributions of this thesis and highlight the key results. Then, we describe some possible directions for future work.

Autonomous platforms such as AUVs (specifically, gliders and other propelled vehicles) are playing an ever-increasing role in several oceanic applications. Their high levels of autonomy and their ability to endure long range sampling missions are highly attractive features for ocean sampling and prediction tasks. This has led to an extensive spurt of research towards the possibility of using several such autonomous vehicles to execute missions in a cooperative fashion. Coordinated missions are expected to help individual units benefit from the information gained by other members of the group. Coordination among several vehicles makes certain types of missions possible, which otherwise, cannot be executed by single-vehicle systems. This has been the main motivation behind this thesis.

Underwater vehicles which operate in the coastal ocean frequently experience dynamic ocean currents whose effects on vehicle motion cannot be ignored. Unlike most robotic scenarios, the environment of the vehicle (i.e. flow-field) directly affects the motion of the vehicle. In addition, the coastal regions (where these vehicles typically operate) often have complex geometries and may be composed of several regions that are not safe for the vehicles to visit. All these factors make it difficult to even generate feasible trajectories that respect motion constraints of the vehicle. The methodology proposed in this thesis not only solves this problem, but also optimizes the travel

time of vehicles in an exact manner.

In this thesis, we developed an efficient and rigorous methodology based on level set methods, to solve the minimum-time path planning problem for autonomous vehicles navigating in dynamic flow-fields. Our methodology is based on solutions to governing partial differential equations and this obviates the need for any heuristic or ad-hoc assumptions.

One of the main contributions of this thesis is the theorem that establishes the optimality of our level set based approach to solve the minimum time path planning problem, in any type of flow-field. Our methodology combines several branches of science and engineering. It is inspired by fluid dynamics and level set methods to derive governing differential equations that predict reachability fronts and time-optimal paths of robots, with roots in control theory and robotic path planning. Equations are then solved efficiently at desired resolution using computational engineering, specifically, computational fluid dynamics and narrow-band schemes.

We have described various features of our methodology with the help of a number of sets of examples. We have also illustrated the versatility of our approach by extending and applying it to several examples of coordinated time-optimal path planning.

An important constraint for real-time underwater path planning operations is the computational cost of the path planning algorithm. Even though we have implemented a narrow-band solver, this has not been extensively tested for optimal efficiency. Hence, one possible direction of future work is to further develop an efficient narrow band solver so that the computational cost of the algorithm can be further reduced.

Secondly, in the context of underwater path planning, one may wish to generate paths that optimize energy spent by the vehicles. We have focused on generation of time optimal vehicle tracks for this thesis. Therefore, a second possible direction of future work may be to develop schemes that can optimize the energy spent by the vehicles.

Thirdly, we have assumed in this work that the flow-field experienced by the

vehicle is exact. However, ocean fields are always associated with uncertainty and the exact flow-field may not always be known. Thus, a third possible direction is to extend our methodology to a stochastic setting to plan stochastic paths for vehicles.

The fourth and final direction of future work in this topic is to merge the level set based path planning algorithm with schemes for adaptive sampling and use it to plan paths for swarms of underwater vehicles. This will enable collaborative platforms to navigate optimally and contribute to better ocean prediction and monitoring.

Bibliography

Adalsteinsson, D. and Sethian, J. A. (1995). A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118(2):269 – 277.

Adalsteinsson, D. and Sethian, J. A. (1999). The fast construction of extension velocities in level set methods. *Journal of Computational Physics*, 148(1):2 – 22.

Agarwal, A. (2009). Statistical field estimation and scale estimation for complex coastal regions and archipelagos. Master's thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering.

Akkaya, N. Path finding using rapidly-exploring random trees. `http://nakkaya.com/2011/10/27/path-finding-using-rapidly-exploring-random-tree/`.

Alvarez, A., Caiti, A., and Onken, R. (2004). Evolutionary path planning for autonomous underwater vehicles in a variable ocean. *IEEE Journal of Oceanic Engineering*, 29(2):418–429.

Amato, C., Schurr, N., and Picciano, P. (2011). Towards a realistic decentralized modeling for use in a real-world personal assistant agent scenario. In *Proceedings of the 10^{th} International Joint Conference on Autonomous Agents and Multi-Agent Systems*.

Arkin, R. C. and Balch, T. (1998). Artificial intelligence and mobile robots. chapter Cooperative multiagent robotic systems, pages 277–296. MIT Press, Cambridge, MA, USA.

Bahr, A., Leonard, J. J., and Fallon, M. F. (2009). Cooperative localization for autonomous underwater vehicles. *The International Journal of Robotics Research*, 28(6):714–728.

Bakolas, E. and Tsiotras, P. (2010a). Minimum-time paths for a light aircraft in the presence of regionally-varying strong winds. In *Infotech at Aerospace*, Atlanta, GA. AIAA Paper 2010-3380.

Bakolas, E. and Tsiotras, P. (2010b). The Zermelo - Voronoi diagram: A dynamic partition problem. *Automatica*, 46(12):2059 – 2067.

Balch, T., Arkin, R. C., and Member, S. (1999). Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, 14:926–939.

Barraquand, J., Kavraki, L., Latombe, J.-C., Li, T.-Y., Motwani, R., and Raghavan, P. (1996). A random sampling scheme for path planning. *International Journal of Robotics Research*, 16:759–774.

Barraquand, J., Langlois, B., and Latombe, J. C. (1992). Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(2):224–241.

Barraquand, J. and Latombe, J. C. (1991). Robot motion planning - a distributed representation approach. *International Journal of Robotics Research*, 10(6):628–649.

Belta, C., Bicchi, A., Egerstedt, M., Frazzoli, E., Klavins, E., and Pappas, G. J. (2007). Symbolic planning and control of robot motion: Finding the missing pieces of current methods and ideas. *Robotics and Automation Magazine*, 14(1):61–70.

Belta, C. and Kumar, V. (2001). Motion generation for formations of robots: A geometric approach. In *Robotics and Automation Proceedings of. IEEE International Conference on*, volume 2, pages 1245 – 1250 vol.2.

Bendjilali, K., Belkhouche, F., and Belkhouche, B. (2009). Robot formation modelling and control based on the relative kinematics equations. *International Journal of Robotics and Automation*, 24(1):79–85.

Bhatta, P., Fiorelli, E., Lekien, F., Leonard, N. E., Paley, D. A., Zhang, F., Bachmayer, R., Davis, R. E., Fratantoni, D. M., and Sepulchre, R. (2005). Coordination of an underwater glider fleet for adaptive sampling. In *in Proceedings of International Workshop on Underwater Robotics*, pages 61–69.

Bhatta, P. and Leonard, N. E. (2002). Stabilization and coordination of underwater gliders. In *41st IEEE Conference on Decision and Control*, volume 2.

Binney, J., Krause, A., and Sukhatme, G. S. (2010). Informative path planning for an autonomous underwater vehicle. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4791–4796.

Bokanowski, O., Forcadel, N., and Zidani, H. (2010). Reachability and minimal times for state constrained nonlinear problems without any controllability assumption. *SIAM Journal of Optimal Control*, 48(7):4292–4316.

Bressan, A. (2011). *Viscosity solutions of Hamilton Jacobi equations and optimal control problems*. An Illustrated Tutorial, Penn State University.

Bruce, J. and Veloso, M. (2002). Real-time randomized path planning for robot navigation. In *Proceedings of IROS-2002*, Switzerland.

Bryson, A. E. and Ho, Y. C. (1975). *Applied Optimal Control: Optimization, Estimation and Control*. John Wiley and Sons.

Canny, J. F. (1988). *The complexity of robot motion planning*. MIT Press, Cambridge, MA, USA.

Carroll, K., McClaran, S., Nelson, E., Barnett, D., Friesen, D., and William, G. (1992). AUV path planning: An A* approach to path planning with consideration of variable vehicle speeds and multiple, overlapping, time-dependent exclusion zones. In *Autonomous Underwater Vehicle Technology, 1992. AUV '92., Proceedings of the 1992 Symposium on*, pages 79 –84.

Chaimowicz, L., Michael, N., and Kumar, V. (2005). Controlling swarms of robots using interpolated implicit functions. In *Robotics and Automation (ICRA), 2005 IEEE International Conference on*, pages 2487–2492.

Choi, H.-L. and How, J. P. (2010). Continuous trajectory planning of mobile sensors for informative forecasting. *Automatica*, 46(8):1266–1275.

Chopp, D. L. (1993). Computing minimal surfaces via level set curvature flow. *Journal of Computational Physics*, 106(1):77–91.

Chopp, D. L. (2009). Another look at velocity extensions in the level set method. *SIAM Journal of Scientific Computing*, 31(5):3255–3273.

Colella, P. and Puckett, E. G. (1994). *Modern Numerical Methods for Fluid Flow*. Lectures conducted at Department of Mechanical Engineering, University of California, Berkeley.

Consolini, L., Morbidi, F., Prattichizzo, D., and Tosques, M. (2007). A geometric characterization of leader-follower formation control. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2397 –2402.

Crandall, M. G., Evans, L. C., and Lions, P. L. (1984). Some properties of viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American Mathematical Society*, 282(2):487–502.

Crandall, M. G., Ishii, H., and Lions, P. L. (1992). User's guide to viscosity solutions of second order partial differential equations. *Bulletin of the American Mathematical Society*, 27:1–67.

Crandall, M. G. and Lions, P. L. (1983). Viscosity solutions of hamilton-jacobi equations. *Transactions of the American Mathematical Society*, 277:1–43.

Cushman-Roisin, B. and Beckers, J. (2010). *Introduction to Geophysical Fluid Dynamics. Physical and Numerical aspects*. Academic Press.

Davis, R. E., Leonard, N. E., and Fratantoni, D. M. (2009). Routing strategies for underwater gliders. *Deep Sea Research Part II: Topical Studies in Oceanography*, 56(35):173 – 187.

Derenick, J. and Spletzer, J. (2007). Convex optimization strategies for coordinating large-scale robot formations. *Robotics, IEEE Transactions on*, 23(6):1252 –1259.

Desai, J. P. (2002). A graph theoretic approach for modeling mobile robot team formations. *Journal of Robotic Systems*, 19(11):511–525.

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

Dijkstra, H. and Katsman, C. (1997). Temporal variability of the wind-driven quasi-geostrophic double gyre ocean circulation: Basic bifurcation diagrams. *Geophysics Astrophysics Fluid Dynamics*, 85:195–232.

Dudenhoeffer, D. D. and Jones, M. P. (2000). A formation behavior for large-scale micro-robot force deployment. In *Proceedings of the 32nd conference on Winter simulation*, WSC '00, pages 972–982, San Diego, CA, USA.

Elisseeff, P., Schmidt, H., Johnson, M., Herold, D., Chapman, N. R., and McDonald, M. M. (1999). Acoustic tomography of a coastal front in Haro Strait, British Columbia. *The Journal of the Acoustical Society of America*, 106(1):169–184.

Esin, Y. H., Unel, M., and Yildiz, M. (2008). Formation control of multiple robots using parametric and implicit representations. In *Proceedings of the 4th international conference on Intelligent Computing: Advanced Intelligent Computing Theories and Applications - with Aspects of Artificial Intelligence*, pages 558–565, Berlin, Heidelberg. Springer-Verlag.

Evans, L. C. (1998). *Partial Differential Equations*. Graduate Studies in Mathematics, Volume 19, American Mathematical Society.

Falcone, M. and Zidani, H. (2012). Numerical method for HJB equations. Optimal control problems and differential games. Lecture 3, ITN SADCO, Applied and Numerical Optimal Control.

Fierro, R., Song, P., Das, A., and Kumar, V. (2002). Cooperative control of robot formations. In *Cooperative Control and Optimization*, volume 66 of *Applied Optimization*, pages 73–93. Springer US.

Fiorelli, E., Leonard, N. E., Member, S., Bhatta, P., Paley, D. A., Member, S., Bachmayer, R., and Fratantoni, D. M. (2004). Multi-AUV control and adaptive sampling in Monterey Bay. In *IEEE Journal of Oceanic Engineering*, pages 935–948.

Fleming, W. and Soner, H. (2006). *Controlled Markov Processes and Viscosity Solutions*. Springer.

Fredslund, J. and Mataric, M. (2002). A general algorithm for robot formations using local sensing and minimal communication. *Robotics and Automation, IEEE Transactions on*, 18(5):837 – 846.

Garau, B., Alvarez, A., and Oliver, G. (2005). Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an A* approach. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 194–198.

Garau, B., Bonet, M., Alvarez, A., Ruiz, S., and Pascual, A. (2009). Path planning for autonomous underwater vehicles in realistic oceanic current fields: Application to gliders in the Western Mediterranean Sea. *Journal of Maritime Research*, 6(2):5–22.

Garrido, S., Moreno, L., and Blanco, D. (2006). Voronoi diagram and fast marching applied to path planning. In *Robotics and Automation, Proceedings 2006 IEEE International Conference on*, pages 3049 –3054.

Haley, P. J., Lermusiaux, P., Robinson, A., Leslie, W., Logoutov, O., Cossarini, G., Liang, X., Moreno, P., Ramp, S., Doyle, J., Bellingham, J., Chavez, F., and Johnston, S. (2009). Forecasting and reanalysis in the Monterey Bay/California current region for the Autonomous Ocean Sampling Network-ii experiment. *Deep Sea Research Part II: Topical Studies in Oceanography*, 56(35):127 – 148.

Hartmann, D., Meinke, M., and Schrder, W. (2010a). The constrained reinitialization equation for level set methods. *Journal of Computational Physics*, 229(5):1514 – 1535.

Hartmann, D., Meinke, M., and Schrder, W. (2010b). On accuracy and efficiency of constrained reinitialization. *International Journal for Numerical Methods in Fluids*, 63(11):1347–1358.

Hinchey, M., Sterritt, R., and Rouff, C. (2007). Swarms and swarm intelligence. *Computer*, 40(4):111 –113.

Inanc, T., Shadden, S. C., and Marsden, J. E. (2005). In *Proceedings of American Control Conference*, volume 1, pages 674–679.

Isern-Gonzlez, J., Hernndez-Sosa, D., Fernndez-Perdomo, E., Cabrera-Gmez, J., Domnguez-Brito, A. C., and Prieto-Maran, V. (2012). Obstacle avoidance in underwater glider path planning. *Journal of Physical Agents*, 6(1):11–20.

Jaillet, L., Cortes, J., and Simeon, T. (2010). Sampling-based path planning on configuration-space costmaps. *Robotics, IEEE Transactions on*, 26(4):635 –646.

Jarvis, R. (2006). Robot path planning: complexity, flexibility and application scope. In *Proceedings of the 2006 International Symposium on Practical Cognitive Agents and Robots*, PCAR '06, pages 3–14, New York, USA.

Jarvis, R. and Byrne, J. (1986). Robot navigation: Touching, seeing and knowing. In *Proceedings of the 1st Australian Conference on Artificial Intelligence*.

Jones, M. W., Baerentzen, J. A., and Sramek, M. (2006). 3D distance fields: A survey of techniques and applications. *Visualization and Computer Graphics, IEEE Transactions on*, 12(4):581–599.

Kalantar, S. and Zimmer, U. (2006). A formation control approach to adaptation of contour-shaped robotic formations. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1490 –1497.

Karaman and Frazzoli (2011). Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894.

Kruger, D., Stolkin, R., Blum, A., and Briganti, J. (2007). Optimal AUV path planning for extended missions in complex, fast-flowing estuarine environments. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4265–4270.

Kuffner, J. J. and LaValle, S. M. (2000). RRT-connect: An efficient approach to single-query path planning. In *Proceedings of IEEE International Confernce on Robotics and Automation*, pages 995–1001.

Latombe, J. C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA.

Latombe, J.-C. (1995). Controllability, recognizability, and complexity issues in robot motion planning. *Foundations of Computer Science, IEEE Annual Symposium on*, 0:484.

LaValle, S. M. The RRT page. `http://msl.cs.uiuc.edu/rrt/about.html`.

Lavalle, S. M. (1998). Rapidly-exploring Random Trees: A new tool for path planning. Technical report, Iowa State University.

Leonard, N. and Fiorelli, E. (2001). Virtual leaders, artificial potentials and coordinated control of groups. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 3, pages 2968 –2973.

Leonard, N. E., Paley, D., Lekien, F., Sepulchre, R., Fratantoni, D., and Davis, R. (2007). Collective motion, sensor networks, and ocean sampling. *Proceedings of the IEEE, special issue on the emerging technology of networked control systems*, (95):48–74.

Leonard, N. E., Paley, D. A., Davis, R. E., Fratantoni, D. M., Lekien, F., and Zhang, F. (2010). Coordinated control of an underwater glider fleet in an adaptive ocean sampling field experiment in monterey bay. *Journal of Field Robotics*, 27(6):718–740.

Lermusiaux, P. F. J. (2007). Adaptive modeling, adaptive data assimilation and adaptive sampling. *Physica D-Nonlinear Phenomena*, 230(1-2):172–196.

LeVeque, R. J. (2002). *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, Cambridge, U.K.

Li, C., Xu, C., Gui, C., and Fox, M. (2005). Level set evolution without re-initialization: a new variational formulation. In *Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society Conference on*, volume 1, pages 430 – 436.

Liu, S., Sun, D., and Zhu, C. (2011). Coordinated motion planning for multiple mobile robots along designed paths with formation requirement. *Mechatronics, IEEE/ASME Transactions on*, 16(6):1021 –1031.

Lolla, T., Ueckermann, M. P., Yigit, K., Haley, P. J., and Lermusiaux, P. F. J. (2012). Path planning in time dependent flow fields using level set methods. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 166–173.

McShane, E. J. (1937). A navigation problem in the calculus of variations. *American Journal of Mathematics*, 59(2):327–334.

Melchior, N. A. and Simmons, R. (2007). Particle RRT for path planning with uncertainty. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1617–1624.

Min, C. (2010). On reinitializing level set functions. *Journal of Computational Physics*, 229(8):2764 – 2772.

Mitchell, I. M., Bayen, A. M., and Tomlin, C. J. (2005). A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *Automatic Control, IEEE Transactions on*, 50(7):947 – 957.

Mulder, W., Osher, S., and Sethian, J. A. (1992). Computing interface motion in compressible gas dynamics. *Journal of Computational Physics*, 100:209–228.

Nishida, T., Sugihara, K., and Kimura, M. (2007). Stable marker-particle method for the Voronoi diagram in a flow field. *Journal of Computational and Applied Mathematics*, 202(2):377 – 391.

Oberhuber, T. (2004). Numerical recovery of the signed distance function. In *Proceedings of Czech-Japanese Seminar in Applied Mathematics 2004*, pages 148–164.

Ogren, P. and Leonard, N. E. (2003). Obstacle avoidance in formation. In *IEEE International Conference on Robotics and Automation*, pages 2492–2497.

Olfati-Saber, R. and Murray, R. M. (2002). Distributed cooperative control of multiple vehicle formations using structural potential functions. In *The 15th International Federation of Automatic Control World Conference, Barcelona, Spain.*

Osher, S. and Fedkiw, R. (2003). *Level Set Methods and Dynamic Implicit Surfaces*. Springer Verlag.

Osher, S. and Sethian, J. A. (1988). Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12 – 49.

Paley, D., Leonard, N. E., and Sepulchre, R. (2006). Collective Motion of Self-Propelled Particles: Stabilizing Symmetric Formations on Closed Curves. In *45th IEEE Conference on Decision and Control*, pages 5067–5072.

Paley, D. A., Zhang, F., and Leonard, N. E. (2008). Cooperative control for ocean sampling: The glider coordinated control system. *IEEE Transactions on Control Systems Technology*, 16(4):735–744.

Pedlosky, J. (1998). *Ocean Circulation Theory*. Springer-Verlag.

Persson, P.-O. (2010). *2.097: Numerical Methods for Partial Differential Equations*. Lectures conducted at MIT, Cambridge, MA.

Petres, C., Pailhas, Y., Patron, P., Petillot, Y., Evans, J., and Lane, D. (2007). Path planning for autonomous underwater vehicles. *Robotics, IEEE Transactions on*, 23(2):331 –341.

Rao, D. and Williams, S. B. (2009). Large-scale path planning for underwater gliders in ocean currents. In *Proceedings of Australasian Conference on Robotics and Automation*.

Reif, J. and Sharir, M. (1994). Motion planning in the presence of moving obstacles. *Journal of the Association for Computing Machinery (ACM)*, 41(4):764–790.

Reif, J. H. (1979). Complexity of the mover's problem and generalizations. In *Foundations of Computer Science, 1979., 20th Annual Symposium on*, pages 421 –427.

Reif, J. H. and Wang, H. (1999). Social potential fields: A distributed behavioral control for autonomous robots. *Robotics and Autonomous Systems*, 27:171–194.

Ren, W. and Sorensen, N. (2008). Distributed coordination architecture for multi-robot formation control. *Robotics and Autonomous Systems*, 56(4):324–333.

Rhoads, B., Mezic, I., and Poje, A. (2010). Minimum time feedback control of autonomous underwater vehicles. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 5828 –5834.

Russo, G. and Smereka, P. (2000). A Remark on Computing Distance Functions. *Journal of Computational Physics*, 163:51–67.

Sabattini, L., Secchi, C., and Fantuzzi, C. (2011). Arbitrarily shaped formations of mobile robots: artificial potential fields and coordinate transformation. *Autonomous Robots*, 30(4):385–397.

Schmidt, H., Bellingham, J. G., Johnson, M., Herold, D., Farmer, D., and Pawlowicz, R. (1996). Real-time frontal mapping with AUVs in a coastal environment. In *OCEANS '96. MTS/IEEE. Prospects for the 21st Century. Conference Proceedings*, volume 3, pages 1094 –1098.

Senatore, C. and Ross, S. (2008). Fuel-efficient navigation in complex flows. In *American Control Conference, 2008*, pages 1244 –1248.

Sethian, J. A. (1999a). Fast marching methods. *SIAM Rev.*, 41(2):199–235.

Sethian, J. A. (1999b). *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press: Cambridge, U.K.

Simmonet, E., Dijkstra, H., and Ghil, M. (2009). *Bifurcation Analysis of Ocean, Atmosphere, and Climate Models. In 'Computational Methods for the Atmosphere and the Oceans'*, volume XIV, p. 187-229. Handbook of Numerical Analysis.

Sisto, M. and Gu, D. (2006). A fuzzy leader-follower approach to formation control of multiple mobile robots. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2515 –2520.

Smith, R. N., Chao, Y., Li, P. P., Caron, D. A., Jones, B. H., and Sukhatme, G. S. (2010). Planning and implementing trajectories for autonomous underwater vehicles to track evolving ocean processes based on predictions from a regional ocean model. *International Journal of Robotic Research*, 29(12):1475–1497.

Sod, G. A. (1985). *Numerical Methods in Fluid Dynamics*. Cambridge-University Press, Cambridge, UK.

Soulignac, M. (2011). Feasible and optimal path planning in strong current fields. *IEEE Transactions on Robotics*, 27(1):89–98.

Soulignac, M., Taillibert, P., and Rueher, M. (2009). Time-minimal path planning in dynamic current fields. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 2473 –2479.

Stommel, H. (1989). The slocum mission. *Oceanography*, pages 22–25.

Sussman, M., Smereka, P., and Osher, S. (1994). A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow. *Journal of Computational Physics*, 114:146–159.

Swaminathan, K. and Minai, A. A. (2005). *Stigmergic Optimization*. Spring-Verlag.

Tan, C. S., Sutton, R., and Chudley, J. (2004). An incremental stochastic motion planning technique for autonomous underwater vehicles. In *IFAC Control Applications in Marine Systems Conference*, pages 483–488.

Techy, L. and Woolsey, C. A. (2009). Minimum-time path planning for unmanned aerial vehicles in steady uniform winds. *Journal of Guidance, Control, and Dynamics*, 32(6):1736–1746.

Thompson, D. R., Chien, S. A., Chao, Y., Li, P., Arrott, M., Meisinger, M., Balasuriya, A. P., Petillo, S., and Schofield, O. (2009). Glider Mission Planning in a Dynamic Ocean Sensorweb. In *SPARK Workshop on Scheduling and Planning Applications, International Conference on Automated Planning and Scheduling*.

Thompson, D. R., Chien, S. A., Chao, Y., Li, P., Cahill, B., Levin, J., Schofield, O., Balasuriya, A. P., Petillo, S., Arrott, M., and Meisinger, M. (2010). Spatiotemporal path planning in strong, dynamic, uncertain currents. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 4778–4783.

Tsitsiklis, J. N. (1995). Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538.

Ueckermann, M. P. and Lermusiaux, P. F. J. (2009). *2.29: Numerical Fluid Mechanics*. Finite Volume MATLAB Framework, Cambridge, MA.

Ueckermann, M. P. and Lermusiaux, P. F. J. (2011). 2.29 Finite Volume MATLAB Framework Documentation. Technical report, Massachusetts Institute of Technology, Cambridge, MA USA.

Ueckermann, M. P., Lermusiaux, P. F. J., and Sapsis, T. P. (2012). Numerical schemes for dynamically orthogonal equations of stochastic fluid and ocean flows. *submitted to Journal of Computational Physics*.

Urmson, C. and Simmons, R. (2003). Approaches for heuristically biasing RRT growth. In *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), Proceedings of*, volume 2, pages 1178 –1183.

Van Leer, B. (1977). Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection. *Journal of Computational Physics*, 23(3):276 – 299.

Vasudevan, C. and Ganesan, K. (1996). Case-based path planning for autonomous underwater vehicles. *Autonomous Robots*, 3:79–89.

Warren, C. W. (1990). A technique for autonomous underwater vehicle route planning. In *Autonomous Underwater Vehicle Technology, Proceedings of the Symposium on*, pages 201–205.

Wikipedia. A* search algorithm. `http://en.wikipedia.org/wiki/A*_search_algorithm`.

Witt, J. and Dunbabin, M. (2008). Go with the flow: Optimal auv path planning in coastal environments. In *Proceedings of Australasian Conference on Robotics and Automation*.

Yang, H. and Zhang, F. (2010). Geometric formation control for autonomous underwater vehicles. In *Proceedings of 2010 International Conference on Robotics and Automation*, pages 4288–4293, Anchorage, AL.

Yang, H. and Zhang, F. (2012). Robust control of formation dynamics for autonomous underwater vehicles in horizontal plane. *ASME Journal of Dynamic Systems, Measurement and Control*, 134(3).

Yang, K., Gan, S., and Sukkarieh, S. (2010). An efficient path planning and control algorithm for ruavs in unknown and cluttered environments. *Journal of Intelligent and Robotic Systems*, 57:101–122.

Yigit, K. (2011). *Path Planning Methods for Autonomous Underwater Vehicles*. S.M. Thesis, Massachusetts Institute of Technology, Cambridge MA.

Yilmaz, N. K., Evangelinos, C., Lermusiaux, P., and Patrikalakis, N. M. (2008). Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming. *IEEE Journal of Oceanic Engineering*, 33(4):522–537.

Yu, C.-H. and Nagpal, R. (2008). Sensing-based shape formation on modular multi-robot systems: A theoretical study. In *Proceedings of* 7^{th} *International Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2008)*.

Zermelo, E. (1931). ´Uber das navigationsproblem bei ruhender oder ver´anderlicher windverteilung. *Z. Angew. Math. Mech*, 11:114–124.

Zhang, F. (2007). Cooperative shape control of particle formations. In *Decision and Control, 2007 46th IEEE Conference on*, pages 2516 –2521.

Zhang, F., Fratantoni, D. M., Paley, D., Lund, J., and Leonard, N. E. (2007). Control of coordinated patterns for ocean sampling. *International Journal of Control*, 80(7):1186–1199.

Zhang, F. and Leonard, N. (2006). Coordinated patterns on smooth curves. In *Proc. of 2006 IEEE International Conf. on Networking, Sensing and Control*, pages 434–439, Ft. Lauderdale, Florida.

Zhang, W., Inane, T., Ober-Blobaum, S., and Marsden, J. E. (2008). Optimal trajectory generation for a glider in time-varying 2d ocean flows b-spline model. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1083 –1088.